ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਪੰਜਾਬ ਸਟੇਟ ਓਪਨ ਯੂਨੀਵਰਸਿਟੀ
ਪਟਿਆਲਾ

The Motto of Our University

(SEWA)

SKILL ENHANCEMENT

EMPLOYABILITY

WISDOM

ACCESSIBILITY

SELF-INSTRUCTIONAL STUDY MATERIAL FOR JGND PSOU

JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

B.Sc.(Data Science)

Semester I

Head Quarter: C/28, The Lower Mall, Patiala-147001

Website: www.psou.ac.in

The Study Material has been prepared exclusively under the guidance of Jagat Guru Nanak Dev Punjab State Open University, Patiala, as per the syllabi prepared by Committee of experts and approved by the Academic Council.

COURSE COORDINATOR AND EDITOR:

Dr. Amitoj Singh
Associate Professor
School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University

Code	Subject	Nature of Course	Credits
BSDB31101T	Problem Solving using Computer	CC-1	4
BSDB31102T	Fundamental of IT	DSC-1	4
BSDB31103T	Introduction Data Science	DSC-2	4
BSDB31101P	Problem Solving using Computer Lab	CC-1	2
BSDB31102P	Fundamental of IT Lab	DSC-3	2
BSDB31103P	Introduction Data Science Lab	DSC-4	2
AE1B31104T	Effective Communication in English	AECC-1	4

ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਪੰਜਾਬ ਸਟੇਟ ਓਪਨ ਯੂਨੀਵਰਸਿਟੀ
ਪਟਿਆਲਾ

The Motto of Our University
(SEWA)
SKILL ENHANCEMENT
EMPLOYABILITY
WISDOM
ACCESSIBILITY

SELF-INSTRUCTIONAL STUDY MATERIAL FOR JGND PSOU

JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

B.Sc.(Data Science)

Semester I

BSDB31103T

Introduction to Data Science

Head Quarter: C/28, The Lower Mall, Patiala-147001
Website: www.psou.ac.in

The Study Material has been prepared exclusively under the guidance of Jagat Guru Nanak Dev Punjab State Open University, Patiala, as per the syllabi prepared by Committee of experts and approved by the Academic Council.

COURSE COORDINATOR AND EDITOR:

Dr. Amitoj Singh
Associate Professor
School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University

LIST OF CONSULTANTS/ CONTRIBUTORS

Sr. No.	Name
1	Dr. Preety Dubey
2	Dr. Chetan Dudhagara
3	Mr. Sandeep Kumar

JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA
(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

PREFACE

Jagat Guru Nanak Dev Punjab State Open University, Patiala was established in December 2019 by Act 19 of the Legislature of State of Punjab. It is the first and only Open University of the State, entrusted with the responsibility of making higher education accessible to all, especially to those sections of society who do not have the means, time or opportunity to pursue regular education.

In keeping with the nature of an Open University, this University provides a flexible education system to suit every need. The time given to complete a programme is double the duration of a regular mode programme. Well-designed study material has been prepared in consultation with experts in their respective fields.

The University offers programmes which have been designed to provide relevant, skill-based and employability-enhancing education. The study material provided in this booklet is self-instructional, with self-assessment exercises, and recommendations for further readings. The syllabus has been divided in sections, and provided as units for simplification.

The University has a network of 10 Learner Support Centres/Study Centres, to enable students to make use of reading facilities, and for curriculum-based counselling and practicals. We, at the University, welcome you to be a part of this instituition of knowledge.

Prof. Anita Gill
Dean Academic Affairs

B.Sc. (Data Science)
Core Course (CC)
Semester I
BSDB31103T: Introduction to Data Science

Total Marks: 100
External Marks: 70
Internal Marks: 30
Credits: 4
Pass Percentage: 35%

Objective

To provide strong foundation for data science and application area related to it and understand the underlying core concepts and emerging technologies in data science. Understand data analysis techniques for applications handling large data.

INSTRUCTIONS FOR THE PAPER SETTER/EXAMINER

1. The syllabus prescribed should be strictly adhered to.
2. The question paper will consist of three sections: A, B, and C. Sections A and B will have four questions from the respective sections of the syllabus and will carry 10 marks each. The candidates will attempt two questions from each section.
3. Section C will have fifteen short answer questions covering the entire syllabus. Each question will carry 3 marks. Candidates will attempt any ten questions from this section.
4. The examiner shall give a clear instruction to the candidates to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.
5. The duration of each paper will be three hours.

INSTRUCTIONS FOR THE CANDIDATES

Candidates are required to attempt any two questions each from the sections A and B of the question paper and any ten short questions from Section C. They have to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.

Section A

Unit I: Data Science-a **Discipline**, Landscape-Data to Data science, Data Growth-issues and challenges, data science process. foundations of data science. Messy data, Anomalies and artefacts in datasets. Cleaning data.

Unit II: Data Acquisition and Processing: introduction, Structured Vs Unsructured data, data preprocessing techniques including data cleaning, selection, integration, transformation and reduction, data mining, interpretation.

Unit III: Representation of Data: Special types-acoustic, image, sensor and network data. Problems when handling large data – General techniques for handling large data, Distributing data storage and processing with Frameworks

Unit IV: **Data Science Ethics** – Doing good data science – Owners of the data - Valuing different aspects of privacy - Getting informed consent - The Five Cs – Diversity – Inclusion – Future Trends.

Section B

Unit V: Data Wrangling Combining and Merging Data Sets – Reshaping and Pivoting – Data Transformation – String manipulations – Regular Expressions

Unit VI: Data Aggregation and Group Operations Group By Mechanics – Data Aggregation – GroupWise Operations – Transformations – Pivot Tables – Cross Tabulations – Date and Time data types.

Unit VII: Data Modeling: Basics of Generative modeling and Predictive modeling. Charts-histograms, scatter plots, time series plots etc. Graphs, 3D Visualization and Presentation.

Unit VIII: Applications of Data Science: Business, Insurance, Energy, Health care, Biotechnology, Manufacturing, Utilities, Telecommunication, Travel, Governance, Gaming, Pharmaceuticals, Geospatial analytics and modeling

Suggested Readings

1. Sinan Ozdemir, Principles of Data Science, Packt Publishing, 2016
2. Joel Grus: Data Science from Scratch, O'Reilly, 2016
3. Foster Provost & Tom Fawcett: Data Science for Business O'Reilly, 2013
4. Roger D. Peng & Elizabeth Matsui: The Art of Data Science, Lean Publishing, 2015
5. Peter Bruce, Andrew Bruce, Peter Gedeck, Practical Statistics for Data Scientists, 2e: 50+ Essential Concepts Using R and Python, O′Reilly

JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA
(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

BSDB31103T: INTRODUCTION TO DATA SCIENCE

COURSE COORDINATOR AND EDITOR: DR. AMITOJ SINGH

UNIT NO.	UNIT NAME
UNIT 1	DATA SCIENCE-A DISCIPLINE
UNIT 2	DATA ACQUISITION AND PROCESSING
UNIT 3	REPRESENTATION OF DATA
UNIT 4	DATA SCIENCE ETHICS
UNIT 5	DATA WRANGLING COMBINING AND MERGING DATA
UNIT 6	DATA AGGREGATION AND GROUP OPERATIONS GROUP BY MECHANICS
UNIT 7	DATA MODELING
UNIT 8	APPLICATIONS OF DATA SCIENCE

B.Sc.(DATA SCIENCE)

SEMESTER-I

INTRODUCTION TO DATA SCIENCE

UNIT I: FOUNDATION OF DATA SCIENCE

STRUCTURE

1.0 OBJECTIVES

1. Introduction to Data Science
2. Familiarize with the issues and challenges in Data Growth
3. Familiarize with data Science Process
4. Familiarize with the concepts of data like clean and messy data, data artifacts and anomalies in data.

1.1 DATA SCIENCE-A DISCIPLINE

Data Science is a blend of various tools, algorithms and machine learning principles with the goal to discover hidden patterns from raw data. It is related to data mining, machine learning and Big Data It is an area that manages, manipulates, extracts, and interprets knowledge from tremendous amount of data. Data science (DS) is a multidisciplinary field of study with a goal to address the challenges in big data. Big Data has given rise to Data Science and is a therefore, a multidisciplinary field of study with goal to address the challenges in big data.

1.2 DATA TO DATA SCIENCE

The growth of data has been seen since 2010, due to the growth in the number of data generating devices like smart phones, wearables, Internet of things, etc. The availability of more data publicly from social media sites like Facebook, YouTube, twitter etc, business transactions, sensors, audio, video, photos etc. This enormous growth of data led to the concept of Big data, which is a term used for collection of large and complex data sets. As the data has increased, so did the need for its storage. Until 2010, the main focus was building framework and solutions to store data, which was successfully solved by HADOOP and other frameworks. In the present time, it has become difficult to process this large and complex data using traditional data management techniques such as, for example, the RDBMS (relational database management systems). This rise in the use of data, sparked the use of Data Science. Data science makes this possible, as it is a multidisciplinary study of data collection for analysis, prediction, learning and prevention. Data Science involves using methods to analyse massive amounts of data and extract the knowledge from the raw data.

1.3 DATA GROWTH ISSUES AND CHALLENGES

Big Data is characterized by five V's namely *volume*, *variety*, *velocity*, *veracity* and *value*. Consequently, the challenges these characteristics bring are being seen in *data capture*, *curation*, *storage*, *search*, *sharing*, *transfer*, and *visualization*.

i. **Volume** refers to the enormous size of data. Big data refers to data volumes in the range of exabytes and beyond e.g. In the year 2016, the estimated global mobile traffic was 6.2 Exabytes (6.2 billion GB) per month. Also, by the year 2020 we will have almost 40000 Exabytes of data. Such volumes exceed the capacity of current on-line storage systems and processing systems. Traditional database systems is not able to capture, store and analyse this large amount of data. Storage solutions have been provided by HADOOP framework, but represent long term challenges that require research and new paradigms.

ii. **Velocity** refers to the high speed of accumulation of data. There is a massive and continuous flow of data from sources like machines, networks, social media, mobile

phones etc. This determines the potential of data that how fast the data is generated and processed to meet the demands e.g. there are more than 3.5 billion searches per day are made on Google. Also, FaceBook users are increasing by 22%(Approx.) year by year. Data is streaming in at unprecedented speed and must be dealt with in a timely manner. RFID tags, sensors and smart metering are driving the need to deal with torrents of data in near-real time. The data has to be available at the right time to mske business decisions accurately. Reacting quickly enough to deal with data velocity is a challenge for most organizations

iii. **Variety** refers to nature of data. Data is available in varied formats and heterogenous sources. It can be structured, semi-structured and unstructured data. It is a challenge to find ways of governing, merging and managing these diverse forms of data.

iv. **Veracity** here means quality of data. It refers to inconsistencies and uncertainty in data. The available data may be messy and thus controlling the quality and accuracy becomes another challenge.

v. **Variability:** In addition to the increasing velocities and varieties of data, data flows can be highly inconsistent with periodic peaks. Big Data is also variable because of the multitude of data dimensions resulting from multiple disparate data types and sources e.g. Data in bulk could create confusion whereas less amount of data could convey half or Incomplete Information. Variability of data can be challenging to manage.

vi. **Value:** The bulk Data having no Value is of no good to the company, unless you turn it into something useful. Data in itself is of no use or importance but it needs to be converted into something valuable to extract Information. The ability to transform the bulk data into business and make this enormous data of use to business to monetize. It is a challenge to connect and correlate relationships, hierarchies and multiple data linkages of the big data.

1.4 FOUNDATION OF DATA SCIENCE

Data science involves using methods to analyse massive amounts of data and extract the knowledge it contains. Data Science is an interdisciplinary field focused on extracting knowledge from datasets which are large in size, applying the knowledge from data to solve problems in a wide range of application domains. Mathematics, statistics, computer science, and domain knowledge are the foundations of Data Science.

The main components of Data Science are given below:

1. Statistics: To analyse the large amount numerical data and to find the meaningful insights from it, knowledge of statistics is required.

2. Domain Expertise: Data is available and applicable in various domains, therefore domain expertise or specialized knowledge of a specific area is required to get best results.

3. Data engineering: Data engineering is a part of data science, which involves acquiring, storing, retrieving, and transforming the data. Data engineering also includes metadata (data about data) to the data.

4. Visualization: Data visualization is meant by representing data in a visual context so that people can easily understand the significance of data. Data visualization makes it easy to access the huge amount of data in visuals.

5. Advanced computing: Advanced computing involves designing, writing, debugging, and maintaining the source code of computer programs. Machine Learning and Deep learning techniques are required for modelling and to make predictions about unforeseen/future data.

1.5 TOOLS FOR DATA SCIENCE

Some tools required for data science are as follows:

- **Data Analysis tools:** R, Python, Statistics, SAS, Jupyter, R Studio, MATLAB, Excel, RapidMiner.
- **Data Warehousing:** ETL, SQL, Hadoop, Informatica/Talend, AWS Redshift
- **Data Visualization tools:** R, Jupyter, Tableau, Cognos.
- **Machine learning tools:** Spark, Mahout, Azure ML studio.

1.6 APPLICATIONS OF DATA SCIENCE

Data Science is used by most organizations for predictive analysis, price optimization, and customer satisfaction. Some areas where data science is being used are Health Care, Finance, Security, Airline Routing, Manufacturing, Speech Recognition, Advertisement, Security, Fraud detection, Banking, Internet of Things etc. Some use cases are given below:

- Genomic Data provides deeper understanding of Genetic issues and reactions to particular drugs and diseases.
- Logistics companies like DHL,Fedex have discovered the best time and routes to ship cost effectively.
- Predict employee attrition and understand the variables that influence employee turnover.
- Airline companies can now easily predict flight delays and notify the passengers before time.
- Banks can make better decisions by predict risk analysis, fraud detection and better customer management

1.7 DATA SCIENCE PROCESS

The structured approach to data science helps in maximizing the chances of success in a data science project at the lowest cost. The data science process typically consists of the following steps:

I. **Business Understanding**: The main purpose here is making sure all the stakeholders understand the what, how, and why of the project. It involves two main steps namely *defining research goals* and *preparing a project charter*.

i. Defining Research Goals: It is very essential to understand the business goals to be able to define the research goal that states the purpose of assignment in a clear and focused manner. The data by for the same can be gathered by repeatedly asking questions and enquiring about business expectations, identifying the research goals so that everybody knows what to do and can agree on the best course of action. The outcome should be a clear research goal, a good understanding of the context, well-defined deliverables, and a plan of action with a timetable. This information is then best placed in a project charter

ii. Creating Project Charter: A project charter has the information gathered while setting the research goal. The project charter must contain a clear research goal, the project mission and context, method for analysis, information about the resources required for project completion, proof that the project is achievable, or proof of concepts, deliverables and a measure of success and also a timeline for the project.

This information is useful to make an estimation of the project costs and the data and people required for the project to become a success.

II. **Data Acquisition or Data Collection or Data Retrieval:** This phase involves data gathering from various sources like webservers, logs, databases, API's and online repositories. It involves acquiring data from all the identified internal & external sources. One should start with data collection from internal sources. The data may be available in many forms ranging from simple text to database records. The data may be structured as well as unstructured. Data may be acquired from sources outsise the organization also. Some sources may be available free of cost or some may be paid. Data collection is a tiresome and time-consuming task.

III. **Data preparation:** Data collection is an error-prone process; in this phase you enhance the quality of the data and prepare it for use in subsequent steps. This phase consists of three subphases: **data cleansing, data integration data transformation.**

i) ***Data Cleansing*** involves cleansing of data collected in the previous phase. It involves removing false values from a data source, removing inconsistencies across data sources, checking misspelled attributes, missing and duplicate values and typing errors, fixing capital letter mismatches as most programming

languages are case sensitive (e.g India and india), removing redundant white spaces, sanity checks (check for impossible values like six-digit phone number etc) and outliers (an outlier is an observation that seems to be distant from other observations). It should be ensured that most errors are corrected in this phase to make the data usable and get better results.

ii) ***Data Integration*** enriches data sources by combining information from multiple data sources. Data comes from several sources and in this sub step the focus is on integrating these different sources. The data can be combined in the following ways:

- ***Joining Tables:*** To combine information about some data in one table with the information available in another table. e.g a table may contain information about purchase of a particular product and the other table may contain information about people who have purchased that product. This can be done using join command in SQL.
- ***Appending or Stacking:*** To add observations from one table to another table e.g. a table may contain the information about the purchase of a particular product in the year 2000 and the other contains the similar data in the year 2001, then appending means to add the records of 2001 to the table containing the records of 2000. This can be done using the union function in SQL.
- ***View:*** View in SQL can be used to virtually combine two tables. This saves on the space requirement
- ***Aggregate Functions:*** Aggregate functions may be used as per requirement for combining data.

iii) ***Data transformation*** ensures that the data is in a suitable format to be used in the project model. Sometimes, the number of variables may have to be reduced, It involves modification of data so that it takes a suitable shape. Tools like talend and informatica can be used for transformation.

IV. Exploratory Data Analysis (EDA) or Data exploration: Data exploration is concerned with building a deeper understanding of the data. It involves the understanding of how variables interact with each other, the distribution of the data, and whether there are outliers. It involves defining and refining the selection of feature variables that will be used in model development. This is the most important step as it involves understanding of the data which will further be used for modelling. Various techniques like visualization, tabulation, clustering, and other modelling techniques can be used for exploratory analysis.

V. Data Modelling or Model building: Model building is an iterative process. In this phase, domain knowledge, and insights about the data are used for modelling. It involves identifying the data model that best fits the business requirement. For this, various machine learning techniques like KNN, Naïve's, decision tree etc may be applied on data. The model is trained on the data set and the testing of the selected

model is done. Data modelling can be done using Python, R, SAS. Most models consist of the following main steps:

i. ***Selection of a modelling technique and variables to enter in the model:*** After the exploratory analysis, the variables to be used are known, so in this phase, the variables can be used to build the model. A model that suits the project requirement has to be chosen.

ii. ***Execution of the model***: The chosen model has to be implemented by coding. Python is most used language for coding as it has many inbuilt libraries.

iii. **Diagnosis and model comparison:** In this step, the best performing model or the model with lowest errors is chosen from among the multiple models that are built.

VI. Presentation and automation: In this stage, the results are presented. These results can take many forms, ranging from presentations to research reports.

VII. Deployment & Maintenance: Before the final deployment in the production environment, testing is done in preproduction environment. After deployment, the reports are used for real time analysis.

The Data Science life cycle is an Iterative process, there is often a need to step back and rework certain findings. If the step 1(business understanding) is performed dedicatedly, rework can be prevented. The figure 1 below describes the Data Science model.

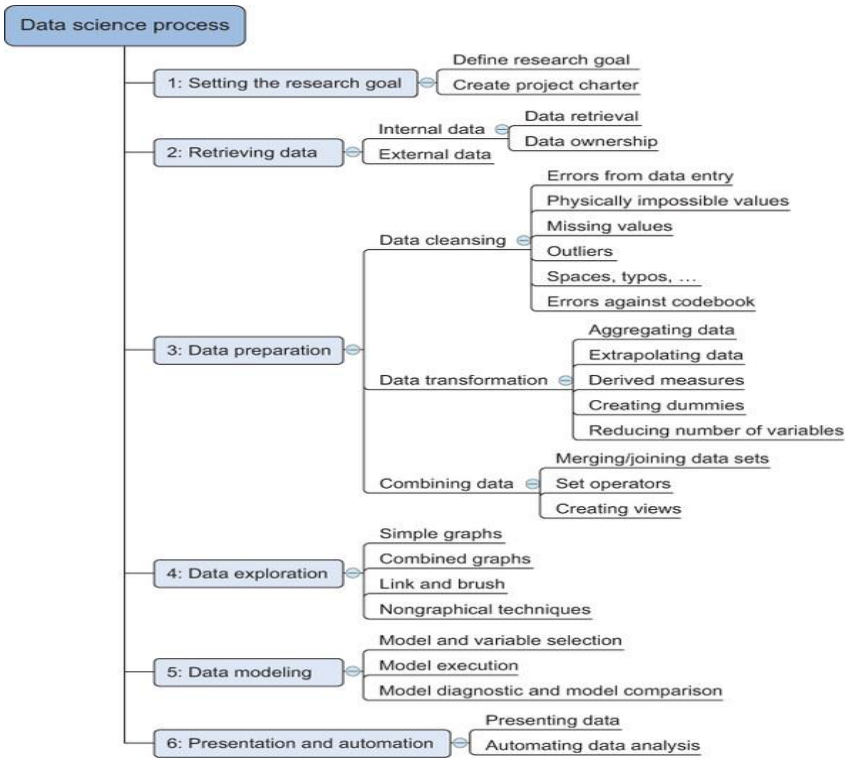


Fig1source: https://livebook.manning.com/book/introducing-data-science/chapter-2/8

1.8 MESSY DATA

Data that is not in usable form or it is impossible to obtain clearly interpretable information from it is messy data. Data science and its algorithms are clean and precise, but the data on which they operate come from the real world, are inherently messy i.e. the data which requires some preparation before you can use them effectively and it is not easy to find clean data. The quality of insights you derive from data depends on the validity of that data, so some preparation is required. Some examples of messy data are missing data, unstructured data, multiple variables in one column, variables stored in wrong places, observations split incorrectly or left together against normalization rules, switched columns and rows, extra spaces etc.

1.9 ANOMALIES AND ARTEFACTS IN DATASETS

Anomaly is a deviation in data from the expected value for a metric at a given point in time. Anomaly detection is any process that finds the outliers (items that do not belong to the dataset) of a dataset. The term anomaly is also referred to as outlier. Outliers are the data objects that stand out among other objects in the data set and do not conform to the normal behaviour in a data set. There are three kinds of anomalies namely: point anomaly, contextual anomaly, and collective anomalies.

- **Point Anomaly:** If a single instance in a given dataset is different from others with respect to its attributes, it is called a point anomaly i.e. when a single instance of data is anomalous, it deviates largely from the rest of the set e.g. detecting credit card fraud based on "amount spent."
- **Contextual anomaly:** If the data is anomalous in some context, it is called contextual anomaly. This type of anomaly is common in time-series data. In the absence of a context, all the data points look normal. E.g. if the context of the temperature is recorder in December and a high temperature reading is seen in December month, which is an abnormal phenomenon.
- **Collective anomalies** can be formed due to a combination of many instances i.e. a set of data instances collectively helps in detecting anomalies. For example, sequence data in network log or an attempt to copy data form a remote machine to a local host unexpectedly, an anomaly that would be flagged as a potential cyber-attack.

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behaviour. It is a technique for finding an unusual point or pattern in a given set. These nonconforming patterns are often referred to as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities, or contaminants in different application domains. Anomaly detection is commonly used for:

- Data cleaning
- Intrusion detection
- Fraud detection
- Systems health monitoring

- Event detection in sensor networks
- Ecosystem disturbances

Artifact It is a data flaw caused by equipment, techniques or conditions. Common sources of data flaws include hardware or software errors, conditions such as electromagnetic interference and flawed designs such as an algorithm prone to miscalculations. Some common data artifacts are:

- **Digital Artifacts:** Flaws in digital media, documents and data records caused by data processing errors e.g a distorted camera recording.
- **Visual Artifacts:** Flaws in visualizations such as user interfaces.
- **Compression Artifacts**: Flaws in data due to lossy compression.
- **Statistical Artifacts**: Flaw such as a bias in statistical data.
- **Sonic Artifacts**: Unwanted sound in a recording.

1.10 CLEANING DATA

Data cleaning is a key part of data science. Clean data increases overall productivity and allows for the highest quality information in decision-making. Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. If data is messy, the outcomes and algorithms are unreliable. It can lead to poor business strategy and decision-making. The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc. During cleansing, missing values may either be filled or removed depending on the data. Data cleaning is discussed in detail in Unit -II

1.11 SUMMARY

This chapter introduces the concept of data science as a discipline. The various issues faced due to the growth in data are discussed. The importance of cleaning data and the anomalies found in data are also discussed.

1.12 PRACTICE QUESTIONS

Q1. What are the foundations of data science?

Q2. Discuss the areas where data science is applicable?

Q3. Discuss the various challenges due to growth in data.

Q4. Explain briefly the data Science Process.

Q5. Define Messy data.

B.Sc.(DATA SCIENCE)

SEMESTER-I

INTRODUCTION TO DATA SCIENCE

UNIT II: DATA ACQUISITION AND PROCESSING

STRUCTURE

2.0 OBJECTIVES

1. Introduction to Data Acquisition

2. familiarize with the different types of data

3. Provide the concept of data pre-processing and familiarize with the various reprocessing techniques

4.Familiarize with the concept of data mining and data interpretation

2.1 INTRODUCTION TO DATA ACQUISITION

The process of gathering data and making it useful by filtering and cleaning it as per the business requirement is termed as data acquisition It is the most important step of data science, but the data acquired must be suitable to the problem in hand, else the output may be inaccurate. It is very important to acquire up to date data. The characteristics of big data namely volume, velocity, variety, and value and very important for the acquisition of data. In data science, there is a variety of data that we need to deal with. The data is majorly characterized as:

- **Structured Data:** The data which is available in some stand format is called structured data. Structured data is organized data. It is stored in the row and column structure like in a relational database and excel sheets. Some examples of structured data are names, numbers, geolocations, addresses etc.

- **Unstructured Data**: This data is unorganized data. There is no particular format for unstructured data. It is available in a variety of formats like texts, pictures, videos etc. Unstructured data is more difficult to search and requires processing to become understandable.

- **Semi- Structured data:** This data is basically a semi-organised data. It is generally a form of data that do not conform to the formal structure of data. Log files are the examples of this type of data.

2.2 DATA PREPROCESSING AND TECHNIQUES

The data acquired need to be preprocessed to make it usable. The raw data may be inconsistent, erroneous, incomplete and not in the required format. These issues need to be resolved to make the data usable. Data Preprocessing is a collaborative term used for the activities involved in transforming the real-world data or raw data into a usable form to make it more valuable and to get it in the required format. The preprocessed data is cleaner and more valuable, and hence used as final training set. Data preprocessing is a very essential step, as more clean and inconsistent data we get, better shall be the final output. In other words, data processing improves the quality of data. The huge size of data collected from heterogenous sources leads to anomalous data. Data preprocessing has become a vital and most fundamental step considering the fact that high quality data leads to better models and predictions. Data preprocessing techniques are majorly categorized in the following methods:

i. Data Cleaning
ii. Data Integration
iii. Data Transformation
iv. Data Reduction

2.2.1 Data Cleaning involves removing duplicate data, filling missing values, identifying outliers, smoothening noise and remove data inconsistencies. The following steps must be followed for cleaning data:

- **Remove duplicate or irrelevant observations**
  It is very important to remove inconsistencies in dataset like removing unwanted observations including duplicate and irrelevant data entries. Duplicate values may be caused at the time of data collection e.g. the names of the countries may have repeated valued like *NewZealand* and *New Zealand*; *Pakistan* and *pakistan*. Inconsistencies in capitalizations and trailing white spaces are very common in text data, The data set can be cleaned using available function in Python or R. functions like unique, sort, lower(), upper(), strip() can be used to handle inconsistencies

- **Fix structural errors:** Structural errors are when you measure or transfer data and notice strange naming conventions, typos, or incorrect capitalization. These inconsistencies can cause mislabelled categories or classes. For example, you may find "N/A" and "Not Applicable" both appear, but they should be analysed as the same category. Set the single date format in case there are multiple date formats in a single column.
- **Filter unwanted outliers** is essential to improve the performance of the data. If an outlier is found to be irrelevant it must be removed.
- **Handle missing data:** During analysis of data, it is important to understand why the missing values exist. Whether the missing value exist because they were not recorded or they imply that data does not exist for the missing values. Missing data may be handled by using the following ways:

  i) In case, the values were not recorded, it becomes essential to fill up the values by guessing based on the other values in that column and row. This is called ***imputation.*** E.g. if a value is missing for gender, it is understood that the value was not recorded, so we need to analyse data and give it a value, we cannot leave the value blank in this case. Therefore, you can input missing values based on other observations; but there is a chance of losing integrity of the data because these values are being filled based on assumptions and not actual observations.

  ii) If a value is missing because it doesn't exist e.g. the height of the oldest child of someone who doesn't have any children. In this case, it would make more sense to either leave it empty or to add a third value like *NA* or *NaN*. The NA

or NaN values should be replaced with 0. Panda's fillna() function to fill in missing values in a data frame can be used.

iii) In case, it is not possible to figure out the reason for the missing, then that particular value can be dropped. Pandas function, dropna() can be used to do this, but doing this can drop or lose information, so be mindful of what is being removed or dropped this before removing it.

- **Noisy Data:** Random variance in the data is called noise. The following methods called smoothening techniques are used to handle noisy data:

a) Binning Method: This method is also known as discretization, is used to smooth sorted data values by consulting the values around it i.e., the neighbouring values. It is a local smoothening method, since it refers to the neighbouring values. In this method, the entire data is divided into equal segments called bins or buckets. Smoothing by binning is done by one of the following methods:

- **Smoothening by Bin Means:** In smoothing by bin means, each value in a bin is replaced by the mean value of the bin.
- **Smoothening by Bin Median:** In smoothing by bin means, each value in a bin is replaced by the median value of the bin.
- **Smoothening by Bin Boundary:** In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries. Each bin value is then replaced by the closest boundary value.

Algorithm:

1. Sort the dataset.
2. Partition the dataset into ‘ n’ segments. Each segment should contain approximately same number of data elements. Partitioning can be done using either of the following methods namely: equal width binning, equal frequency binning or entropy-based binning.
3. Calculate the arithmetic mean or media or replace by boundary (min and max value)
4. Replace each data element in each bin by the calculated mean/ median/ boundaries.

Example: Given data:18,22,6,6,9,14,20,21,12,18,18,16. Illustrate binning by mean, median and boundary replacement. Given bin depth=3

Step1: Sort the data: 6,6,9,12,14,16,18,18,18,20,21,22,

Step 2: Partition the data into equal frequency bins of size of bin depth(n/d) where n= no. of elements and d= bin depth

N/D=12/3=4 bins

Bin 1: 6, 6 , 9

Bin 2: 12,14, 16
Bin 3: 18,18,18
Bin 4: 20, 21, 22

Step 3: Calculate Arithmetic mean

Arithmetic mean= Sum of observations ÷ number of observations

Bin 1= (6+6+9)/3=21/3=7

Bin 2= (12+14+16)/3= 42/3=14

Bin 3=(18+18+18)/3= 54/3=18

Bin 4= (20+21+22)/3= 63/3=21

Step 4: Replace each data element in each bin by the calculated mean

Bin 1: 7, 7, 7
Bin 2: 14,14,14
Bin 3: 18,18,18
Bin 4: 21, 21, 21

- **Binning using Median:** In this method, Step 1 and step 2 are same.

Step 3: Calculate Median (50% percentile)

Median is the observation 2 in each bin

Step 4: Replace each data element in each bin by the calculated mean

Bin 1: 6, 6, 6
Bin 2: 14,14,14
Bin 3: 18,18,18
Bin 4: 21, 21, 21

- **Binning using Boundary Values:** In this, we keep the minimum as well as maximum values.

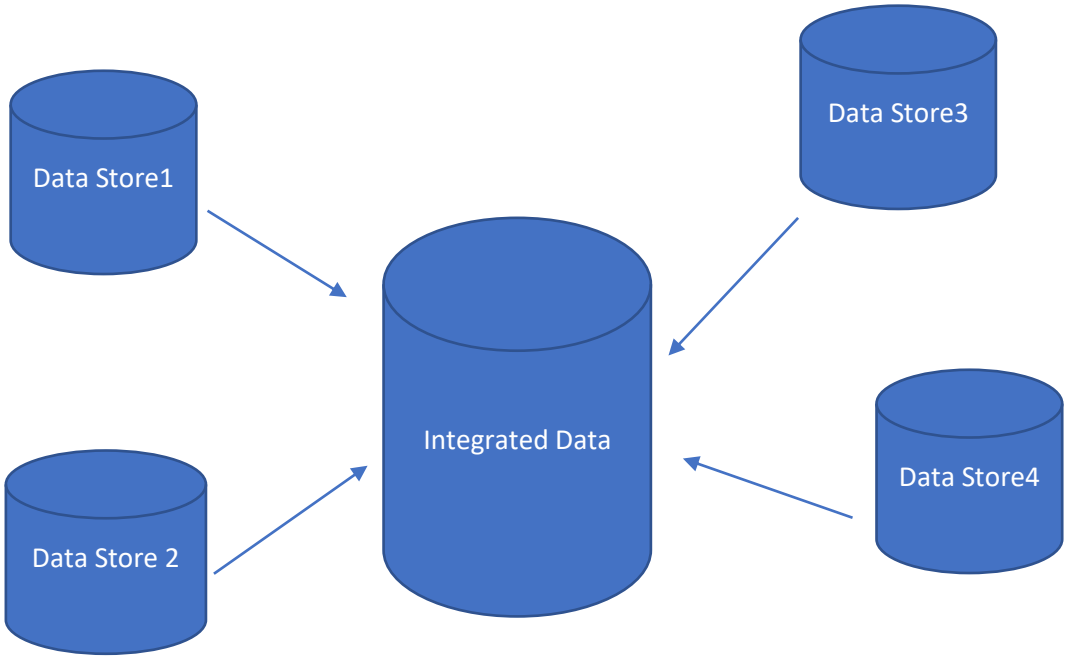
Bin 1: 6, 6, 9
Bin 2: 12,12,16
Bin 3: 18,18,18
Bin 4: 20, 20, 22

b) Regression: Regression is used to find a mathematical equation to fit the data to smooth out the noise. Regression may be linear or multiple. Linear regression is used to find the best line that fits two variables such that one predicts the other. Multiple linear regression involves more than two variables are involved.

c) Clustering: This approach is useful for organizing similar data in groups or clusters. The outliers may be detected by clustering. Data Integration makes data more comprehensive and more usable.

2.2.2 Data Integration

Data Integration is a vital step in which the data acquired from multiple sources is integrated into a single centralized location. Data Integration makes data comprehensive and more usable.



The most common approaches to integrate data are:

a) Data Consolidation
b) Data Propagation
c) Data Virtualization
d) Data Warehousing

a) Data consolidation means to consolidate data from several separate sources into one data store, so that it is available to all the stake holders to enable better decision making. It involves eliminating redundancies, removing inaccuracies before consolidating to a single store. The most common data consolidation techniques are ETL (Extract, Transform, Load), Data virtualization and Data warehousing.

- **ETL** is the most widely used data consolidation technique. The data is first extracted from multiple sources, then it is transformed into an understandable format by using various functions like sorting, aggregating, cleaning etc and then transfer it to a centralized store like another database or data ware house. The ETL process cleans, filters, and transforms data, and then applies business rules before data populates the new source. ETL is further of two types namely Real time ETL used in real time systems and Batch processing ETL used for high volume databases.
- **Data Virtualization:** In this method, the data stays in the original location, but changes are made in a virtual manner and can be seen in a consolidated manner by

the users. It is a logical layer that amalgamates data from various sources without performing actual ETL process. It is an abstraction such that only the required data is visible to the users without requiring technical details about the location or structure of the data source. It provides enhanced data security.

- **Data Warehousing** is the integration of data from multiple sources to a centralized source to facilitate decision making, reporting and query handling. A centralized source of data enables better decision making.

b) Data Propagation involves copying data from one location i.e., source to another location i.e., target location. It is event driven. These applications usually operate online and push data to the target location. They are majorly useful for real time data movement such as workload balancing, backup and recovery. Data propagation can be done asynchronously or synchronously.

The two methods for data propagation are: Enterprise Application Integration (EAI) and Enterprise Data Replication (EDR). The key advantage of data propagation is that it can be used for real-time / near-realtime data movement and can also be used for workload balancing, backup and recovery. EAI is used majorly for the exchange of messages and transactions in real-time business transaction processing; whereas for transfer of voluminous data between databases, is used.

c) Data Virtualization: In this, data is not stored in a single location, but is abstracted and can be viewed as unified view of data from multiple sources. Data Federation is a form of data virtualization, supported by Enterprise information technology (EII). EII products have evolved from two different technological backgrounds – relational DBMS and XML, but the current trend of the industry is to support both approaches, via SQL (ODBC and JDBC) and XML (XML Query Language - XQuery - and XML Path Language - XPath) data interfaces.

d) Data Warehousing is the integration of data from multiple sources to a centralized source to facilitate decision making, reporting and query handling. A centralized source of data enables better decision making.

2.2.3 Data Transformation

After the data has been acquired, it is cleaned as discussed above. The clean data may not be in a standard format. The process of changing the structure and format of data to make it more usable is called data transformation. Data transformation may be constructive, destructive, aesthetic or structural.

- *Constructive Data Transformation* involves adding, copying, and replicating data.
- *Destructive Data Transformation* involves deleting fields and records.
- *Aesthetic Data Transformation* involves standardizing salutations or street names.
- *Structural Data Transformation* involves renaming, moving, and combining columns in a database.

Data Transformation may require smoothing, aggregation, discretization, attribute Construction, generalization and normalization to make data manageable.

Scripting languages like Python or domain-specific languages like SQL are usually used for data transformation.

2.2.4 Data Reduction

It is the process of reducing the volume of data such that data integrity is preserved. i.e., the volume of data is reduced but the results of data mining before and after mining are the same. Data reduction increasing the efficiency of data mining. It helps in reducing the storage requirement, reduced computation time and removal of redundancy. The various techniques used for data mining are: *Dimensionality reduction, numerosity reduction and data compression.*

2.2.4.1 Dimensionality Reduction: is the transformation of data from high dimensional space to a low dimensional space. It is difficult to visualize data that has higher number of features. Sometimes, most of these features are correlated, and hence redundant. This is where the need of dimensionality reduction arises. In other words, dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. e.g., 3-dimensional data may be reduced to a 2D data. The various methods used for dimensionality reduction include:

- **Wavelet Transformation** is mostly used in image compression. It is a lossy method for dimensionality reduction, where a data vector X is transformed into another vector X', such that both X and X' represent the same length. The result of wavelet transform can be truncated, unlike its original, thus achieving dimensionality reduction. Wavelet transforms are well suited for data cube, sparse data or data which is highly skewed.

- **Principal Component Analysis (PCA**) is applied to skewed and sparse data. In this method, the entire data set is represented by few independent tuples with 'n' attributes. This method was introduced by Karl Pearson. It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum. It involves the following steps:

    i. Construct the covariance matrix of the data.
    ii. Compute the eigenvectors of this matrix.
    iii. Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of the original data.

  Hence, lesser number of eigenvectors are left.

- **Attribute Subset Selection:** In this method, a subset of some selected attributes is created for reducing the volume of data. The goal of attribute subset selection is to find a minimum set of attributes such that dropping of those irrelevant attributes does not much affect the utility of data and the cost of data analysis could be reduced.
  Attribute Subset Selection is done by the following methods:
  1. Stepwise Forward Selection.
  2. Stepwise Backward Elimination.
  3. Combination of Forward Selection and Backward Elimination.
  4. Decision Tree Induction.

2.2.4.2 Numerosity Reduction: is the reduction of original data and its representation in a smaller form. It can be done in two ways: parametric and non-parametric numerosity reduction.

i) **Parametric Numerosity Reduction:** In this method, only the data parameters are stored, instead of the entire original data. The data is represented using some model to estimate the data, so that only parameters of data are required to be stored, instead of actual data. Regression and Log-Linear methods are used for creating such models.

ii) **Non- Parametric Numerosity Reduction** methods are used for storing reduced representations of the data include *histograms*, *clustering*, *sampling* and *data cube aggregation.*

- **Histogram** is the data representation in terms of frequency. It uses binning to approximate data distribution and is a popular form of data reduction.
- **Clustering** divides the data into groups/clusters. This technique partitions the whole data into different clusters. In data reduction, the cluster representation of the data are used to replace the actual data. It also helps to detect outliers in data.
- **Sampling** can be used for data reduction because it allows a large data set to be represented by a much smaller random data sample (or subset).
- **Data Cube Aggregation** involves moving the data from detailed level to a fewer number of dimensions. The resulting data set is smaller in volume, without loss of information necessary for the analysis task.

2.2.4.3 Data Compression is a technique in which the original data is compressed for reduction. This compressed data can again be reconstructed to form the original data. If the data is reconstructed without losing any information, then it is a 'lossless' data reduction.

2.3 DATA MINING

Data mining is the process that helps in extracting information from a given data set to identify trends, patterns, and useful data. The objective of using data mining is to make data-supported decisions from enormous data sets. Different types of data can be mined such as Data stored in database, data warehouse, transactional data and other types of data

such as data streams, engineering design data, sequence data, graph data, spatial data, multimedia data, and more. In recent data mining projects, various major data mining techniques have been developed and used, including association, classification, clustering, prediction, sequential patterns, and regression.

- **Association** is a data mining technique useful to discover a link between two or more items. It uses if-then statements to show the probability of interactions between data items within large data sets. It finds a hidden pattern in the data set. It is commonly used to help sales correlations in data or medical data sets.
- **Clustering** is a data mining technique in which clusters are created of objects that share the same characteristics. Clusters relate to hidden patterns. It is based on unsupervised learning.
- **Classification** classifies items or variables in a data set into predefined groups or classes. It is based on unsupervised learning.
- **Prediction** is used in predicting the relationship that exists between independent and dependent variables as well as independent variables alone. It can be used to predict future trends. It analyses past events or instances in the right sequence to predict a future event.
- **Sequential patterns** is a data mining technique specialized for evaluating sequential data to discover sequential patterns. It comprises of finding interesting subsequence in a set of sequences, where the stake of a sequence can be measured in terms of different criteria like length, occurrence frequency, etc.

2.3.1 Data Mining Applications

Data mining is useful in predictions, classification, clustering and trends, which makes it applicable to many domains like health care, fraud detection, education, market basket analysis, manufacturing, sales, customer relationship management, finance and banking etc.

2.4 DATA INTERPRETATION

The process of reviewing data through some predefined processes to be able to assign some meaning to the data and arrive at a relevant conclusion is called data interpretation. It involves taking the result of data analysis, making inferences on the relations studied, and using them to reach conclusions and develop recommendations. Data interpretation is done by analyst to make inferences from the data. There are two ways to interpret data namely Qualitative and Quantitative.

Qualitative Data Interpretation: Qualitative data also called categorical data, does not contain numbers, it consists of text, pictures, observations, symbols etc. The interpretation of patterns and themes in qualitative data is done using qualitative methods.

Quantitative Data Interpretation is done on quantitative data i.e. numerical data. It involves statistical methods such as mean, standard deviation, variance, frequency distribution, regression etc. Data analysis and interpretation, regardless of method and qualitative/quantitative status, may include the following characteristics:

- Data identification and explanation

- Comparing and contrasting of data
- Identification of data outliers
- Future predictions

Data Interpretation is helpful in improving processes by identifying problems. Data interpretations is used for predicting trends after studying the patterns in the data. It is helpful in better decision making.

2.5 SUMMARY

This chapter introduces the various facets of data and the varius data preprocessing techniques. A brief introduction to dat mining and data interpretation is provided in this chapter.

2.6 PRACTICE QUESTIONS

Q1. What are the various types of data?
Q2.Explain the concept of data preprocessing. What are the techniques used for preprocessing of data?
Q3. Write a short note on Data mining.
Q4. What are the techniques used for data mining?
Q5. What is the importance of data Interpretation?

B.Sc.(DATA SCIENCE)

SEMESTER-I

INTRODUCTION TO DATA SCIENCE

UNIT III: REPRESENTATION OF DATA

STRUCTURE

3.0 OBJECTIVES

1. To familiarize with the data representation and types of data
2. To familiarize with the general techniques of handling data

3.1 BIG DATA ANALYTICS

The assortment of any kind of data which is large, highly complex and difficult to manage and handle in the real time scenario with the traditional management techniques becomes the Big Data. The most extensively technique for the management of data was Relational DBMS which was a kind of dataset that fits in all the scenarios for the storage, manipulation and interpretation of data but it failed in case of big data. A groundbreaking study in 2013 reported 90% of the entirety of the world's data has been created within the previous two years. In the previous two years researchers have composed and handled data which is 9 times the data being processed in the previous 1000 years of the survival. As per the statistical institute, already 2.7 zettabytes of data have been generated and by 2025 it might escalate to a limit beyond belief which ca be 100zettabytes or even more.

What do we do with all of this data? How do we make it useful to us? What are its real-world applications? These questions are the domain of data science.

3.2 DATA SCIENCE - WORKING

In order to handle the query for complete in depth and a sophisticated approach for the exploration of the raw unprocessed data into the real time multiple disciplines with the expertise in machine learning and AI based dimensions, data science gives the solution with the most advanced means and methods. The scrutiny of the relevant information from the irrelevant raw data and pass on or forward only the most vital data for the enhancing the computing efficiency with the revolution in the fields of engineering, mathematics, statistics, advanced computing and visualizations. The data science is the basic field of exploration of data where the researchers or the data scientists also rely heavily on artificial intelligence for the creation of simulated models and then predicting the values of the parameters with the algorithms designed for manipulation of data into information. The working in the field of the data science is thereby based on the types of data being explored and manipulation needed e.g., if the simple data in form of tables is available then RDBMS is used and if image data is present then RDBMS fails.

3.3 FACETS OF DATA

In data science and big data, the data required for the manipulation and interpretation changes with the change in the types of tools and techniques applied in the algorithm. The types of the data explored in the field of data science can be categorized as follows:

- Structured
- Unstructured
- Natural language
- Machine-generated
- Streaming

- Acoustic data, video, and images
- Sensor Data
- Graph-based/ Network data

As in the previous chapter we have explored the data categories structured versus unstructured data. Let's talk about the rest of the representation of data and the explore the characteristics. The structured data is the main type of data explored in a model and can be expressed as a record with a field of fixed length and dimension. The RDBMS and the Excel data is usually the structured data with known or used defined datatypes with structural information. Bu the unstructured data is data that may not fit any kind of mathematical or simulated model for the study of data due to the context-specific or varying nature of the stored raw data. One example of unstructured data is the email.

3.3.1 Natural Language Data

Another special type of unstructured data is the Natural language data with the challenging approach to progression the data and manipulate it. The handling of large amount of natural language data and processing in itself necessitates the data scientists to acquire the knowledge of specific data science techniques and linguistics. The natural language

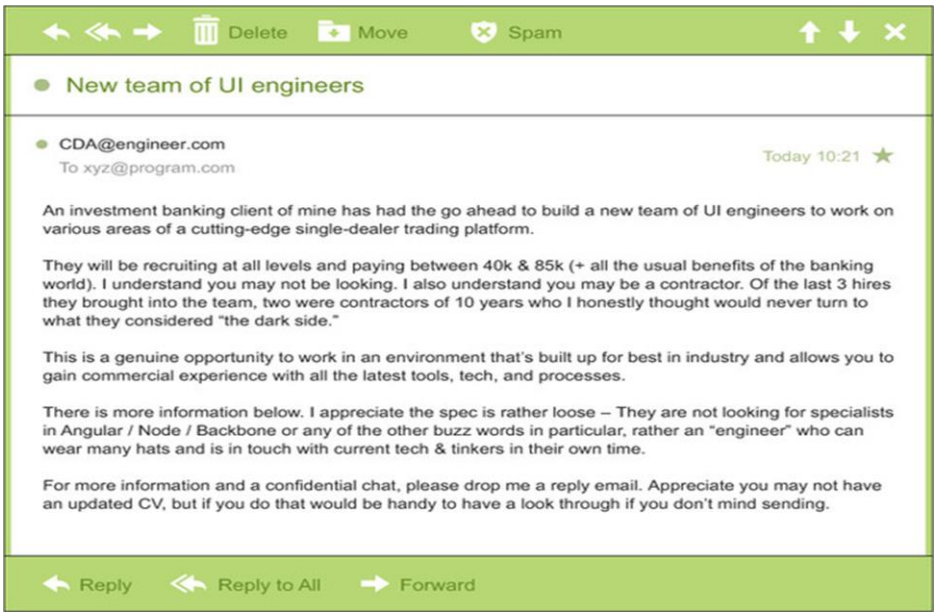


Fig. 3.1.1 Perfect example of Natural Language data [12]

processing community has had success in entity recognition, topic recognition, summarization, text completion, and sentiment analysis, but models trained in one domain don't generalize well to other domains. Even state-of-the-art techniques aren't able to decipher the meaning of every piece of text. The meaning of the same words can vary when coming from someone upset or joyous. This is illustrated in figure 3.1.

3.3.2 Machine Generated Data

The machine or the computer has the capability to generate the data based on the requirement of the developer or the application developed. The data generated by the machine or any robotic process without the intervention of a human is a fast process and also help to forecast certain information for the application, or other machine without human intervention. Machine-generated data is becoming a major data resource and will continue to do so. Wikibon has forecast that the market value of the industrial Internet (a term coined by Frost & Sullivan to refer to the integration of complex physical machinery with networked sensors and software) will be approximately $540 billion in 2020. IDC (International Data Corporation) has estimated there will be 26 times more connected things than people in 2020. This network is commonly referred to as the internet of things.The analysis of machine data relies on highly scalable tools, due to its high volume and speed. Examples of machine data are web server logs, call detail records, network event logs, and telemetry. This is illustrated by figure 3.2.

```
CSIPERF:TXCOMMIT;313236
2014-11-28 11:36:13, Info                  CSI    00000153 Creating NT transaction (seq
69), objectname [6]"(null)"
2014-11-28 11:36:13, Info                  CSI    00000154 Created NT transaction (seq 69)
result 0x00000000, handle @0x4e54
2014-11-28 11:36:13, Info                  CSI    00000155@2014/11/28:10:36:13.471
Beginning NT transaction commit...
2014-11-28 11:36:13, Info                  CSI    00000156@2014/11/28:10:36:13.705 CSI perf
trace:
CSIPERF:TXCOMMIT;273983
2014-11-28 11:36:13, Info                  CSI    00000157 Creating NT transaction (seq
70), objectname [6]"(null)"
2014-11-28 11:36:13, Info                  CSI    00000158 Created NT transaction (seq 70)
result 0x00000000, handle @0x4e5c
2014-11-28 11:36:13, Info                  CSI    00000159@2014/11/28:10:36:13.764
Beginning NT transaction commit...
2014-11-28 11:36:14, Info                  CSI    0000015a@2014/11/28:10:36:14.094 CSI perf
trace:
CSIPERF:TXCOMMIT;386259
2014-11-28 11:36:14, Info                  CSI    0000015b Creating NT transaction (seq
71), objectname [6]"(null)"
2014-11-28 11:36:14, Info                  CSI    0000015c Created NT transaction (seq 71)
result 0x00000000, handle @0x4e5c
2014-11-28 11:36:14, Info                  CSI    0000015d@2014/11/28:10:36:14.106
Beginning NT transaction commit...
2014-11-28 11:36:14, Info                  CSI    0000015e@2014/11/28:10:36:14.428 CSI perf
trace:
CSIPERF:TXCOMMIT;375581
```

Fig. 3.1.2 Example of machine generated data [12]

3.3.3 Streaming Data

The streaming data when being administered for the information can take nay form and format which is an interesting property. This type of data only gets loaded into the server or the data warehouse when any relevant activity occurs or there's an interpretable change in the parameters. The data is not accessed in batch mode. This type of data is not actually a different category but the system for the processing of such varying formats needs to be adaptive to the minutest variations in the information to be handled.

Examples are the "What's trending" on Twitter, live sporting or music events, and the stock market.

3.3.4 Acoustic, video and image data

The world of animation and digitization has developed multimedia datasets with audio, video and images. These types of datsets can be stored, handled and interpreted with the Object-Oriented Databases. The databases include the class inheritance and interface in a pre-specified format for handling the complexity of the data and finding its application in Digital libraries, video-on demand, news-on demand, musical database, etc. The other type of

information in the multimedia datasets can be represented and reproduced as sensory experiences - touch, sense and hear which are typically leading to storing a huge amount of data handled by digitizing. Furthermore, data compression for images and sounds can exploit limits on human senses performed by throwing away information not needed for good-quality experience which is performed by compression. There are certain limitations when you deal with such a large amount of data are described below:

1. **Range is limited and might lead to misinformation**
    - only certain pitches and loudness can be heard
    - only certain kinds of light are visible, and there must be enough / not too much light
2. **Discrimination due to the descriptive features of the data**
    - pitches, loudness, colors, intensities can't be distinguished unless they are different enough (color1, color2)
3. **Coding the information of the sensory data into digital world.**
    - nervous systems "encode" experience, e.g., rods and cones in the eye

3.3 REPRESENTATION OF IMAGE DATA

The image data is expanding vastly and due to enhancement in the cameras the encoding is needed for computation and manipulation of image data. The techniques are vector array-based encoding and bit map-based encoding.

Vector graphics encode the image sequences as a series of lines or curves. The process is expensive in term of image computation but smoothly rescales.

Bit map encode the image as an array of pixels. The encoding process is cost effective in terms of computation but scales inefficiently leading to loss of image data.

The Basic idea of image data is the array of receptors where each receptor records a pixel by "counting" the number of photons that strike it during exposure. The Red, green, blue recorded separately at each point on image produced by group of three receptors where each receptor is behind a color filter.

Representation of Acoustic data: In recent years, the analysis of acoustic characteristics of speech and sound has been one of the areas that data mining has found its way through. The present research study is also related to this topic which aims to detect the gender of the speaker by using the acoustic feature of his voice. When an instrument is played or a voice speaks, periodic (many times per second) changes occur in air pressure, which we interpret as acoustics. For the representation of the acoustic data compression is needed. Codecs (compression/decompression) implement various compression/decompression techniques which are either lossy or lossless. The lossy compression of the acoustic data may lead to loss of certain information as it is non-repetitive: MPEG (like JPEG) a family of perceptually-based techniques are all lossy techniques. While the other type of techniques is where the information of the acoustic data is preserved in which WMA Lossless, ALAC, MPEG-4 ALS methods are applied.

The encoding of data in form of the sounds heard or the visuals captured are highly challenging for the data scientists to process the information. Some tasks performed by the human brain have been of great difficulty to the computers for the recognition of the object in the images. MLBAM (Major League Baseball Advanced Media) announced in 2014 that they'll increase video capture to approximately 7 TB per game for the purpose of live, in-game analytics. The higher resolution cameras and acquiring sensors have the capability to capture the motion of ball and the player in a real time scenario e.g., the path taken by a defender relative to two baselines.

Recently a company called DeepMind succeeded at creating an algorithm that's capable of learning how to play video games. This algorithm takes the video screen as input and learns to interpret everything via a complex process of deep learning. It's a remarkable feat that prompted Google to buy the company for their own Artificial Intelligence (AI) development plans. The learning algorithm takes in data as it's produced by the computer game; it's streaming data.

3.3.5 Sensor Data

Any input acquired from the physical environment which is detected and responded from the devices as an output is the collaborative approach of the sensor data. The extracted data from the sensor devices act as an output for a real time system or as an input to another system for the performance of any activity. These sensors can be used to perceive any type of physical element with the approach to detect the events or changes in the environment. A sensor is always used with other electronics, as simple as a lamp or as complex as a computer. Advanced chip technology makes it possible to integrate all the required functions at low cost, in a small volume and with low energy consumption. The number of sensors around us is increasing rapidly. Estimates vary, but many expect that by 2030 more than 500 billion sensors will be connected to each other via the Internet of Things (IoT).

The exponential growth of the IoT based systems leads to ever demanding rise in the input sensor devices which are responsible for the collection, storage and interpretation of the captured data. In addition, consumers, organizations, governments and companies themselves produce more and more data, for example on social media. The amount of data is growing exponentially. People speak of Big Data when they work with one or more datasets that are too large to be maintained with regular database management systems.

The pros of applying sensor data to the input devices is that the decisions of the IoT devices are subjected to information accessed from the evidences and not from any kind of irrelevant details and subjective experiences. This type of knowledge base makes the system cost effective with the enhanced streamlined processes, boosted product quality and better services. By combining data intelligently and by interpreting / translating, new insights are created that can be used for new services, applications and markets. This information can also be combined with data from various external sources, such as weather data or demographics.

3.3.6 Graph/ Network Data

"Graph data" is the type of data which can be represented as graph with a special

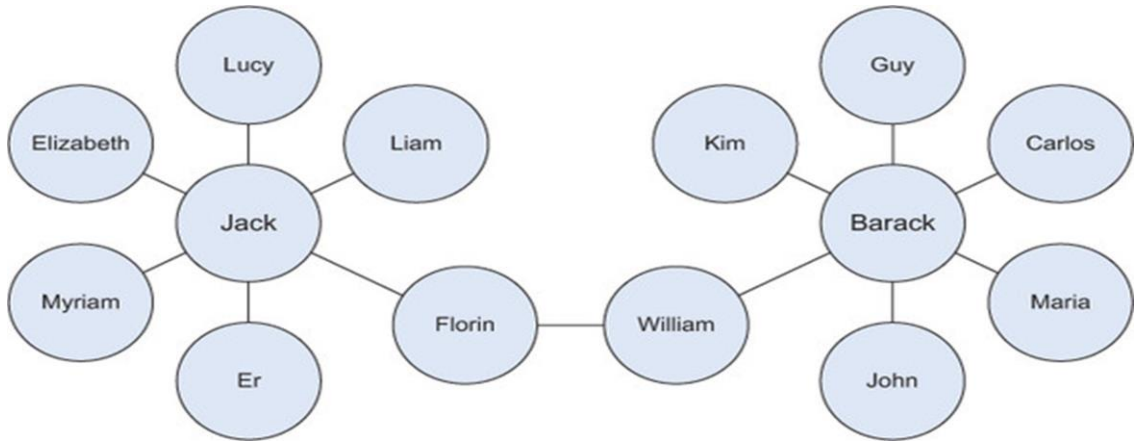


Fig. 3.1.3 Friends in social network are example of Graph Network[12]

characteristic of comprising the mathematical graph theory into the mined information. "Graph" in the network data generally represents a model in the statistical and mathematical domain with pair wise relationship in the constituent objects. Graph or network data is, in short, data that focuses on the relationship or adjacency of objects. The representation of the graph models includes the nodes, edges, and relationships between the stored data in the nodes. Graph-based data is a natural way to represent social networks, and its structure allows you to calculate specific metrics such as the influence of a person and the shortest path between two people.

Examples of graph-based data can be found on many social media websites (figure 3.3). For instance, on LinkedIn you can see who you know at which company. Your follower list on Twitter is another example of graph-based data. Graph databases are used to store graph-based data and are queried with specialized query languages such as SPARQ.

3.4 MULTIPLE PROBLEMS IN HANDLING LARGE DATASETS

The multiple problems in the large datasets are discussed by numerous data scientists with the need to understand the growing demand of the data and handling the mathematical as well as statistical operations for the manipulation of the data. In the last two years, over 90% of the world's data was created, and with 2.5 quintillion bytes of data generated daily, it is clear that the future is filled with more data, which can also mean more data problem in context to the following:

- Collecting, storing, sharing and securing data
- Creating and utilizing meaningful insights from their data.

Some common big data problems and the respective solutions have been discussed below.

1. Lack of Understanding

The lack of understanding of the mined data in the data science-based companies might lead to knocked down the performance in many areas. Many of the major areas for the data-

based companies were: depreciate the expenses of mining information, innovate new ideas for interpretation, new product launching, enhance performance and so on. Despite the benefits, companies have been slow to adopt data for a data centric approach.

Solution: Follow a top-down approach for the introduction and manipulation of the data science based on the procedures followed up. In case of lack of a data science professional, the consultancy services or an IT proficient with data science knowledge should be hired to get a better understanding.

2. **High Cost of Data Solutions**

The companies have understood that buying and maintaining of necessary components make the system less cost effective. In addition to cost of the servers and the software-based storage, the high-end cost of the data science experts makes the system time consuming.

Solution: The solution is to understand the need and use of the data with a collaborative method to find a goal, conduct a research or solution and implement the execution with a plan.

3. **Too Many Choices**

Coined as the "paradox of choice," Schwartz explains how option overload can cause inaction on behalf of a buyer. In the world of data and data tools, the options are almost as widespread as the data itself, so it is understandably overwhelming when deciding the solution that's right for the business, especially when it will likely affect all departments and hopefully be a long-term strategy.

Solution: Like understanding data, a good solution is to leverage the experience of your in-house expert, perhaps a CTO. If that's not an option, hire a consultancy firm to assist in the decision-making process. Use the internet and forums to source valuable information and ask questions.

4. **Complex Systems for Managing Data**

The systems to understand the data management and finding a relevant solution for the manipulation of data is in itself a problem Due to the vast expanse of the different types of data with the IT teams creating their own data during the process of data handling results in increased complexity.

Solution: Find a solution with a single command center, implement automation whenever possible, and ensure that it can be remotely accessed 24/7.

5. **Security Gaps**

Another important aspect of the data science is the security of the data and the biasing of the large amount of data is always possible. In order to handle it the encryption and decryption must be performed with the data store with proper storage.

Solution: The data need to be handled with automated security updates of the data warehouse and automated backups.

6. Low Quality and Inaccurate Data

Having data is only useful when it's accurate. Low quality data not only serves no purpose, but it also uses unnecessary storage and can harm the ability to gather insights from clean data.

A few ways that data can be considered low quality is:

- Inconsistent formatting (which will take time to correct and can happen when the same elements are spelled differently like "US" versus "U.S."),
- Missing data (i.e. a first name or email address is missing from a database of contacts),
- Inaccurate data (i.e. it's just not the right information or the data has not be updated).
- Duplicate data (i.e. the data is being double counted)
- If data is not maintained or recorded properly, it's just like not having the data in the first place.

Solution: Begin by defining the necessary data you want to collect (again, align the information needed to the business goal). Cleanse data regularly and when it is collected from different sources, organize and normalize it before uploading it into any tool for analysis. Once you have your data uniform and cleansed, you can segment it for better analysis.

7. Compliance Hurdles

When collecting information, security and government regulations come into play. With the somewhat recent introduction of the General Data Protection Regulation (GDPR), it's even more important to understand the necessary requirements for data collection and protection, as well as the implications of failing to adhere. Companies have to be compliant and careful in how they use data to segment customers for example deciding which customer to prioritize or focus on. This means that the data must: be a representative sample of consumers, algorithms must prioritize fairness, there is an understanding of inherent bias in data, and Big Data outcomes have to be checked against traditionally applied statistical practices.

Solution: The only solution to adhere to compliance and regulation is to be informed and well-educated on the topic. There's no way around it other than learning because in this case, ignorance is most certainly not bliss as it carries both financial and reputational risk to your business. If you are unsure of any regulations or compliance you should consult expert legal and accounting firms specializing in those rules.

3.5 General Techniques for Handling Large Data

The multiple challenges have been discussed in the section above and the solutions for the defies are categorized based on the algorithms and out of memory errors. Never-ending algorithms, out-of-memory errors, and speed issues are the most common challenges you face when working with large data. In this section, we'll investigate solutions to overcome or alleviate these problems.

The solutions can be divided into three categories: using the correct algorithms, choosing the right data structure, and using the right tools. (Figure 3.4)

There is no such relationship between the problems discussed in the section above and

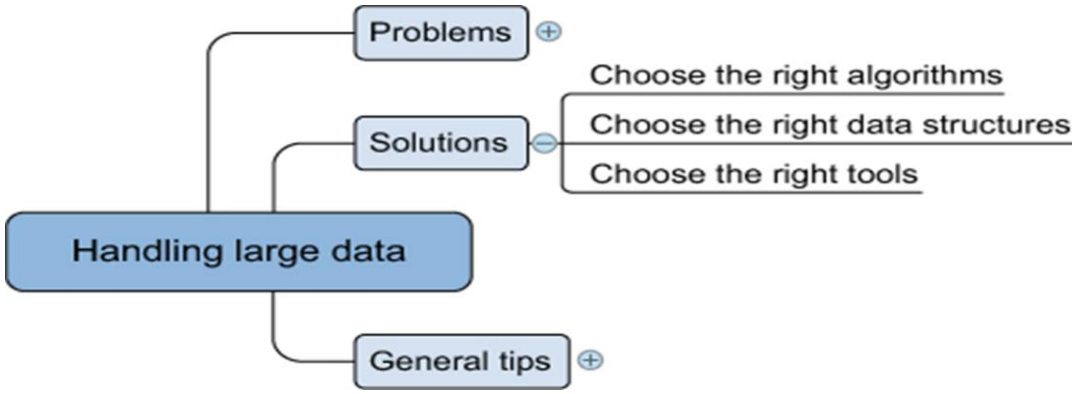


Fig. 3.1.4 Overview of solutions for handling large data sets[12]

the solutions to be incorporated. There are multiple solutions to a given problem which can also handle the issue of memory and computational overhead. This type of data science solutions can be generalized for challenges in the exploration of data. For instance, the compression and decompression of the data set help resolve the memory issues but this also affects computation speed with a shift from the slow hard disk to the fast CPU. Contrary to RAM (random access memory), the hard disc will store everything even after the power goes down, but writing to disc costs more time than changing information in the fleeting RAM. When constantly changing the information, RAM is thus preferable over the (more durable) hard disc.

3.5.1 Choosing the Right Algorithm

T

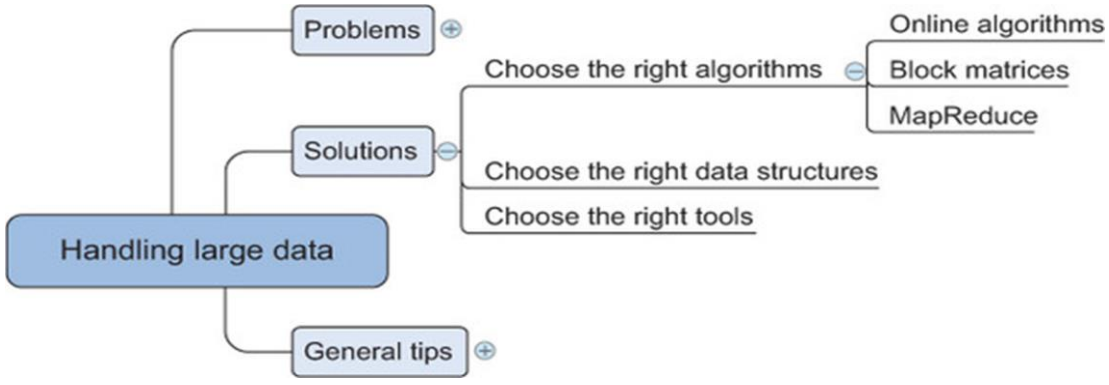


Fig. 3.1.5 The algorithms for handling the Big Data[12]

The selection of an effective algorithm for the processing of the data can provide better results as compared to the enhanced hardware. In order to perform the predictive or

selective analysis the best-chosen algorithm need not necessarily load the complete data into the memory or RAM, it rather supports the parallel computations with parallel or distributed databases. In this section three types of algorithms have been discussed that can perform the computations parallelly reducing the computation or memory overhead: online algorithms, block algorithms, and MapReduce algorithms, as shown in figure 3.5.

Several, but not all, machine learning algorithms can be trained using one observation at a time instead of taking all the data into memory. The model can be trained based on the current parameters and the previous parameter values can be made to forget by the algorithm. This technique of "use and forget " helps to attain high memory effectiveness in the system. This way as the new data values is acquired by the algorithm, the previous values are forgotten or overwritten but the effect can be witnessed or observed in the performance metrics of the proposed model. Most online algorithms can also handle mini-batches; this way, the data science exert or the developer can feed the batches of 10 to 1,000 observations at one single instance and then applying the sliding window protocol to access the data. This learning can be handled by multiple means discussed as follows:

- Full batch learning (also called statistical learning) —Feed the algorithm all the data at once.
- Mini-batch learning —Feed the algorithm a spoonful (100, 1000, ..., depending on what your hardware can handle) of observations at a time.
- Online learning —Feed the algorithm one observation at a time.

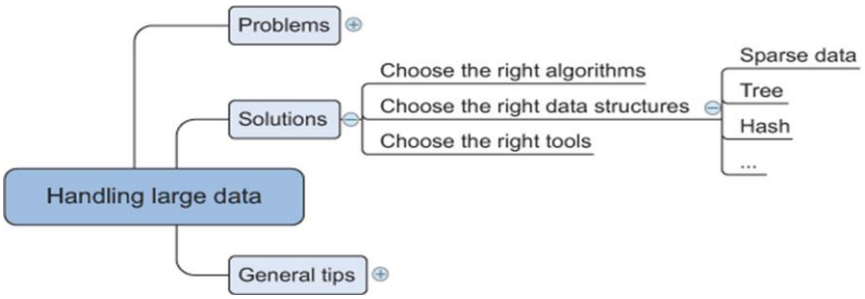


Fig. 3.1.6 Data structures applied in the data science[12]

3.5.2 Right Data Structure

The algorithms as discussed can enhance the performance and execution for the manipulation of the data in the warehouse. This process in the data science field actually leads to fragmentation in the raw data so that is the reason the structures for the storage of the data is equally important for the data scientist or a data science researcher. Data structures have different storage requirements, but also influence the performance of CRUD (create, read, update, and delete) and other operations on the data set.

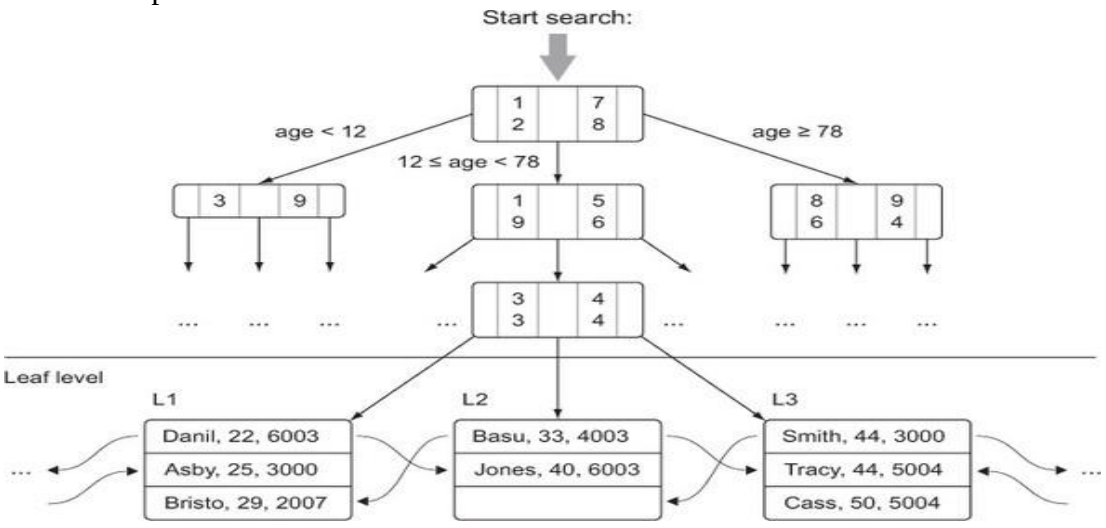
- **Sparse Data**

Most of the IT professional and the data scientist understand the representation through a sparse matrix. Similarly, the sparse data illustration can be done by giving a minimal detail about the data as compared to the total number of entries. As per figure 3.7, the conversion of the data from the text to binary or an image to binary can be expressed as the sparse data. Imagine a set of 100,000 completely unrelated Twitter tweets. Most of them probably have fewer than 30 words, but together they might have hundreds or thousands of distinct words. For this reason, the text documents are processed for the stop words, cut into fragments and stored as vectors instead of the binary information. The basic idea is that any word present in the tweet is expressed as 1 and not in tweet is expressed as 0 resulting in sparse data indeed. But the matrix generated would require equal memory as compared to any other matrix even though it has a little information.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **Tree Structure**

Trees is a special type of data structure that has faster retrieval of information in comparison to the table or sparse data. In these data structures the root node is the first directory for accessing the information and the child nodes are the sub directories of the root node. The information can be accessed form the child or leaf nodes by either using pointers or indexing of the tree structure. The figure 3.8 helps the researcher to understand the process of information retrieval in the tree structure.



- **Hash Tables**

The process of the calculation of the key for each data entry and allotting the key to the relevant bucket is the important process of the hash table structure for th storage of data. The process of storage and handling in the hash table makes it more reliable source for the retrieval of information based on the key value. This way you can quickly retrieve the information by looking in the right bucket when you encounter the data. Dictionaries in Python are a hash table implementation, and they're a close relative of key-value stores. You'll encounter them in the last example of this chapter when you build a recommender system within a database. Hash tables are used extensively in databases as indices for fast information retrieval.

3.5.2 Choosing the Right Tools

As discussed earlier in section 3.4.1 and 3.4.2 with the need to have a right algorithm

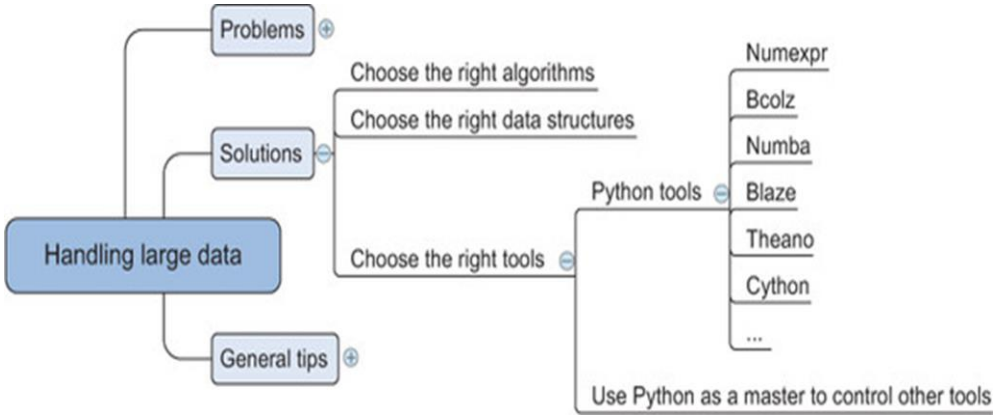


Fig. 3.1.7 Tools applied by data scientist for handing data[12]

and the right data structure, the requirement of a good tool is also important. The right tool can be a Python library or at least a tool that's controlled from Python, as shown figure 3.9.

Python has a number of libraries that can help you deal with large data. They range from smarter data structures over code optimizers to just-in-time compilers. The following is a list of libraries we like to use when confronted with large data:

Cython —The closer to the actual hardware of a computer, the more vital it is for the computer to know what types of data it has to process. For a computer, adding 1 + 1 is different from adding 1.00 + 1.00. The first example consists of integers and the second consists of floats, and these calculations are performed by different parts of the CPU. Python needs no specifications of the types of the data but the compiler can itself interpret the values based on the integer or float. This is although a slow process and so a superset of python can be used for the solution which forces the user to define the data type before the execution and hence can be implemented much faster. See http://cython.org/ for more information on Cython.

Numexpr —Numexpr is at the core of many of the big data packages, as is NumPy for in-memory packages. Numexpr is a numerical expression evaluator for NumPy but can be many times faster than the original NumPy. https://github.com/pydata/numexpr.

Numba —Numba is the package which compiles (just-in-time compiling) the code for the data manipulation and handling before actually executing it which makes it faster and less prone to errors. With the user gets a platform to develop a high level code wit the speed of the compilation as in C. http://numba.pydata.org/.

Bcolz —Bcolz helps you overcome the out-of-memory problem that can occur when using NumPy. It can store and work with arrays in an optimal compressed form. It not only slims down your data need but also uses Numexpr in the background to reduce the calculations needed when performing calculations with bcolz arrays. http://bcolz.blosc.org/.

Blaze — Blaze is ideal in case the data scientist use the power of a database backend but like the "Pythonic way" of working with data. Blaze will translate your Python code into SQL but can handle many more data stores than relational databases such as CSV, Spark, and others. Blaze delivers a unified way of working with many databases and data libraries. http://blaze.readthedocs.org/en/latest/index.html.

Theano —Theano enables you to work directly with the graphical processing unit (GPU) and do symbolical simplifications whenever possible, and it comes with an excellent just-in-time compiler. On top of that it's a great library for dealing with an advanced but useful mathematical concept: tensors. http://deeplearning.net/software/theano.

3.6 DISTRIBUTING DATA STORAGE AND PROCESSING WITH FRAMEWORKS

New big data technologies such as Hadoop and Spark make it much easier to work with and control a cluster of computers. Hadoop can scale up to thousands of computers, creating a cluster with petabytes of storage. This enables businesses to grasp the value of the massive amount of data available. "Big data Analytics" is a phrase that was coined to refer to amounts of datasets that are so large, traditional data processing software simply can't manage them. For example, big data is used to pick out trends in economics, and those trends and patterns are used to predict what will happen in the future. These vast amounts of data require more robust computer software for processing, best handled by data processing frameworks. These are the top preferred data processing frameworks, suitable for meeting a variety of different needs of businesses.

3.6.1 Hadoop

This is an open-source batch processing framework that can be used for the distributed storage and processing of big data sets. Hadoop relies on computer clusters and modules that have been designed with the assumption that hardware will inevitably fail, and those failures should be automatically handled by the framework.

There are four main modules within Hadoop. Hadoop Common is where the libraries and utilities needed by other Hadoop modules reside. The Hadoop Distributed File System (HDFS) is the distributed file system that stores the data. Hadoop YARN (Yet Another Resource Negotiator) is the resource management platform that manages the

computing resources in clusters, and handles the scheduling of users' applications. The Hadoop MapReduce involves the implementation of the MapReduce programming model for large-scale data processing.

Hadoop operates by splitting files into large blocks of data and then distributing those datasets across the nodes in a cluster. It then transfers code into the nodes, for processing data in parallel. The idea of data locality, meaning that tasks are performed on the node that stores the data, allows the datasets to be processed more efficiently and more quickly. Hadoop can be used within a traditional onsite datacenter, as well as through the cloud.

3.6.2 Apache Spark

Apache Spark is a batch processing framework that has the capability of stream processing, as well, making it a hybrid framework. Spark is most notably easy to use, and it's easy to write applications in Java, Scala, Python, and R. This open-source cluster-computing framework is ideal for machine-learning, but does require a cluster manager and a distributed storage system. Spark can be run on a single machine, with one executor for every CPU core. It can be used as a standalone framework, and you can also use it in conjunction with Hadoop or Apache Mesos, making it suitable for just about any business.

Spark relies on a data structure known as the Resilient Distributed Dataset (RDD). This is a read-only multiset of data items that is distributed over the entire cluster of machines. RDDs operate as the working set for distributed programs, offering a restricted form of distributed shared memory. Spark is capable of accessing data sources like HDFS, Cassandra, HBase, and S3, for distributed storage. It also supports a pseudo-distributed local mode that can be used for development or testing.

The foundation of Spark is Spark Core, which relies on the RDD-oriented functional style of programming to dispatch tasks, schedule, and handle basic I/O functionalities. Two restricted forms of shared variables are used: broadcast variables, which reference read-only data that has to be available for all the nodes, and accumulators, which can be used to program reductions. Other elements included in Spark Core are:

Spark SQL, which provides domain-specific language used to manipulate Data Frames.

Spark Streaming, which uses data in mini-batches for RDD transformations, allowing the same set of application code that is created for batch analytics to also be used for streaming analytics. Spark MLlib, a machine-learning library that makes the large-scale machine learning pipelines simpler. GraphX, which is the distributed graph processing framework at the top of Apache Spark.

3.6.3 Apache Storm

This is another open-source framework, but one that provides distributed, real-time stream processing. Storm is mostly written in Clojure, and can be used with any programming language. The application is designed as a topology, with the shape of a Directed Acyclic Graph (DAG). Spouts and bolts act as the vertices of the graph. The idea behind Storm is to define small, discrete operations, and then compose those operations into a topology, which acts as a pipeline to transform data.

3.6.4 Samza

Samza is another open-source framework that offers near a real-time, asynchronous framework for distributed stream processing. More specifically, Samza handles immutable streams, meaning transformations create new streams that will be consumed by other components without any effect on the initial stream. This framework works in conjunction with other frameworks, using Apache Kafka for messaging and Hadoop YARN for fault tolerance, security, and management of resources.

3.6.5 Flink

Flink is a hybrid framework, open-source, and stream processes, but can also manage batch tasks. It uses a high-throughput, low-latency streaming engine that is written in Java and Scala, and the runtime system that is pipelined allows for the execution of both batch and stream processing programs. The runtime also supports the execution of iterative algorithms natively. Flink's applications are all fault-tolerant and can support exactly-once semantics. Programs can be written in Java, Scala, Python, and SQL, and Flink offers support for event-time processing and state management.

3.7 SUMMARY

Data science involves using methods to analyze massive amounts of data and extract the knowledge it contains. You can think of the relationship between big data and data science as being like the relationship between crude oil and an oil refinery. Data science and big data evolved from statistics and traditional data management but are now considered to be distinct disciplines.

The foremost aspect for the data scientist to conduct the refining of the raw data is the representation of data dealing with the problems of big data. Another aspect of the dealing with the problems of big data is to perform the processing of the big data with the frameworks. For this multiple special type of data have been explored.

There are multiple problems being faced by the data scientists for the processing of the raw data and mining it into useful and relevant information. Luckily, there are pragmatic solutions that companies can take to overcome their data problems and thrive in the data-driven economy. The problems and the relevant solutions have been discussed in the chapter. Data processing frameworks are not intended to be one-size-fits-all solutions for businesses. Hadoop was originally designed for massive scalability, while Spark is better with machine learning and stream processing. A good IT services consultant can evaluate your needs and offer advice. What works for one business may not work for another, and to get the best possible results, you may find that it's a good idea to use different frameworks for different parts of your data processing.

3.8 PRACTICE QUESTIONS

Q1. Discuss the various representation techniques of different data types.
Q2.What are the issues that are faced in handling large data?
Q3. Explain briefly the techniques to handle large data.
Q4.What are the popular frameworks for big data storage?

B.Sc.(DATA SCIENCE)

SEMESTER-I

INTRODUCTION TO DATA SCIENCE

UNIT IV: DATA ACQUISITION AND PROCESSING

STRUCTURE

4.1 OBJECTIVE

1. To familiarize with the Data Science Ethics
2. To familiarize with different aspects of privacy
3. To familiarize with the future trends of data Science

4.2 DATA SCIENCE ETHICS

The skill to extract or mine the relevant patterns and the transforming capability to revolutionize the products in data science helps to bring a positive change in the social and technological sphere making it ethically neutral. It does not come with its own perspective of either: what is correct or incorrect; nor: what is good or bad in using it. There is no such kind of a value-based framework while the companies working on the data store have a value-based system for the handling of the information. Anything which is private or protected is not for anyone to access except the administrator or the data scientist himself/herself. The problems of the ethical mishandling of data and the relevant solutions to the problems must be able to amalgamate with the ethics of the companies working in Bigdata.

The future of the technology world is in the hands of machine learning techniques with AI based systems where data science is the solution for the data on which these systems are trained. The data science is kind of fuel for he working intelligent systems as there training can be done on millions of raw data and all of it can be mined or extracted from the data science sources through the extractive approach of the data scientists. Data Ethics is a rapidly improvising field-of-study. The IT professionals and the data scientist work for the collection, sharing and the manipulation of data which is done by keeping ethics in the exploration of data and are sometimes even forced to deal with the ethics to avoid any kind of negative public opinion. Loss of ethics sometimes can prove to be alienating to the reputation and work culture of any kind of organization.

4.3 DATA SCIENCE – GOOD ASPECTS FOR TECHNOLOGY

Data science involves a plethora of disciplines and expertise areas to produce a holistic, thorough and refined look into raw data. The professionals might be trained for the handling, manipulation and the interpretation of data but the most important and vital part of the data science is to perform the scrutiny of the relevant information from the disarrayed or unorganized form of raw data and only interconnect or forward only those data values which are of vital importance for the invention or the improvisation of the existing system. These reasons are sufficient enough for the data scientists to completely depend upon the existing technological advancement in the field of machine learning or deep learning with the ability to create the simulated models and predict the values in the models with the proposed algorithms and techniques.

To acclimatize the organization with the ethical data science, there's a need to understand what actually are the ethics about data science with the cost requirement for the implementation of ethical values and what can be the ways to execute such practices.

There are several ways and means with the respective solutions for the same.

Firstly, the data scientist needs to understand about the daintiness of the information in the data store. With this the relevant implementation or the operations can be performed on the data keeping in the view the consequences with improper or unacceptable access of the data for information. Indeed, the security perspective in computer or network has proved many times that ignoring the consequences of accessing the data unethically might cause a trouble to the data professionals or the company as a whole in terms of loss of reputation or money.

With the available systems it is difficult for any data scientist to predict the inevitable unethical uses of the data but with the AI based techniques especially using machine learning and deep learning, a prediction can be made for the unintended consequences. For example, Facebook's "Year in Review" that reminded people of deaths and other painful events. This can be handled by the data professionals in the field of data science by keeping in view the patterns of the data to be explored and how to think of better means to represent the data with a new or enhanced approach.

Another important step to stop the data mining if the administrator finds any kind of problem in the production line. This idea goes back to Toyota's Kanban: any assembly line worker can stop the line if they see something going wrong. The line doesn't restart until the problem is fixed. Workers don't have to fear consequences from management for stopping the line; they are trusted, and expected to behave responsibly

The issue lurking behind all of these concerns is, of course, corporate culture. Corporate environments can be hostile to anything other than short-term profitability. That's a consequence of poor court decisions and economic doctrine, particularly in the U.S. But that inevitably leads us to the biggest issue: how to move the needle on corporate culture. Susan Etlinger has suggested that, in a time when public distrust and disenchantment is running high, ethics is a good investment. Upper-level management is only starting to see this; changes to corporate culture won't happen quickly. Users want to engage with companies and organizations they can trust not to take unfair advantage of them. Users want to deal with companies that will treat them and their data responsibly, not just as potential profit or engagement to be maximized. Those companies will be the ones that create space for ethics within their organizations. We, the data scientists, data engineers, AI and ML developers, and other data professionals, have to demand change. We can't leave it to people that "do" ethics. We can't expect management to hire trained ethicists and assign them to our teams. We need to live ethical values, not just talk about them. We need to think carefully about the consequences of our work. We must create space for ethics within our organizations. Cultural change may take time, but it will happen—if we are that change. That's what it means to do good data science

4.4 OWNERS OF DATA

Data owners are either individuals or teams who make decisions such as who has the right to access and edit data and how it's used. Owners may not work with their data every day, but are responsible for overseeing and protecting a data domain.

A data owner is responsible for the data in a particular data domain. They may belong to the steering committee and ensure that the data under their view is governed throughout the organization. Data owners approve data glossaries and definitions as well as initiate data quality activities.

Owning' data – who owns data, what's capable of being owned, and what rights and responsibilities ownership attracts – is gaining a lot of attention.

4.4.1 Responsibilities of the Data Owner

The first responsibility of the data owner is to classify the data correctly. Once a classification has been set it is up to the data owner to determine who has access to the data. Usually, this access is based upon roles as opposed to individuals.

The data classification is one of the most important steps. Data classification has a different meaning for different organizations but at the basic level it is knowing the type of data a company has, determining its value, and categorizing it. For example, if your company has a secret sauce or original intellectual property it may be considered “top secret” or “confidential”. The reason to label or classify it as “top secret” or “confidential” is so it can be handled and ultimately protected appropriately. In addition to not putting the correct controls on data there is the potential to retain data for longer than needed or destroy data before it should be based on laws or contractual commitments. Many people have an opinion on how data should be classified or labeled but at the end of the day it is the responsibility of the data owner who is ultimately accountable for the data to make the final decision. The data owner will have the most knowledge of the use of the data and the value to the company. It is advisable for the data owner to get input from various sources like the data custodian or data users but the data owner has complete control over the data.

- Who has access to the data? Clarify the roles of people who can access the data. Example: Employees can see an organization chart with departments, manager names, and titles but not salary information (Classification = internal). But a very limited audience like HR should only have access to salary data, performance data, or social security numbers (Classification = confidential).
- How is the data secured? Sensitive data elements within HR documentation have been classified to be confidential and therefore it requires additional security controls to protect it. Some of the additional controls to secure confidential data stored in electronic medium could include being saved in a location on the network with appropriate safeguards to prevent unauthorized access (secure folders protected by passwords).
- How long the data is retained? Many industries require that data be retained for a certain length of time. For example, the finance industry requires a seven-year retention period and some health care industries requires a 100-year retention period. Data owners need to know the regulatory requirements for their data and if there is no clear guidance on retention then it should be based off the company’s retention policy or best practices.

- How data should be destroyed? Based on the classification of the data there should be clear guidance on how to dispose or destroy the data. For example:

Information Medium	Public	Internal	Confidential
Hard Copy	Place in recycling bins or trash receptacles.	Place in secured shedding bins or manually shred.	Place in secured shedding bins or manually shred with a cross-cut shredder or pulped. A record must be maintained that indicates the records disposed of and the data of disposal.
Electronic records	Electronic records can be deleted normally.	Electronic records need to be overwritten to 1's and 0's or with a secure delete option.	Electronic records need to be degaussed off magnetic media after securely deleted or the physical media should be destroyed with a record maintained that indicates the records disposed of and the date of disposal.

- What data needs to be encrypted? Data owners should decide whether their data needs to be encrypted. To make this determination the data owner should know the applicable laws or regulation requirements set that must be complied with. A good example of a regulation requirement is set by the Payment Card Industry (PCI) Data Security Standard and it requires that the transmission of cardholder data across open, public networks must be encrypted.

4.4.2 The Importance of Assigning Data Owners

In most organizations, as data passes through different teams and systems, assigning data owners can be cumbersome. However, this is a critical step for GDPR compliance.

Here's why assigning data owners is important:

1. Accountability — Ownership creates accountability. Since GDPR introduces many controls on personal data, assigning responsibilities ensures that data will be continuously monitored for compliance by the owners.

2. Defining policies — As they have a vested interest in the integrity of their data, owners focus on defining policies (for example retention or deletion policies) and standards that ensure the alignment of their data to the GDPR.

3. Creating trusted data — Data ownership is a key ingredient to gain customer trust and achieve measurable business benefits. Poor data could easily result in bad customer

experiences and ultimately losing customers. In particular, when personal data is not reconciled into a data subject 360° view, compliance with data subject access rights, such as rights of portability or rights to be forgotten cannot be fully achieved.

4. Eliminating redundancies — As organizations strive to put the appropriate governance framework in place for GDPR, one common frustration is a loss of productivity. This issue stems from multiple teams addressing the same problem either because there isn't a clear understanding of data or they're not even aware that the problem has been resolved by another team. Federated ownership eliminates these painful issues.

4.4.3 Identification of Data Owners: Three Questions to Ask

The mandatory introduction of a data protection officer (DPO) role by GDPR in most organizations effectively creates a master data owner. Each and every element in a data taxonomy needs an individual owner, however, and there is little likelihood that a single DPO can hold this responsibility on such a large scale. In addition, this could create a security issue. Delegation and segregation of duties are needed.

Asking the right questions helps. Once these questions have been answered, the data owner should become clearer:

1. Who is most impacted by data accuracy?
2. Who has authority to decide the next step?
3. Who owns the related data attributes?

4.5 DIFFERENT ASPECTS OF PRIVACY

First there is the data where security and privacy has always mattered and for which there is already an existing and well galvanized body of law in place. Foremost among these is classified or national security data where data usage is highly regulated and enforced. Other data for which there exists a considerable body of international and national law regulating usage includes:

Proprietary Data – specifically the data that makes up the intellectual capital of individual businesses and gives them their competitive economic advantage over others, including data protected under copyright, patent, or trade secret laws and the sensitive, protected data that companies collect on behalf of its customers;

Infrastructure Data - data from the physical facilities and systems – such as roads, electrical systems, communications services, etc. – that enable local, regional, national, and international economic activity; and

Controlled Technical Data - technical, biological, chemical, and military-related data and research that could be considered of national interest and be under foreign export restrictions.

It may be possible to work with publicly released annualized and cleansed data within these areas without a problem, but the majority of granular data from which significant insight can be gleaned is protected. In most instances, scientists, researchers, and other authorized developers take years to appropriately acquire the expertise, build the personal relationships,

and construct the technical, procedural and legal infrastructure to work with the granular data before implementing any approach. Even using publicly released datasets within these areas can be restricted, requiring either registration, the recognition of or affiliation to an appropriate data governing body, background checks, or all three before authorization is granted.

The second group of data that raises privacy and security concerns is personal data. Commonly referred to as Personally Identifiable Information (PII), it is any data that distinguishes individuals from each other. It is also the data that an increasing number of digital approaches rely on, and the data whose use tends to raise the most public ire. Personal data could include but is not limited to an individual's:

- Government issued record data (social security numbers, national or state identity numbers, passport records, vehicle data, voting records, etc.);
- Law enforcement data (criminal records, legal proceedings, etc.);
- Personal financial, employment, medical, and education data;
- Communication records (phone numbers, texts data, message records, content of conversations, time and location, etc.);
- Travel data (when and where traveling, carriers used, etc.);
- Networks and memberships (family, friends, interests, group affiliations, etc.);
- Location data (where a person is and when);
- Basic contact information (name, address, e-mail, telephone, fax, twitter handles, etc.);
- Internet data (search histories, website visits, click rates, likes, site forwards, comments, etc.);
- Media data (which shows you're watching, music you're listening to, books or magazines you're reading, etc.);
- Transaction data (what you're buying or selling, who you're doing business with, where, etc.); and
- Bio and activity data (from personal mobile and wearable devices).

In industries where being responsible for handling highly detailed personal data is the established business norm – such as in the education, medical and financial fields – there are already government regulations, business practices and data privacy and security laws that protect data from unauthorized usage, including across new digital platforms. But in many other industries, particularly in data driven industries where personal data has been treated as proprietary data and become the foundation of business models, there is currently little to no regulation. In the new normal, the more that a data approach depends on data actively or passively collected on individuals, the more likely that consumers will speak up and demand privacy protection, even if they previously gave some form of tacit approval to use their data.

` Despite this new landscape, there are lots of different ways to use personal data, some of which may not trigger significant privacy or security concerns. This is particularly true in cases where individuals willingly provide their data or data cannot be attributed to an

individual. Whether individuals remain neutral to data approaches tends to be related to the level of control they feel they have over how their personal data is used. Some organizations that collect personal data extensively, such as Facebook and Google, work to increasingly provide their users with methods to control their own data. But for others, the lack of due diligence on data privacy in their approaches has already had their effect.

A third category of data needing privacy consideration is the data related to good people working in difficult or dangerous places. Activists, journalists, politicians, whistle-blowers, business owners, and others working in contentious areas and conflict zones need secure means to communicate and share data without fear of retribution and personal harm.

4.6 FIVE C' S OF DATA SCIENCE

What does it take to build a good data product or service? Not just a product or service that's useful, or one that's commercially viable, but one that uses data ethically and responsibly.

Users lose trust because they feel abused by malicious ads; they feel abused by fake and misleading content, and they feel abused by "act first, and apologize profusely later" cultures at many of the major online companies. And users ought to feel abused by many abuses they don't even know about. Why was their insurance claim denied? Why weren't they approved for that loan? Were those decisions made by a system that was trained on biased data? The slogan goes, "Move fast and break things." But what if what gets broken is society?

Data collection is a big business. Data is valuable: "the new oil," as the Economist proclaimed. We've known that for some time. But the public provides the data under the assumption that we, the public, benefit from it. We also assume that data is collected and stored responsibly, and those who supply the data won't be harmed. Essentially, it's a model of trust. But how do you restore trust once it's been broken? It's no use pretending that you're trustworthy when your actions have proven that you aren't. The only way to get trust back is to be trustworthy, and regaining that trust once you've lost it takes time.

There's no simple way to regain users' trust, but a "golden rule" for data as a starting point: "treat others' data as data scientist would like other to treat their data." However, implementing a "golden rule" in the actual research and development process is challenging. The golden rule isn't enough by itself. There has to be certain guidelines to force discussions with the application development teams, application users, and those who might be harmed by the collection and use of data. Five framing guidelines help us think about building data products. We call them the five Cs: consent, clarity, consistency, control (and transparency), and consequences (and harm).

4.6.1 Consent

The trust between the people who are providing data and the people who are using it cannot be established without agreement about what data is being collected and how that data will be used . Agreement starts with obtaining consent to collect and use data. Unfortunately, the agreements between a service's users (people whose data is collected) and the service itself (which uses the data in many ways) are binary (meaning that you either accept or decline)

and lack clarity. In business, when contracts are being negotiated between two parties, there are multiple iterations (redlines) before the contract is settled. But when a user is agreeing to a contract with a data service, you either accept the terms or you don't get access. It's non-negotiable.

Data is frequently collected, used, and sold without consent. This includes organizations like Acxiom, Equifax, Experian, and Transunion, who collect data to assess financial risk, but many common brands also connect data without consent. In Europe, Google collected data from cameras mounted on cars to develop new mapping products. AT&T and Comcast both used cable set top boxes to collect data about their users, and Samsung collected voice recordings from TVs that respond to voice commands.

4.6.2 Clarity

Clarity is closely related to consent. Users must have clarity about what data they are providing, what is going to be done with the data, and any downstream consequences of how their data is used. All too often, explanations of what data is collected or being sold are buried in lengthy legal documents that are rarely read carefully, if at all. Observant readers of Eventbrite's user agreement recently discovered that listing an event gave the company the right to send a video team, and exclusive copyright to the recordings. And the only way to opt out was by writing to the company. The backlash was swift once people realized the potential impact, and Eventbrite removed the language.

Facebook users who played Cambridge Analytica's "This Is Your Digital Life" game may have understood that they were giving up their data; after all, they were answering questions, and those answers certainly went somewhere. But did they understand how that data might be used? Or that they were giving access to their friends' data behind the scenes? That's buried deep in Facebook's privacy settings. It really doesn't matter which service you use; you rarely get a simple explanation of what the service is doing with your data, and what consequences their actions might have. Unfortunately, the process of consent is often used to obfuscate the details and implications of what users may be agreeing to. And once data has escaped, there is no recourse.

4.6.3 Consistency and Trust

Trust requires consistency over time. You can't trust someone who is unpredictable. They may have the best intentions, but they may not honor those intentions when you need them to. Or they may interpret their intentions in a strange and unpredictable way. And once broken, rebuilding trust may take a long time. Restoring trust requires a prolonged period of consistent behavior.

Consistency, and therefore trust, can be broken either explicitly or implicitly. An organization that exposes user data can do so intentionally or unintentionally. In the past years, we've seen many security incidents in which customer data was stolen: Yahoo!, Target, Anthem, local hospitals, government data, and data brokers like Experian, the list grows longer each day. Failing to safeguard customer data breaks trust—and safeguarding data means nothing if not consistency over time.

4.6.4 Control and Transparency

All too often, users have no effective control over how their data is used. They are given all-or-nothing choices, or a convoluted set of options that make controlling access overwhelming and confusing. It's often impossible to reduce the amount of data collected, or to have data deleted later. A major part of the shift in data privacy rights is moving to give users greater control of their data. For example, Europe's General Data Protection Regulation (GDPR) requires a user's data to be provided to them at their request and removed from the system if they so desire.

4.6.5 Consequences

Data products are designed to add value for a particular user or system. As these products increase in sophistication, and have broader societal implications, it is essential to ask whether the data that is being collected could cause harm to an individual or a group. The unforeseen consequences and the "unknown unknowns" about using data and combining data sets have been witnessed frequently. Risks can never be eliminated completely. However, many unforeseen consequences and unknown unknowns could be foreseen and known, if only people had tried. All too often, unknown unknowns are unknown because we don't want to know.

While Strava and AOL triggered a chain of unforeseen consequences by releasing their data, it's important to understand that their data had the potential to be dangerous even if it wasn't released publicly. Collecting data that may seem innocuous and combining it with other data sets has real-world implications. It's easy to argue that Strava shouldn't have produced this product, or that AOL shouldn't have released their search data, but that ignores the data's potential for good. In both cases, well-intentioned data scientists were looking to help others. The problem is that they didn't think through the consequences and the potential risks.

Many data sets that could provide tremendous benefits remain locked up on servers. Medical data that is fragmented across multiple institutions limits the pace of research. And the data held on traffic from ride-sharing and gps/mapping companies could transform approaches for traffic safety and congestion. But opening up that data to researchers requires careful planning.

4.7 DIVERSITY – INCLUSION

Reports of AI gone wrong abound, and Responsible AI has started to take a foothold in business — Gartner has even added Responsible AI as a new category on its Hype Cycle for Emerging Technologies. Yet when talking about solutions, increasing diversity and making data science a more inclusive field unfortunately don't often top the list. Noelle Silver, Head of Instruction, Data Science, Analytics, and Full Stack Web Development at HackerU, is looking to change that.

A 2018 study revealed that only 15% of data scientists are women, and sadly, a 2020 study found exactly the same results: it seems we haven't managed to move the needle. While

diversity obviously encompasses more than just women, few studies have been able to quantify other types of representation in the field. Inclusivity is similarly difficult to quantify, both in terms of people working on technology and the ways in which technology can be accessed by all. But that doesn't mean there aren't solutions.

Problem

"The reality is that when we train machine learning models with a bunch of data, it's going to make predictions based on that data. If that data comes from a room of people that look the same, talk the same, act the same, they're all friends — it's not a bad scenario. In the moment, you feel like things are good. No one is really seeing any problems; you don't feel any friction. It's very misleading, especially in artificial intelligence. So, you go to market.

The problem, though, is not everyone looks like you, or talks like you, or thinks like you. So even though you found a community of people that built this software that thinks the same, as soon as you go to market and someone other than that starts using it, they start to feel that friction."

Solution

Of course, there's no easy, magic bullet solution to this problem, but foundations of a good start are:

- Committing the time and resources to practice inclusive engineering: This includes, but certainly isn't limited to, doing whatever it takes to collect and use diverse datasets.
- Create an experience that welcomes more people to the field: This might mean looking at everything from education to hiring practices.
- Think beyond regulations: Simply being compliant doesn't necessarily mean experiences are optimized.

4.8 FUTURE TRENDS

Trend 1: Smarter, faster, more responsible AI

Within the current pandemic context, AI techniques such as machine learning (ML), optimization and natural language processing (NLP) are providing vital insights and predictions about the spread of the virus and the effectiveness and impact of countermeasures. AI and machine learning are critical realigning supply and the supply chain to new demand patterns.

Trend 2: Decline of the dashboard

Dynamic data stories with more automated and consumerized experiences will replace visual, point-and-click authoring and exploration. As a result, the amount of time users spends using predefined dashboards will decline. The shift to in-context data stories means that the most relevant insights will stream to each user based on their context, role or use. These dynamic

insights leverage technologies such as augmented analytics, NLP, streaming anomaly detection and collaboration.

Trend 3: Decision intelligence

Decision intelligence brings together a number of disciplines, including decision management and decision support. It encompasses applications in the field of complex adaptive systems that bring together multiple traditional and advanced disciplines. It provides a framework to help data and analytics leaders design, compose, model, align, execute, monitor and tune decision models and processes in the context of business outcomes and behavior. Explore using decision management and modeling technology when decisions need multiple logical and mathematical techniques, must be automated or semi-automated, or must be documented and audited.

Trend 4: X analytics

Gartner coined the term "X analytics" to be an umbrella term, where X is the data variable for a range of different structured and unstructured content such as text analytics, video analytics, audio analytics, etc.

Data and analytics leaders use X analytics to solve society's toughest challenges, including climate change, disease prevention and wildlife protection with analytics capabilities available from their existing vendors, such as cloud vendors for image, video and voice analytics, but recognize that innovation will likely come from small disruptive startups and cloud providers.

Trend 5: Augmented data management

Augmented data management uses ML and AI techniques to optimize and improve operations. It also converts metadata from being used in auditing, lineage and reporting to powering dynamic systems. Augmented data management products can examine large samples of operational data, including actual queries, performance data and schemas. Using the existing usage and workload data, an augmented engine can tune operations and optimize configuration, security and performance. Data and analytics leaders should look for augmented data management enabling active metadata to simplify and consolidate their architectures, and also increase automation in their redundant data management tasks.

Trend 6: Cloud is a given

As data and analytics moves to the cloud, data and analytics leaders still struggle to align the right services to the right use cases, which leads to unnecessary increased governance and integration overhead. Data and analytics leaders need to prioritize workloads that can exploit cloud capabilities and focus on cost optimization and other benefits such as change and innovation acceleration when moving to cloud.

Trend 7: Data and analytics worlds collide

The collision of data and analytics will increase interaction and collaboration between historically separate data and analytics roles. This impacts not only the technologies and

capabilities provided, but also the people and processes that support and use them. The spectrum of roles will extend from traditional data and analytics roles in IT to information explorer, consumer and citizen developer as an example. To turn the collision into a constructive convergence, incorporate both data and analytics tools and capabilities into the analytics stack.

Trend 8: Data marketplaces and exchanges

Data marketplaces and exchanges provide single platforms to consolidate third-party data offerings. These marketplaces and exchanges provide centralized availability and access (to X analytics and other unique data sets, for example) that create economies of scale to reduce costs for third-party data. To monetize data assets through data marketplaces, data and analytics leaders should establish a fair and transparent methodology by defining a data governance principle that ecosystems partners can rely on.

Trend 9: Blockchain in data and analytics

Blockchain technologies address two challenges in data and analytics. First, blockchain provides the full lineage of assets and transactions. Second, blockchain provides transparency for complex networks of participants. Data and analytics should position blockchain technologies as supplementary to their existing data management infrastructure by highlighting the capabilities mismatch between data management infrastructure and blockchain technologies.

Trend 10: Relationships form the foundation of data and analytics value

Graph analytics is a set of analytic techniques that allows for the exploration of relationships between entities of interest such as organizations, people and transactions. It helps data and analytics leaders find unknown relationships in data and review data not easily analyzed with traditional analytics.

4.9 SUMMARY

When combined with ML algorithms, these technologies can be used to comb through thousands of data sources and documents that could help medical and public health experts rapidly discover new possible treatments or factors that contribute to more negative outcomes for some patients.

Data and analytics leaders need to evaluate opportunities to incorporate graph analytics into their analytics portfolios and applications to uncover hidden patterns and relationships. In addition, consider investigating how graph algorithms and technologies can improve your AI and ML initiatives.

4.10 PRACTICE QUESTIONS

Q1. Write a short note on Data Science Ethics.

Q2. What are the privacy aspects of data?

Q3. What are the five C's of data Science

Q4. Discuss briefly the future trends in Data Science.

REFERENCES

1. https://www.knowledgehut.com/blog/big-data/5-best-data-processing-frameworks
2. Mark J. Costello, Peter T. Harris, Bryony Pearce, Andrea Fiorentino, Jean-François Bourillet, Sarah M. Hamylton, "A Glossary of Terminology Used in Marine Biology, Ecology, and Geology", Michael I. Goldstein, Dominick A. DellaSala, Encyclopedia of the World's Biomes, Elsevier, 2020, Pages 471-478, ISBN 9780128160978, https://doi.org/10.1016/B978-0-12-409548-9.11944-X.
3. https://www.solvexia.com/blog/15-big-data-problems-you-need-to-solve
4. https://blog.quantela.com/the-ethical-challenges-of-a-data-science-practitioner/
5. https://towardsdatascience.com/5-key-ai-problems-related-to-data-privacy-f39558290530
6. https://www.oreilly.com/radar/the-five-cs
7. https://builtin.com/data-science
8. https://www.edx.org/course/subject/data-science
9. https://www.geeksforgeeks.org/types-of-sources-of-data-in-data-mining/
10. https://internetofthingsagenda.techtarget.com/definition/sensor-data
11. https://3bplus.nl/how-do-smart-devices-work-sensors-iot-big-data-and-ai/
12. Introducing Data science, big data, machine learning, and more, using python tools, Cielen D., Meysman A., Ali M., Manning Publications Co ., 2016, ISBN: 9781633430037
13. https://www.andrew.cmu.edu/user/nbier/15110/lectures/lec15a_sound_video.pdf

B.Sc.(DATA SCIENCE)

SEMESTER-I

INTRODUCTION TO DATA SCIENCE

UNIT V: DATA WRANGLING COMBINING AND MERGING DATA SETS

STRUCTURE

5.0 OBJECTIVES

The main goal of this module is to help students learn, understand and practice the data science approaches, which include the study of latest data science tools with latest programming languages. The main objectives of this module are data wrangling which includes data discovery, structuring, cleaning, enriching, validating and publishing, combining and merging datasets, data reshaping, pivoting, transformation, string manipulation operations and regular expression.

5.1 INTRODUCTION

Data science become a buzzword that everyone talks about the data science. Data science is an interdisciplinary field that combines different domain expertise, computer programming skills, mathematics and statistical knowledge to find or extract the meaningful or unknown patterns from unstructured and structure dataset.

Data science is useful for extraction, preparation, analysis and visualization of various information. Various scientific methods can be applied to get insight in data.

Data science is all about using data to solve problems. Data has become the fuel of industries. It is most demandable field of 21st century. Every industry require data to functioning, searching, marketing, growing, expanding their business.

The application of areas of data science are health care, fraud detection, disease predicting, real time shipping routes, speech recognition, targeting advertising, gaming and many more.

5.2 DATA WRANGLING

Data wrangling is a key component of any data science project. Data wrangling is a process where one transforms *"row"* data for making it more suitable for analysis and it will improve the quality of the data. Data wrangling is the process of collecting, gathering and transforming of raw data into appropriate format for accessing, analyzing, easy understanding and further processing for better and quick decision making. It is also known as Data Munging or Data Pre-Processing.

Data wrangling is a crucial first steps in the preparation of data for broad analysis of huge amount of data or big data. It requires significant amount of time and efforts. If data wrangling is properly conducted, it gives you insights into the nature of the data. It is not just a single time process but it is an iterative or repetitive process. Each step in the data wrangling process exposes the new potential ways for the data re-wrangling towards deriving the goal of complex data manipulation and analysis.

5.2.1 Steps for Data Wrangling

Data wrangling process includes six core activities:

- Discovery
- Structuring
- Cleaning
- Enriching
- Validating

- Publishing

- **Discovery:**

  Data discovering is an umbrella term. It describes the process to understand the dataset and insight into it. It involves the collection and evaluation of data from various sources and is often used to identify the spot trends and detecting the patterns of the data and gain instant insight.

  Now a day's large business organization or companies have a large amount of data related to the customers, suppliers, production, sells, marketing etc.

  For example, if a company have a customer database, you can identify that the most of the customers are from which part of the city or state or country.

- **Structuring:**

  Data is coming from the various sources with the difference formats. Structuring is necessary because of the different size and shape of the data. Data structuring is the process or actions that change the form or schema of the dataset. Data splitting into the columns, deleting some fields from the dataset, pivoting rows are the form of data structuring.

- **Cleaning:**

  Before start the data manipulation or data analysis, you need to perform the data cleaning. Data cleaning is the process to identify the data quality related issues, such as missing or mismatched values, duplicate records etc. and apply the appropriate transformation to correct, replace or delete those values or records from the dataset to make high quality of data.

- **Enriching:**

  The data is useful for decision making process in the business. The data needed to take business related decision can be stored into multiple files. To gather all necessary insights into single file and you need to enriched your existing dataset by performing joining and aggregating multiple data sources.

- **Validating:**

  After completion of data cleaning and data enriching, you need to check the accuracy of the data. If dataset is not accurate, it might be creating a problem. It is necessary to do the data validation. This is the final check that any missing or mismatched data was not corrected during the transformation process. It is also need to validate that output dataset has the intended structure and content before publishing it.

- **Publishing:**

  After successful completion of data discovering, structuring, cleaning, enriching and validating's, it's a time to published the wrangled output data for the further analytics processes, if any. The published data can be uploaded in the

organization's software or store into the file in a specific location where organizations peoples knows it is ready to use.

5.2.2 Tools for Data Wrangling

There are some tools available for the data wrangling. Some of the popular tools are as follow:

- **Python**
  Python is a most popular general-purpose high-level programming language. There are many popular Python libraries available for data science. The pandas is a most popular and open source library and it becomes a game changer for data science. It is a very fast, flexible, powerful and easy to used library which includes Data Frame to perform more complex operation such as data joining, data merging, data transformation etc. for in data science.

- **R**
  R is also popular, open source and more power tool for data science and management. It also supports many libraries such as dplyr, tidyr, ggplot2 etc. for data manipulation and data visualization.

- **Tabula**
  Tabula is a tool for liberating data tables trapped inside the PDF files. It allows the user to upload the files in PDF format and extract the selected rows and columns from any tables available in the PDF file. It supports to extract this data from PDF to CSV or Microsoft Excel file format.

- **DataWrangler**
  It is an interactive tool for data cleaning. It takes the read word data and transform it into data tables which can be used for further processing or analysis. It also supports to export the data tables in Microsoft Excel, R, Tabula etc.

- **OpenRefine**
  OpenRefine, previously known as GoogleRefine. It is a Java based open source powerful tool for manipulates the huge data. It is used for data loading, understanding, cleaning and transforming from one format to another format. It also supports to extending the data with web services

.

- **CSVKit**
  CSVKit is a suite for command line tools for converting and working with CSV file. It supports the covert the data from Excel to CSV, JSON to CSV, query with SQL etc.

5.3 COMBINING DATASET

Combining is the process to put two or more dataset together for further processing. The pandas library of Python provides easy functionality to combining the dataset together. Here we learn the combining dataset with concat, merge and join functions using pandas library.

- **Concat:** The concat() function is used for combining dataset across the rows or columns.
- **Merge:** The merge() function is used for combining the dataset on common columns.
- **Join:** The join() function is used for combining the dataset on key column or an index.

5.4 CONCATENATING DATASET

The ***"concat"*** function is used to perform the concatenation operation with the data frame along with an axis. The datasets are just stitched together along with axis (rows axis and column axis). Here we concatenate the datasets using pandas.

Syntax:

```
pd.concat(objs, axis, join, join_axes, ignore_index, keys)
```

Here,

- **objs:** This is a sequence or mapping of Series, DataFrame, objects.
- **axis:** This is an axis to concatenate. This value is {*0, 1, ...*}, default is 0.
- **join:** This is used how to handle indexes on another axis(es). This value is {*'inner', 'outer'*}, default is *'outer'*. The outer is used for union operation and inner is used for intersection operation.
- **join_axes:** This is the list of index objects. Its specific indexes to use for the other (n-1) axes instead of performing inner/outer set logic.
- **ignore_index:** This value is Boolean type, default is False. If this value is True, do not use the index values on the concatenation axis. The resulting axis will be labeled 0, ..., n-1.
- **keys:** This is a sequence to add an identifier to the result indexes, default is None.

Example: Here we perform the concatenation operation using two different data frames i.e. df1 and df2.

The below code creates the two different data frame df1 and df2.

```
# Importing pandas library
import pandas as pd

# First dataframe creation
df1 = pd.DataFrame({
      "Name":["Rahul","Shreya","Pankaj","Monika","Kalpesh"],
      "Age":[20, 19, 24, 25, 25],
      "Gender":["Male","Female","Male","Male","Female"],
```

```
        "Course":["B.E.","B.Tech.","MCA","M.Tech.","M.E."]},
        index = [1, 2, 3, 4, 5])

# Second dataframe creation
df2 = pd.DataFrame({
        "Name":["Mayank","Jalpa","Sanjana","Vimal","Raj"],
        "Age":[22, 21, 24, 26, 23],
        "Gender":["Male","Female","Female","Male","Male"],
        "Course":["MBA","MCA","B.E.","B.Tech.","M.Sc."]},
        index = [1, 2, 3, 4, 5])
```

Now we display both data frames df1 and df2.
The below code will display data frame df1.

```
# Display first dataframe
df1
```

The above code will give the following output.

	Name	Age	Gender	Course
1	Rahul	20	Male	B.E.
2	Shreya	19	Female	B.Tech.
3	Pankaj	24	Male	MCA
4	Monika	25	Male	M.Tech.
5	Kalpesh	25	Female	M.E.

The below code will display data frame df2.

```
# Display second dataframe
df2
```

The above code will give the following output.

	Name	Age	Gender	Course
1	Mayank	22	Male	MBA
2	Jalpa	21	Female	MCA
3	Sanjana	24	Female	B.E.
4	Vimal	26	Male	B.Tech.
5	Raj	23	Male	M.Sc.

Now we perform the concatenation operation on both data frames.
The below code will perform the concatenation operation on both the data frame df1 and df2.

```
# Concatenation of both dataframe
df3 = pd.concat([df1, df2])
df3
```

The above code will give the following output, which concatenate the five records of data frame df1 and five records of data frame df2 into single data frame df3 with ten records.

	Name	Age	Gender	Course
1	Rahul	20	Male	B.E.
2	Shreya	19	Female	B.Tech.
3	Pankaj	24	Male	MCA
4	Monika	25	Male	M.Tech.
5	Kalpesh	25	Female	M.E.
1	Mayank	22	Male	MBA
2	Jalpa	21	Female	MCA
3	Sanjana	24	Female	B.E.
4	Vimal	26	Male	B.Tech.
5	Raj	23	Male	M.Sc.

Now we perform the concatenation with axis as an argument.
The below code will perform the concatenation operation on both data frame df1 and df2 with using axis as an argument.

```
# Concatenation of both dataframe with axis as an argument
df3 = pd.concat([df1, df2],axis=1)
df3
```

The above code will give the following output, which concatenate the data frame df1 and data frame df2 into single data frame df3 horizontally with ***axis=1*** as an argument.

SN	Name	Age	Gender	Course	Name	Age	Gender	Course
1	Rahul	20	Male	B.E.	Mayank	22	Male	MBA
2	Shreya	19	Female	B.Tech.	Jalpa	21	Female	MCA
3	Pankaj	24	Male	MCA	Sanjana	24	Female	B.E.
4	Monika	25	Male	M.Tech.	Vimal	26	Male	B.Tech.
5	Kalpesh	25	Female	M.E.	Raj	23	Male	M.Sc.

Now we perform the concatenation with keys as an argument which is associated with specific keys.
The below code will perform the concatenation operation on both data frame df1 and df2 with keys as an argument.

```
# Concatenation of both dataframe with keys as an argument
df3 = pd.concat([df1, df2], keys=['x','y'])
df3
```

The above code will give the following output, which concatenate the data frame df1 and data frame df2 into single data frame df3 with ***x*** and ***y*** as ***keys*** argument.

		Name	Age	Gender	Course
x	**1**	Rahul	20	Male	B.E.
	2	Shreya	19	Female	B.Tech.
	3	Pankaj	24	Male	MCA
	4	Monika	25	Male	M.Tech.
	5	Kalpesh	25	Female	M.E.
y	**1**	Mayank	22	Male	MBA
	2	Jalpa	21	Female	MCA
	3	Sanjana	24	Female	B.E.
	4	Vimal	26	Male	B.Tech.
	5	Raj	23	Male	M.Sc.

Now we perform the concatenation with ***keys*** and ***ignore_index*** as an argument. It follows its own indexing if we set ignore_index is True.
The below code will perform the concatenation operation on both data frame df1 and df2 with ***keys*** and ***ignore_index*** as an argument.

```
# Concatenation of both dataframe using keys argument
df3 = pd.concat([df1, df2], keys=['x','y'], ignore_index=True)
df3
```

The above code will give the following output, which concatenate data frame df1 and data frame df2 into single data frame df3 using ***x*** and ***y*** as ***keys*** arguments and ***ignore_index=True*** argument.

	Name	Age	Gender	Course
0	Rahul	20	Male	B.E.
1	Shreya	19	Female	B.Tech.
2	Pankaj	24	Male	MCA
3	Monika	25	Male	M.Tech.
4	Kalpesh	25	Female	M.E.
5	Mayank	22	Male	MBA
6	Jalpa	21	Female	MCA
7	Sanjana	24	Female	B.E.
8	Vimal	26	Male	B.Tech.
9	Raj	23	Male	M.Sc.

Now we perform the concatenation using append() function.
The below code will perform the concatenation operation on both data frame df1 and df2. The data frame df2 is appended with the data frame df1.

```
# Concatenation of both dataframe using append
df1.append(df2)
```

The above code will give the following output, which concatenate the data frame df2 with the data frame df1.

	Name	Age	Gender	Course
1	Rahul	20	Male	B.E.
2	Shreya	19	Female	B.Tech.
3	Pankaj	24	Male	MCA
4	Monika	25	Male	M.Tech.
5	Kalpesh	25	Female	M.E.
1	Mayank	22	Male	MBA
2	Jalpa	21	Female	MCA
3	Sanjana	24	Female	B.E.
4	Vimal	26	Male	B.Tech.
5	Raj	23	Male	M.Sc.

5.5 MERGING DATASET

The word ***"merge"*** and ***"join"*** both are used relatively interchangeable in SQL, R and Pandas. Both merge and join doing the similar things, but there are separate "merge" and "join" functions in Pandas.

The merging/joining is the process of bringing two or more datasets together into single dataset and aligning the rows from each dataset based on the common attributes or columns.

The ***'merge'*** function is used to perform the merging operation with the data frame. Here we merge the datasets using pandas.

Syntax:

```
pd.merge(left, right, how, on, left_on, right_on, left_index, right_index, sort)
```

Here,

- **left:** This is the first data frame.
- **right:** This is the second data frame.
- **how:** This is the method how to perform the merge operation. The values of this field are one of *'left', 'right', 'inner', 'outer'*, default is *'inner'*.
- **On:** This is the name of column in which action to be perform. This column must be available in both left and right data frame object.
- **left_on:** This is the name of column from left data frame to use as keys.
- **right_on:** This is the name of column from right data frame to use as keys.
- **left_index:** This is using the index (row label) from left data frame as its join keys, if it is True.
- **right_index:** This is used the index (row label) from right data frame as its join keys, if it is True.

- **sort:** This is use to sort the result data frame by join keys in specific order. The value of this field is Boolean, default is True.

Example: Here we perform the merge operation using two different data frames i.e. left and right.

The below code creates two different data frames left and right.

```
# Importing pandas library
import pandas as pd

# Left dataframe creation
left = pd.DataFrame({
      "Rno":[1, 2, 3, 4, 5],
      "Name":["Rahul","Shreya","Pankaj","Monika","Kalpesh"],
      "Course":["B.E.","B.Tech.","MCA","M.Tech.","M.E."]})

# Right dataframe creation
right = pd.DataFrame({
      "Rno":[1, 2, 3, 4, 5],
      "Name":["Mayank","Jalpa","Sanjana","Vimal","Raj"],
      "Course":["B.E.","MBA","MCA","B.Tech.","M.Sc."]})
```

Now we display both data frame left and right.

The below code will display data frame left.

```
# Display left dataframe
left
```

The above code will give the following output.

	Rno	Name	Course
0	1	Rahul	B.E.
1	2	Shreya	B.Tech.
2	3	Pankaj	MCA
3	4	Monika	M.Tech.
4	5	Kalpesh	M.E.

The below code will display the data frame right.

```
# Display right dataframe
right
```

The above code will give the following output.

	Rno	Name	Course
0	1	Mayank	B.E.
1	2	Jalpa	MBA
2	3	Sanjana	MCA
3	4	Vimal	B.Tech.
4	5	Raj	M.Sc.

Now we perform the merge operation on both data frame using ***on*** as an argument.

The below code will perform the merge operation on both data frame left and right using single ***on*** key as an argument.

```
# Merge both left and right dataframe using single on key
pd.merge(left, right, on='Rno')
```

The above code will give the following output, which merge both data frame left and right using single ***on*** key as an argument.

	Rno	Name_x	Course_x	Name_y	Course_y
0	1	Rahul	B.E.	Mayank	B.E.
1	2	Shreya	B.Tech.	Jalpa	MBA
2	3	Pankaj	MCA	Sanjana	MCA
3	4	Monika	M.Tech.	Vimal	B.Tech.
4	5	Kalpesh	M.E.	Raj	M.Sc.

The below code will perform the merge operation on both data frame left and right using multiple ***on*** key as an argument.

```
# Merge left and right dataframe using multiple on keys
pd.merge(left, right, on=['Rno','Course'])
```

The above code will give the following output, which merge both data frame left and right using multiple ***on*** key as an argument.

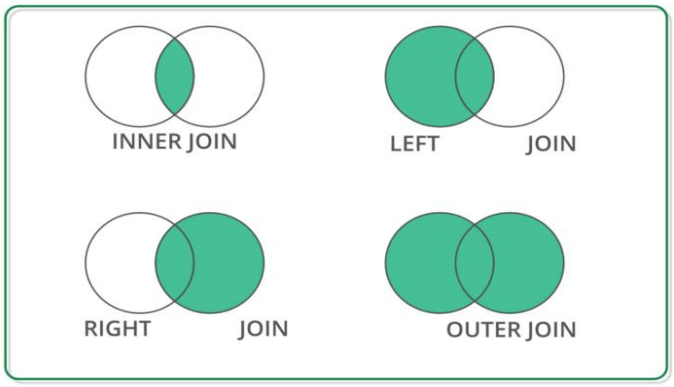
	Rno	Name_x	Course	Name_y
0	1	Rahul	B.E.	Mayank
1	3	Pankaj	MCA	Sanjana

Now we perform the merge operation using ***how*** as an argument. This argument specifies how to determine which keys are to be included in the resulting data frame or table. If the combination does not appear in any of the data frame or table, NA will be display in joined table.

The merge methods are same as SQL join equivalent as below:

Merge Method	SQL Join Equivalent	Description
left	Left Outer Join	Use keys from left object
right	Right Outer Join	Use keys from right object
inner	Inner Join	Use intersection of keys
outer	Full Outer Join	Use unions of keys

It will represent graphically as follows:



The below code will perform the merge operation on both data frame left and right ***on*** course using ***how='left'*** method.

```
# Merge both left and right dataframe using on and how
pd.merge(left, right, on='Course', how='left')
```

The above code will give the following output, which merge both data frame left and right using left join. It will display left data frame records plus common records as follows:

	Rno_x	Name_x	Course	Rno_y	Name_y
0	1	Rahul	B.E.	1.0	Mayank
1	2	Shreya	B.Tech.	4.0	Vimal
2	3	Pankaj	MCA	3.0	Sanjana
3	4	Monika	M.Tech.	NaN	NaN
4	5	Kalpesh	M.E.	NaN	NaN

The below code will perform the merge operation on both data frame left and right ***on*** course using ***how='right'*** method.

```
# Merge both left and right dataframe using on and how
pd.merge(left, right, on='Course', how='right')
```

The above code will give the following output, which merge both data frame left and right using right join. It will display right data frame records plus common records as follows:

	Rno_x	Name_x	Course	Rno_y	Name_y
0	1.0	Rahul	B.E.	1	Mayank
1	NaN	NaN	MBA	2	Jalpa
2	3.0	Pankaj	MCA	3	Sanjana
3	2.0	Shreya	B.Tech.	4	Vimal
4	NaN	NaN	M.Sc.	5	Raj

The below code will perform the merge operation on both data frame left and right ***on*** course using ***how='inner'*** method.

```
# Merge both left and right dataframe using on and how
pd.merge(left, right, on='Course', how='inner')
```

The above code will give the following output, which merge both data frame left and right using inner join. It performs the intersection operation on both data frame, which will display only common records as follows:

	Rno_x	Name_x	Course	Rno_y	Name_y
0	1	Rahul	B.E.	1	Mayank
1	2	Shreya	B.Tech.	4	Vimal
2	3	Pankaj	MCA	3	Sanjana

The below code will perform the merge operation on both data frame left and right ***on*** course using ***how='over'*** method.

```
# Merge both left and right dataframe using on and how
pd.merge(left, right, on='Course', how='outer')
```

The above code will give the following output, which merge both data frame left and right using outer join. It performs the union operation on both data frame, which will display left data frame records, right data frame records and common records as follows:

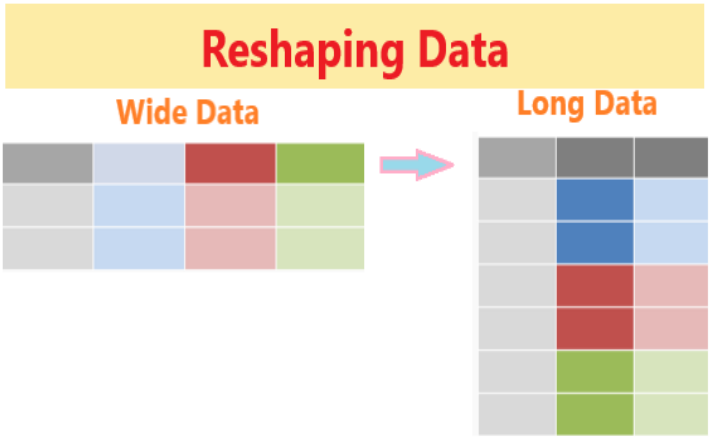
	Rno_x	Name_x	Course	Rno_y	Name_y
0	1.0	Rahul	B.E.	1.0	Mayank
1	2.0	Shreya	B.Tech.	4.0	Vimal
2	3.0	Pankaj	MCA	3.0	Sanjana
3	4.0	Monika	M.Tech.	NaN	NaN
4	5.0	Kalpesh	M.E.	NaN	NaN
5	NaN	NaN	MBA	2.0	Jalpa
6	NaN	NaN	M.Sc.	5.0	Raj

5.6 RESHAPING DATASET

The way in which dataset is arranged into rows and columns is referred as the shape of data. In a dataset, each row represents one observation in a vertical or long data and each column is considered a variable with multiple distinct values.

It is needed to convert or transform the dataset from one format to another format, which is called reshaping of dataset.

Reshaping is the process to change the shape or structure of datasets, such as convert ***"wide"*** data tables into ***"long".*** The below figure shows the reshaping process graphically.



The data frame will be reshaped by using melt(), stack(), unstack() and pivot() functions as follows:

The below code creates and display data frame df1.

```
# Importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
      "Rno":[1, 2, 3, 4, 5],
      "Name":["Rajan","Shital","Mayur","Mittal","Mahesh"],
      "Age":[25, 27, 24, 25, 21]})

# Display dataframe
df
```

The data frame output as follows:

	Rno	Name	Age
0	1	Rajan	25
1	2	Shital	27
2	3	Mayur	24
3	4	Mittal	25
4	5	Mahesh	21

5.6.1 Using melt() Function

We can reshape the data frame using this function. This function is used to wide data frame columns into rows.

The below code will perform the reshape operation using melt function.

```
# Performing melt function on dataframe
df.melt()
```

The above code will give the following output.

	Variable	value
0	Rno	1
1	Rno	2
2	Rno	3
3	Rno	4
4	Rno	5
5	Name	Rajan
6	Name	Shital
7	Name	Mayur
8	Name	Mittal
9	Name	Mahesh
10	Age	25
11	Age	27
12	Age	24
13	Age	25
14	Age	21

5.6.2 Using stack() and unstack() Function

We can reshape the data frame using these functions. The stack() function is used to increase the level of index in a data frame.

The below code will perform the reshape operation using stack function.

```
# Performing stack function on dataframe
df.stack()
```

The above code will give the following output.

0	Rno	1
	Name	Rajan
	Age	25
1	Rno	2
	Name	Shital
	Age	27
2	Rno	3
	Name	Mayur
	Age	24

3	Rno	4
	Name	Mittal
	Age	25
4	Rno	5
	Name	Mahesh
	Age	21
dtype: object		

The unstack() function is used to do the revert back changes in a data frame was perform by the stack() function.

The below code will perform the reshape operation using unstack function.

```
# Performing unstack function on dataframe
df.unstack()
```

The above code will give the following output.

Rno	0	1
	1	2
	2	3
	3	4
	4	5
Name	0	Rajan
	1	Shital
	2	Mayur
	3	Mittal
	4	Mahesh
Age	0	25
	1	27
	2	24
	3	25
	4	21
dtype: object		

5.6.3 Using pivot() Function

We can reshape the data frame using this function. This function is used to reshape the data frame based on the specified column in a data frame.

The below code will perform the reshape operation on **"Rno"** column using pivot function.

```
# Performing pivot function on dataframe
df.pivot(columns='Rno')
```

The above code will give the following output.

	Name					Age				
Rno	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
0	Rajan	NaN	NaN	NaN	NaN	25.0	NaN	NaN	NaN	NaN
1	NaN	Shital	NaN	NaN	NaN	NaN	27.0	NaN	NaN	NaN
2	NaN	NaN	Mayur	NaN	NaN	NaN	NaN	24.0	NaN	NaN
3	NaN	NaN	NaN	Mittal	NaN	NaN	NaN	NaN	25.0	NaN
4	NaN	NaN	NaN	NaN	Mahesh	NaN	NaN	NaN	NaN	21.0

5.7 DATA TRANSFORMATION

Data is collected from the various sources and combine it into a unified data frame. This data frame has large number of columns with different data types.

Data transformation is the process to transform or convert the data as per required format for further processing as and when needed.

The data transformation includes add new columns, find NaN values, drop NaN, replace with mean value, field encoding and decoding, column splitting etc.

5.7.1 Data Frame Creation

To perform the various transformation operation on data, first we have to create the data frame.

The below code will create and display a data frame df which contains the three columns such as **"Name", "Gender"** and **"City"**.

```
#importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
   "Name":["Jayesh Patel","Priya Shah","Vijay Sharma"],
   "Gender": ["Male","Female","Male"],
   "City": ["Rajkot","Delhi","Mumbai"]})

# Display dataframe
df
```

The above code will create and display data frame as follows:

	Name	Gender	City
0	Jayesh Patel	Male	Rajkot
1	Priya Shah	Female	Delhi
2	Vijay Sharma	Male	Mumbai

5.7.2 Missing Value

The dataset contains the many rows and columns. There are some cells in a dataset that have NA or empty cell. This is called missing data in a dataset.

It is needed first to check that missing value before further processing. The common and very simple method to handle this missing value is to delete the rows which contain missing values.

The below code will check the data to containing the missing value or not.

```
df.isna().sum()
```

Here there are no any missing values in each column so it returns zero value in each column.

If there are any missing values in each column, it returns the number of missing values.

```
Name      0
Gender    0
City      0
dtype: int64
```

The below code will drop all the rows which containing missing value.

```
df = df.dropna()
```

The below code will drop the columns where all elements are missing values.

```
df.dropna(axis=1, how='all')
```

The below code will drop the columns where any of the elements containing missing values.

```
df.dropna(axis=1, how='any')
```

The below code will keep only the rows which contains maximum two missing values.

```
df.dropna(thresh=2)
```

The below code will fill all missing values with mean value of the particular column.

```
df.fillna(df.mean())
```

The below code will fill any missing values in specified column with median value of the particular column. Here we taken “Age” column for example.

```
df['Age'].fillna(df['Age'].median())
```

The below code will fill any missing values in specified column with mode value of the particular column. Here we taken “Age” column for example.

```
df['Age'].fillna(df['Age'].mode())
```

5.7.3 Encoding

The dataset contains both numerical and categorical value. The categorical data is not much useful for data processing or analytics. It is needed to encoding the categorical value into numeric value.

Data encoding is the process to convert a categorical variable into a numerical form.

Here we discuss the label encoding which is simply converting each value in a column to a number. Our dataset has **“Gender”** column which has only two values **“male”** and **“female”** It encodes like this:

- Male → 0
- Female → 1

The below code will replace the “Male” to 0 and “Female” to 1.

```
df=df.Gender.replace({"Male":0,"Female":1})
df
```

The above code will display data frame as follows:

```
0   0
1   1
2   0
Name: Gender, dtype: int64
```

5.7.4 Inset New Column

It is needed to add one or more columns in an existing dataset.

The below code will insert a new column in a data frame and display it.

```
df.insert(1, "Age", [21, 23, 24], True)
df
```

The above code will insert a new column **“Age”** with the values **21, 23** and **24** respectively at second column in a data frame. The newly added column data frame will display as follow:

	Name	Age	Gender	City
0	Jayesh Patel	21	Male	Rajkot
1	Priya Shah	23	Female	Delhi
2	Vijay Sharma	24	Male	Mumbai

5.7.5 Split Column

It is needed to split one column into two or more different columns. The process to create two or more different columns from single column in a dataset is called column splitting. Sometimes the **"Full Name"** column of dataset may be need to split into **"First Name"** and **"Last Name"** as a separate column.

The below code will split the column into two different columns and display new data frame.

```
df[['First Name','Last Name']] = df.Name.str.split(expand=True)
df
```

The above code will create two different columns (i.e. First Name, Last Name) from the single column **"Name"** of dataset. The newly split data frame will be display as follow:

	Name	Age	Gender	City	First Name	Last Name
0	Jayesh Patel	21	Male	Rajkot	Jayesh	Patel
1	Priya Shah	23	Female	Delhi	Priya	Shah
2	Vijay Sharma	24	Male	Mumbai	Vijay	Sharma

5.8 STRING MANIPULATION

String manipulation is the process of handling and analyzing the strings. The various operation can be performed on string such as string modification, parsing of string, string conversion etc. The various in-built functions available for string manipulation in different language. He we perform some common string manipulation operations in Python.

We take the following strings as an example.

```
str1 = "Data Science"
str2 = "using"
str3 = "Python"
str4 = "2021"
str5 = "   Data Science   "
```

Function	Description	Example	Output
capitalize()	It converts the first character of string into upper case	str2.capitalize()	'Using'
casefold()	It converts the string into lower case	str1.casefold()	'data science'
center()	It returns the string in center of the specified size	str1.center(15)	' Data Science '
count()	It returns the number of times a specified value occurs in a string	str1.count("a")	2
endswith()	It returns true if the string ends with the specified value	str1.endswith("nce")	True
find()	It searches the string for a specified value and returns the position of where it is found	str1.find("i")	7
format()	It formats the specified values in a string	str4.format()	'2021'
index()	It searches the string for a specified value and returns the position of where it was found	str1.index("c")	6
isalnum()	It returns True if all the characters in a string are alphanumeric	str4.isalnum()	True
isalpha()	It returns True if all characters in a string are alphabet	str2.isalpha()	True
isdigit()	It returns True if all characters in a string are digit	str4.isdigit()	True
islower()	It returns True if all characters in a string are lower case	str2.islower()	True
isnumeric()	It returns True if all characters in a string are numeric	str4.isnumeric()	True
isprintable()	It returns True if all characters in a string are printable	str1.isprintabl()	True
isspace()	It returns True if all characters in a string are whitespaces	str1.isspace()	False
istitle()	It returns True if the string follows the title case rules	str1.istitle()	True
isupper()	It returns True if all characters in a string are upper case letter	str1.isupper()	False
len(string)	It returns the length of a string	len(str1)	12
lower()	It converts the string into lower case	str1.lower()	'data science'
lstrip()	It returns the string with left trim version	str5.lstrip()	'Data Science '
replace(old,new)	It replaces the old string with new string	str1.replace("Science","Analytics")	'Data Analytics'
rfind()	It searches the string for a specified value and returns the last position of	str1.rfind("e")	11

	where it was found		
rindex()	It searches the string for a specified value and returns the last index position of where it is found	str1.rindex("a")	3
rstrip()	It returns the string with right trim version	str5.rstrip()	' Data Science'
split()	It splits the string with specified separator and returns a list	str1.split(" ")	['Data', 'Science']
startswith()	It returns true if the string starts with the specified value	str3.startswith ("P")	True
strip()	It returns the both left and right trim version	str5.strip()	'Data Science'
swapcase()	It returns the swaps cases, lower case becomes upper case and vice versa	str1.swapcase ()	'dATA sCIENCE'
title()	It converts the first character of each word to upper case	str2.title()	'Using'
upper()	It converts a string into upper case	str1.upper()	'DATA SCIENCE'
zfill()	It returns the string with filled by 0 for specified number of times in a string at the beginning	str3.zfill(10)	'0000Python'
+	It concatenates or join the two strings	str1+" "+str2+" "+str3	'Data Science using Python'
*	It repeated the string using n times	str3 * 3	'PythonPythonPython'
string[0]	It returns the first character of a string	str1[0]	'D'
string[7]	It returns the eighth character of a string	str1[7]	'i'
string[2:8]	It returns the string from third character to eighth character	str1[2:8]	'ta Sci'
string[3:]	It returns the string from fourth character to last character	str1[3:]	'a Science'
string[:8]	It returns the string from first character to eighth character	str1[:8]	'Data Sci'
string[-4:]	It returns the last four character of a sting	str1[-4:]	'ence'
string[:-4]	It removes the last four character of a string	str1[:-4]	'Data Sci'

5.9 REGULAR EXPRESSION

Regular expression or RegEx is generally used to identify whether a sequence or character or pattern is exists in a given string or not. It is also used to identify the position of such pattern in a string or file. It mainly used to find and replace specific patterns in a string or file. It helps to manipulate text-based datasets.

Python has a built-in package called ***re***, to work with Regular Expression.

The ***re*** package of Python has set of functions to search the string for matching. The common Python RegEx functions are as follow:

Function	Description
findall	It returns a list of all match values.
search	It returns a match object if match found anywhere in the string.
split	It returns a list and spit the string where each match found
sub	It replaces the one or more matches with a specified string.

5.9.1 The findall() Function

This function returns a list which contains all match values.

Example:

```
import re
string = "Working with Data Science using Python"
result = re.findall("th",string)
result
```

The list contains all match values in an order of they are found.

Output:

```
['th', 'th']
```

Example:

```
import re
string = "Working with Data Science using Python"
result = re.findall("ds",string)
result
```

If no matches found, an empty list will return.

Output:

```
[]
```

5.9.2 The search() Function

This function returns a match object if match found anywhere in the string. Only first occurrence of match will be returned, if there are more than one match found. The none will return if no match found.

Example:

```
import re
string = "Working with Data Science using Python"
result = re.search("wi",string)
```

```
result
```

It found the match value "***wi***".

Output:

```
<re.Match object; span=(8, 10), match='wi'>
```

Example:

```
import re
string = "Working with Data Science using Python"
result = re.search("\s",string)
result
```

It found the match value "**\s**" i.e. space.

Output:

```
<re.Match object; span=(7, 8), match=' '>
```

5.9.3 The split() Function

This function returns a list of where the string has been split at each match found.

Example:

```
import re
string = "Working with Data Science using Python"
result = re.split("\s",string)
result
```

It found the match value **"\s"** i.e. space.

Output:

```
['Working', 'with', 'Data', 'Science', 'using', 'Python']
```

Example:

```
import re
string = "Working with Data Science using Python"
result = re.split("\s",string,2)
result
```

It found the match value **"\s"** i.e. space. But here the string will split into first 2 occurrence of found only. The remaining string will print as it is.

Output:

```
['Working', 'with', 'Data Science using Python']
```

5.9.4 The sub() Function

This function replaces the string with the specified text at each match found.

It will print same string, if not match found.

Example:

```
import re
string = "Working with Data Science using Python"
result = re.sub("\s", "-",string)
result
```

It found the match value **"\s"** i.e. space. Every **"\s"** (space) will replace with the **"-"** (dash).

Output:

```
'Working-with-Data-Science-using-Python'
```

Example:

```
import re
string = "Working with Data Science using Python"
result = re.sub("\s","-",string,2)
result
```

It found the match value **"\s"** i.e. space. But here **"\s"** (space) will replace with the **"-"** (dash) in first two occurrence of found only. The remaining string will print as it is.

Output:

```
'Working-with-Data Science using Python'
```

5.10 SUMMARY

The students will learn many things related to data pre-processing in this module and they will be able to perform the various data science related operation using Python.

- Ability to do the data wrangling which includes data discovery structuring, cleaning, enriching, validating and publishing.
- Ability to do the combining different datasets using concat, merge and join function with different arguments.
- Ability to do the reshaping of dataset using melt, stack and unstack and pivot functions.
- Ability to work with missing value, encoding categorical data into numerical value, splitting dataset etc.
- Ability to perform the various functions related to string and regular expression.

REFERENCES

Books

1. Davy Cielen, Arno D. B. Meysman, Mohamed Ali : Introducing Data Science, Manning Publications Co.
2. Stephen Klosterman (2019) : Data Science Projects with Python, Packt Publishing

3. Jake VanderPlas (2017) : Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly
4. Wes McKinnery and Pandas Development Team (2021) : pandas : powerful Python data analysis toolkit, Release 1.2.3

Web References

1. https://www.geeksforgeeks.org
2. https://www.tutorialspoint.com
3. https://www.w3schools.com
4. https://pandas.pydata.org
5. https://pbpython.com

QUESTIONS

Short Answer:

1. What is data wrangling?
2. What is data cleaning?
3. List tools for data wrangling.
4. What is merging dataset?
5. What is reshaping dataset?

Long Answer:

1. Explain steps for data wrangling process.
2. Explain concat function with example.
3. Explain merge operation with syntax and example.
4. Explain reshaping dataset different functions.
5. Explain string function with example.
6. Explain regular expression with example.

PRACTICALS

1. Create and display a data frame.
2. Create a data frame and find null values and remove it.
3. Create a data frame and inert new column in data frame.
4. Create a data frame and convert categorical data into numerical values.
5. Create a data frame and split the column.
6. Create data frames and combine using concat function.
7. Create data frames and merge with different arguments.
8. Create data frame and reshape using melt function.
9. Perform string manipulation operations on string.
10. Perform regular expression functions on string.

B.Sc.(DATA SCIENCE)

SEMESTER-I

INTRODUCTION TO DATA SCIENCE

UNIT VI: AGGREGATION AND GROUP OPERATIONS GROUP BY MECHANICS

STRUCTURE

6.0 OBJECTIVES

The main goal of this module is to help students learn, understand and practice the data science approaches, which include the study of latest data science tools with latest programming . The main objectives of this module are data aggregation, group wise operation including data splitting, applying and combining, data transformation using lamda function, pivot table, cross tabulation using two-way and three-way cross table and data and time data type.

6.1 INTRODUCTION

Data science become a buzzword that everyone talks about the data science. Data science is an interdisciplinary field that combines different domain expertise, computer programming skills, mathematics and statistical knowledge to find or extract the meaningful or unknown patterns from unstructured and structure dataset.

Data science is useful for extraction, preparation, analysis and visualization of various information. Various scientific methods can be applied to get insight in data.

Data science is all about using data to solve problems. Data has become the fuel of industries. It is most demandable field of 21^{st} century. Every industry require data to functioning, searching, marketing, growing, expanding their business.

The application of areas of data science are health care, fraud detection, disease predicting, real time shipping routes, speech recognition, targeting advertising, gaming and many more.

6.2 Data Aggregation

Data aggregation is the process to gather the raw data and express in a summarised form for statistical analysis. Data may be collected from various sources and combine into a summary format for data analysis.

A dataset contains large amount of data in a rows and columns. There are thousands or more data records are in a single dataset. Data aggregation will useful to access and process the large amount of data quickly. Aggregate data can be access quickly to gain insight instead of accessing all the data records. A single raw of aggregated data can represent this large number of data records over a given time period to calculate the statistics such as sum, minimum, maximum, average and count.

- **Sum :** This function add all the specified data to get a total.
- **Min :** This function displays the lowest value of each specified category.
- **Max :** This function displays the highest value of each specified category.
- **Average :** This function calculates the average value of the specific data.
- **Count :** This function counts the total number of data entries for each category.

Data aggregation provides more insight information based on related cluster of data. For example, a company want to know the sales performance of different district, they would aggregate the sales data based on the district. Data can aggregate by date also, if you want to know the trends over a period of months, quarters, years, etc.

Example: Here, we will perform the aggregation operation on data frame.

The below code will create and display data frame df.

```
#importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
   "Rno":[1,2,3,4,5,6,7,8,9,10],
   "Maths":[67,83,74,91,55,70,86,81,92,67],
   "Physics":[56,67,72,84,89,79,90,89,92,82],
   "Chemistry":[81,88,78,69,74,72,83,90,58,68],
   "Biology":[90,83,86,75,68,79,67,71,91,89],
   "English":[60,55,63,71,88,75,91,82,85,80]})

# Display dataframe
df
```

The above code will give the following output.

	Rno	Maths	Physics	Chemistry	Biology	English
0	1	67	56	81	90	60
1	2	83	67	88	83	55
2	3	74	72	78	86	63
3	4	91	84	69	75	71
4	5	55	89	74	68	88
5	6	70	79	72	79	75
6	7	86	90	83	67	91
7	8	81	89	90	71	82
8	9	92	92	58	91	85
9	10	67	82	68	89	8

Now, we perform the aggregation using min, max and sum.

The below code will find min and max value of different subject of data frame df.

```
df.aggregate(["min","max"])
```

The above code will give the following output.

	Rno	Maths	Physics	Chemistry	Biology	English
Min	1	55	56	58	67	55
Max	10	92	92	90	91	91

The below code will find min and max value and calculate average and sum value of different subject of data frame df.

```
df.agg(["min","max","average","sum"])
```

The above code will give the following output.

	Rno	Maths	Physics	Chemistry	Biology	English
min	1.0	55.0	56.0	58.0	67.0	55.0
max	10.0	92.0	92.0	90.0	91.0	91.0
average	5.5	76.6	80.0	76.1	79.9	75.0
sum	55.0	766.0	800.0	761.0	799.0	750.0

Now, we perform the different aggregation functions on different columns.

The below code will find min and max value of “Maths” subject max and sum of “Physics” subject and min, median and std of “English” subject of data frame df.

```
df.agg({"Maths":["min","max"],
      "Physics":["max","sum"],
      "English":["min","median","std"]})
```

The above code will give the following output.

Here, NaN is display where the specific function is not applying for particular subject.

	Maths	Physics	English
max	92.0	92.0	NaN
median	NaN	NaN	77.500000
min	55.0	NaN	55.000000
std	NaN	NaN	12.400717
sum	NaN	800.0	NaN

Now, we will apply sum function on each column.

The below code will calculate sum of each column of data frame df.

```
df.sum()
```

The above code will give the following output.

Here, the sum of score of each subject will be display.

```
Rno          55
Maths        766
Physics      800
Chemistry    761
Biology      799
English      750
dtype: int64
```

Now, we will apply min function on each column.

The below code will find min value of each column of data frame df.

```
df.min()
```

The above code will give the following output.

Here, the min value of score of each subject will be display.

```
Rno          1
Maths        55
Physics      56
Chemistry    58
Biology      67
English      55
dtype: int64
```

Now, we will apply max function on each column.

The below code will find max value of each column of data frame df.

```
df.max()
```

The above code will give the following output.

Here, the max value of score of each subject will be display.

```
Rno         10
Maths       92
Physics     92
Chemistry   90
Biology     91
English     91
dtype: int64
```

Now, we will apply mean function on each column.

The below code will calculate mean value of each column of data frame df.

```
df.mean()
```

The above code will give the following output.

Here, the mean value of score of each subject will be display.

```
Rno         5.5
Maths       76.6
Physics     80.0
Chemistry   76.1
Biology     79.9
English     75.0
dtype: float64
```

Now, we will apply count function on each column.

The below code will count the numbers of values in each column of data frame df.

```
df.count()
```

The above code will give the following output.

Here, the total numbers of values of score of each subject will be display.

```
Rno         10
Maths       10
Physics     10
Chemistry   10
Biology     10
English     10
dtype: int64
```

Now we will apply std function on each column.

The below code will calculate standard deviation of each column of data frame df.

```
df.std()
```

The above code will give the following output.

Here, the standard deviation of score of each subject will be display.

```
Rno          3.027650
Maths       11.992590
Physics     11.718931
Chemistry    9.859570
Biology      9.230986
English     12.400717
dtype: float64
```

Now, we will apply describe function.

The below code will calculate the basic statistics of each column of data frame df.

```
df.describe()
```

The above code will give the following output.

	Rno	Maths	Physics	Chemistry	Biology	English
count	10.00000	10.00000	10.000000	10.00000	10.000000	10.000000
mean	5.50000	76.60000	80.000000	76.10000	79.900000	75.000000
std	3.02765	11.99259	11.718931	9.85957	9.230986	12.400717
min	1.00000	55.00000	56.000000	58.00000	67.000000	55.000000
25%	3.25000	67.75000	73.750000	69.75000	72.000000	65.000000
50%	5.50000	77.50000	83.000000	76.00000	81.000000	77.500000
75%	7.75000	85.25000	89.000000	82.50000	88.250000	84.250000
max	10.00000	92.00000	92.000000	90.00000	91.000000	91.000000

6.3 GROUP WISE OPERATION

The groupby() function is used to perform the group wise operation on a large dataset. This is a versatile and easy to use function which help to get summary of large dataset. The summary is easy to explore the dataset and shows the relationship between variables.

We can create a grouping of different categories and apply various functions to each category. This function is widely used in real data science projects which dealing with large

amounts of data. It has ability to aggregate data efficiently. This function refers to a process of involving one or more of the following steps:

- **Splitting:** It is a process in which we split the dataset into different groups based on some criteria.
- **Applying:** It is a process in which we apply different functions to each group independently. To apply the function to each group, we perform some operations:
  - **Aggregation:** It is a process to compute a statistical summary of the group such as sum, mean, median, etc.
  - **Transformation:** It is a process to perform some group specific computations and return a like-indexed such as filling NA within group with a value derived from each group.
  - **Filtration:** It is a process to remove some groups based on some criteria such as filtering out dataset based on group wise sum or mean.
- **Combining:** It is a process to combine different datasets after applying groupby function and results will store in a dataset.

The syntax of groupby function is as follows:

Syntax:

```
DataFrame.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, **kwargs)
```

Here,

- **by:** mapping, function, label, str
- **axis:** int, 0 or index and 1 or columns, default is 0, split along rows (0) or columns (1).
- **level:** int, level name, default is None, If the axis is a MultiIndex, group by a particular level or levels.
- **as_index:** boolean, default is True, for aggregated output, return object with group labels as the index.
- **sort:** boolean, default is True, sort group keys.
- **group_keys:** boolean, default is True, when calling apply, add group keys to index to identify pieces.
- **squeeze:** boolean, default is False, reduce the dimensionality of the return type, if possible

Example: Here, we will perform the groupby operation on data frame.

The below code will create and display data frame df.

```
# Importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
```

```
"Product":["Mango","Corn","Orange","Cabbage","Mango","Corn","Watermelon","Apple","Pumkin","Mango",],
"Category":["Fruit","Vegetable","Fruit","Vegetable","Fruit","Vegetable","Fruit","Fruit","Vegetable","Fruit"],
"Qty":[12, 5, 10, 2, 10, 3, 5, 8, 2, 10],
"Price":[350, 80, 320, 50, 200, 50, 280, 380, 60, 400]})



# Display dataframe
df
```

The above code will give the following output.

	Product	Category	Qty	Price
0	Mango	Fruit	12	350
1	Corn	Vegetable	5	80
2	Orange	Fruit	10	320
3	Cabbage	Vegetable	2	50
4	Mango	Fruit	10	200
5	Corn	Vegetable	3	50
6	Watermelon	Fruit	5	280
7	Apple	Fruit	8	380
8	Pumkin	Vegetable	2	60
9	Mango	Fruit	10	400

Now, we will perform the groupby operation on “Category” columns.

The below code will find sum of “Qty” and “Price” based on “Category” on data frame df.

```
df.groupby("Category").sum()
```

The above code will give the following output.

Here, the total Qty of “Fruit” category is 55 and total Price of “Fruit” category is 1930, while total Qty of “Vegetable” category is 12 and total Price of “Vegetable” category is 240.

	Qty	Price
Category		
Fruit	55	1930
Vegetable	12	240

Now, we will perform the groupby operation on “Product” columns.

The below code will find sum of “Qty” and “Price” based on different “Product” on data frame df.

```
df.groupby("Product").sum()
```

The above code will give the following output.

	Qty	Price
Product		
Apple	8	380
Cabbage	2	50
Corn	8	130
Mango	32	950
Orange	10	320
Pumkin	2	60
Watermelon	5	280

Now, we will perform the groupby operation on “Category” columns.

The below code will find mean of “Price” based on “Category” on data frame df.

```
df.groupby("Category")["Price"].mean()
```

The above code will give the following output.

```
Category
Fruit       321.666667
Vegetable    60.000000
Name: Price, dtype: float64
```

The below code will find mean of “Qty” based on “Category” on data frame df.

```
df.groupby("Category")["Qty"].mean()
```

The above code will give the following output.

```
Category
Fruit       9.166667
Vegetable   3.000000
Name: Qty, dtype: float64
```

Now, we will perform the groupby operation on “Product” columns.

The below code will find mean of “Price” based on “Product” on data frame df.

```
df.groupby("Product")['Price'].mean()
```

The above code will give the following output.

```
Product
Apple         380.000000
Cabbage        50.000000
Corn           65.000000
Mango         316.666667
Orange        320.000000
Pumkin         60.000000
Watermelon    280.000000
Name: Price, dtype: float64
```

Now, we will perform the groupby operation on “Category” columns.

The below code will find median of “Price” based on “Category” on data frame df.

```
df.groupby("Category")["Price"].median()
```

The above code will give the following output.

```
Category
Fruit       335
Vegetable    55
Name: Price, dtype: int64
```

The below code will find standard deviation of “Price” based on “Category” on data frame df.

```
df.groupby("Category")["Price"].std()
```

The above code will give the following output.

```
Category
Fruit        73.325757
Vegetable    14.142136
Name: Price, dtype: float64
```

6.4 Transformation

Transformation is a process in which we perform some group-specific computations and return a like-indexed. Transformation perform on a group or a column which returns an object that is indexed the same size of that is being grouped. Thus, the transform should return a result that is the same size as that of a group chunk.

Syntax:

```
DataFrame.transform(func, axis=0, *args, **kwargs)
```

Here,

- **func:** this is the function to use for data transformation.
- **axis:** the axis in which the transformation will perform, {0 or 'index', 1 or 'columns'}, default is 0.
- ***args:** positional arguments to pass in func.
- ****kwargs:** keyword arguments to pass in func.

Example: Here, we will perform some group specific operation and return a like-indexed.

The below code will create and display data frame df.

```
#importing pandas library
import pandas as pd

# Creating the DataFrame
df = pd.DataFrame({
    "A":[8, 7, 15, 12, 15],
    "B":[None, 22, 32, 9, 7],
    "C":[10, 6, None, 8, 14]})

# Display dataframe
df
```

The above code will give the following output.

	A	B	C

0	8	NaN	10.0
1	7	22.0	6.0
2	15	32.0	NaN
3	12	9.0	8.0
4	15	7.0	14.0

Now, we will perform the transform operation using lambda.

The below code will multiply by 5 to each value of all the columns A, B and C of data frame df.

```
result = df.transform(func = lambda x : x * 5)
result
```

The above code will give the following output.

	A	**B**	**C**
0	40	NaN	50.0
1	35	110.0	30.0
2	75	160.0	NaN
3	60	45.0	40.0
4	75	35.0	70.0

The below code will calculate the square root of value of all the columns A, B and C of data frame df.

```
result = df.transform(func = ["sqrt"])
result
```

The above code will give the following output.

	A	B	C
	sqrt	sqrt	sqrt
0	2.828427	NaN	3.162278
1	2.645751	4.690416	2.449490
2	3.872983	5.656854	NaN
3	3.464102	3.000000	2.828427
4	3.872983	2.645751	3.741657

The below code will calculate the exponential of value of all the columns A, B and C of data frame df.

```
result = df.transform(func = ["exp"])
result
```

The above code will give the following output.

	A	B	C
	exp	exp	exp
0	2.980958e+03	NaN	2.202647e+04
1	1.096633e+03	3.584913e+09	4.034288e+02
2	3.269017e+06	7.896296e+13	NaN
3	1.627548e+05	8.103084e+03	2.980958e+03
4	3.269017e+06	1.096633e+03	1.202604e+06

The below code will calculate both square root and exponential together of value of all the columns A, B and C of data frame df.

```
result = df.transform(func = ["sqrt","exp"])
result
```

The above code will give the following output.

	A		B		C	
	sqrt	exp	sqrt	exp	sqrt	exp

0	2.828427	2.980958e+03	NaN	NaN	3.162278	2.202647e+04
1	2.645751	1.096633e+03	4.690416	3.584913e+09	2.449490	4.034288e+02
2	3.872983	3.269017e+06	5.656854	7.896296e+13	NaN	NaN
3	3.464102	1.627548e+05	3.000000	8.103084e+03	2.828427	2.980958e+03
4	3.872983	3.269017e+06	2.645751	1.096633e+03	3.741657	1.202604e+06

The below code will create and display data frame df.

```
#importing pandas library
import pandas as pd

# Creating the DataFrame
df = pd.DataFrame({
"Team":["MI","CSK","RR","MI","KKR","KKR","MI","CSK","KKR",
"RR"],
"Score":[210,150,215,180,185,205,230,190,160,185]})

# Display dataframe
df
```

The above code will give the following output.

	Team	Score
0	MI	210
1	CSK	150
2	RR	215
3	MI	180
4	KKR	185
5	KKR	205
6	MI	230
7	CSK	190
8	KKR	160
9	RR	185

The below code will perform the transformation operation using groupby function.

Here, groupby function will apply on “Team” and calculate the “sum” of “Score” using transform function.

```
df.groupby("Team")["Score"].transform("sum")
```

The above code will give the following output.

Here, the sum of “Score” of each “Team” will be display.

```
0   620
1   340
2   400
3   620
4   550
5   550
6   620
7   340
8   550
9   400
Name: Score, dtype: int64
```

6.5 PIVOT TABLE

Pivot table is a statistical table which summarizes a substantial table like a big dataset. The summary in a pivot tables may include sum, min, max, mean, median or other statistical terms. The pivot() function provides general purpose pivoting with various data type such as string, numeric, etc. The pivot_table() function is used to create pivot table with aggregation of numeric data using data frame of pandas library of Python. The syntax of this function is as follow:

Syntax:

```
DataFrame.pivot(data, index=None, columns=None, values=None, aggfunc)
```

Here,

- **data:** dataframe object
- **index:** a column which has the same length as data. Keys to group by on the pivot table index.
- **columns:** a column which has the same length as data. Keys to group by on the pivot table column.
- **values:** column or list of columns to aggregate
- **aggfunc:** function to use for aggregation

The below example will create a pivot table using pivot_table() function of pandas library of Python.

Example :

The below code will create and display data frame df.

```
# Importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
```

```
"Product":["Mango","Corn","Orange","Cabbage","Mango","Corn","Watermelon","Apple","Pumkin","Mango"],
"Category":["Fruit","Vegetable","Fruit","Vegetable","Fruit","Vegetable","Fruit","Fruit","Vegetable","Fruit"],
"Qty":[12, 5, 10, 2, 10, 3, 5, 8, 2, 10],
"Price":[350, 80, 320, 50, 200, 50, 280, 380, 60, 400]})


# Display dataframe
df
```

The above code will give the following output.

	Product	Category	Qty	Price
0	Mango	Fruit	12	350
1	Corn	Vegetable	5	80
2	Orange	Fruit	10	320
3	Cabbage	Vegetable	2	50
4	Mango	Fruit	10	200
5	Corn	Vegetable	3	50
6	Watermelon	Fruit	5	280
7	Apple	Fruit	8	380
8	Pumkin	Vegetable	2	60
9	Mango	Fruit	10	400

Now, we will perform some examples of pivot table.

Example: To create a pivot table of total sales of each product.

```
# Pivot table of total sales of each product
tot_sales = df.pivot_table(index=["Product"], values=["Price"],aggfunc="sum")

# Display pivot table of total sales
print(tot_sales)
```

Here, we set the index as a “Product” and “sum” as an aggregate function to calculate the total sales of each product. The sum function will do the summation of each products. The above code will give the following output.

```
          Price
Product
Apple        380
Cabbage       50
Corn         130
Mango        950
Orange       320
Pumkin        60
Watermelon    280
```

Example: To create a pivot table of total sales of each category.

```
# Pivot table of total sales of each category
tot_sales = df.pivot_table(index=["Category"], values=["Price"], aggfunc="sum")


# Display pivot table of total sales
print(tot_sales)
```

Here, we set the index as a “Category” and “sum” as an aggregate function to calculate the total sales of each category. The above code will give the following output.

```
          Price
Category
Fruit      1930
Vegetable   240
```

Example: To create a pivot table of total sales of each product.

```
# Pivot table of total sales of both product and category
tot_sales = df.pivot_table(index=["Category","Product"], values=["Price"],
aggfunc="sum")

# Display pivot table of total sales
print(tot_sales)
```

Here, we set the index as a both “Category” and “Product” and “sum” as an aggregate function to calculate the total sales of each product. The above code will give the following output.

```
                Price
Category  Product
Fruit     Apple        380
          Mango        950
          Orange       320
          Watermelon   280
Vegetable Cabbage      50
          Corn         130
          Pumkin       60
```

Example: To create a pivot table to find the minimum, maximum, mean and median of price of each category wise.

```
# Pivot table of min, max, mean and media of sales
tot_sales = df.pivot_table(index=["Category"], values=["Price"],
aggfunc={"min","max","mean","median"})

# Display pivot table of total sales
print(tot_sales)
```

Here, we set the index as a “Category” and “min”, “max”, “mean” and “median” as an aggregate function to calculate the minimum, maximum, mean and median of price of each category wise. The above code will give the following output.

```
          Price
            max        mean median    min
Category
Fruit      400.0  321.666667  335.0  200.0
Vegetable   80.0   60.000000   55.0   50.0
```

Example: To create a pivot table of total product count of each category.

```
# Pivot table of minimum, maximum and average sales
tot_sales = df.pivot_table(index=["Category", "Product"],
values=["Price"],aggfunc=["count"])

# Display pivot table of total sales
print(tot_sales)
```

Here, we set the index as a both “Category” and “Product” and “count” as an aggregate function to count total product of each category. The above code will give the following output.

		count
		Price
Category	**Product**	
Fruit	Apple	1
	Mango	3
	Orange	1
	Watermelon	1
Vegetable	Cabbage	1
	Corn	2
	Pumkin	1

6.6 CROSS TABULATIONS

The cross-tabulation method is used to calculate the simple cross-tabulation of two or more factors. The pandas provide crosstab() function to build a cross-tabulation table which shows the frequency with which certain groups of data appear. The syntax of crosstab() function in pandas is as follow:

Syntax:

```
pd.crosstab(index, columns, values=None, rownames=None, colnames=None,
aggfunc=None, margins=False, margins_name='All', normalize=False,
dropna=True)
```

Here,

- **index:** array-like, values to group by in the rows.
- **columns:** array-like, values to group by in the columns.
- **values:** array-like, optional, array of values to aggregate according to the factors.
- **rownames:** sequence, must match number of row arrays passed, default is None
- **colnames:** sequence, must match number of column arrays passed if passed, default is None
- **aggfuncs:** function, optional, if no values array is passed, its computers a frequency table.
- **margins:** boolean, add row / column margins (i.e. subtotals), default is False.
- **margins_name:** string, name of the row / column that will contain the subtotals if margins is True, default is “All”.
- **normalize:** boolean, {‘all’, ’index’, ’columns’}, or {0,1}, normalize by dividing all values by the sum of values, default is False.
- **dropna:** boolean, do not include columns whose all entries are NaN, default is True.

The below example will create a cross-table using crosstab() function of pandas library of Python.

Example :

The below code will create and display data frame df.

```
# Importing pandas library
import pandas as pd

# Dataframe creation
df = pd.DataFrame({
"Name":["Rahul","Jyoti","Rupal","Rahul","Jyoti","Rupal",
"Rahul","Jyoti","Rupal","Rahul","Jyoti","Rupal","Rahul",
"Jyoti","Rupal","Rahul","Jyoti","Rupal"],
"Examination":["SEM-I","SEM-I","SEM-I","SEM-I","SEM-I",
"SEM-I","SEM-I","SEM-I","SEM-I","SEM-II","SEM-II",
"SEM-II","SEM-II","SEM-II","SEM-II","SEM-II","SEM-II",
"SEM-II"],
"Subject":["Physics","Physics","Physics","Chemistry",
"Chemistry","Chemistry","Biology","Biology","Biology",
"Physics","Physics","Physics","Chemistry","Chemistry",
"Chemistry","Biology","Biology","Biology"],
"Result":["PASS","PASS","FAIL","PASS","FAIL","PASS","FAIL",
"PASS","FAIL","PASS","PASS","PASS","FAIL","PASS","PASS",
"PASS","PASS","FAIL"]})

# Display dataframe
df
```

The above code will give the following output.

	Name	Examination	Subject	Result
0	Rahul	SEM-I	Physics	PASS
1	Jyoti	SEM-I	Physics	PASS
2	Rupal	SEM-I	Physics	FAIL
3	Rahul	SEM-I	Chemistry	PASS
4	Jyoti	SEM-I	Chemistry	FAIL
5	Rupal	SEM-I	Chemistry	PASS
6	Rahul	SEM-I	Biology	FAIL

7	Jyoti	SEM-I	Biology	PASS
8	Rupal	SEM-I	Biology	FAIL
9	Rahul	SEM-II	Physics	PASS
10	Jyoti	SEM-II	Physics	PASS
11	Rupal	SEM-II	Physics	PASS
12	Rahul	SEM-II	Chemistry	FAIL
13	Jyoti	SEM-II	Chemistry	PASS
14	Rupal	SEM-II	Chemistry	PASS
15	Rahul	SEM-II	Biology	PASS
16	Jyoti	SEM-II	Biology	PASS
17	Rupal	SEM-II	Biology	FAIL

Two-way cross table:

There are two columns is used to create a cross table is called two-way cross table.

Here, we will create a cross table of two columns “Subject” and “Result” as follow:

```
# Two-way cross table creation
pd.crosstab(df.Subject, df.Result, margins=True)
```

Here, we set margin=True to display the row wise sum and column wise sum of the cross table. The above code will give the following output.

Result	**FAIL**	**PASS**	**All**
Subject			
Biology	**3**	**3**	**6**
Chemistry	**2**	**4**	**6**
Physics	**1**	**5**	**6**
All	**6**	**12**	**18**

Three-way cross table:

There are three columns is used to create a cross table is called three-way cross table.

Here, we will create a cross table of three columns “Subject”, “Examination” and “Result” as follow:

```
# Three-way cross table creation
pd.crosstab([df.Subject, df.Examination], df.Result, margins=True)
```

Here, we set margin=True to display the row wise sum and column wise sum of the cross table. The above code will give the following output.

	Result	FAIL	PASS	All
Subject	**Examination**			
Biology	**SEM-I**	**2**	**1**	**3**
	SEM-II	**1**	**2**	**3**
Chemistry	**SEM-I**	**1**	**2**	**3**
	SEM-II	**1**	**2**	**3**
Physics	**SEM-I**	**1**	**2**	**3**
	SEM-II	**0**	**3**	**3**
All		**6**	**12**	**18**

6.7 DATE AND TIME DATA TYPE

Python does not have date and time data types, but it has a module named “datetime” can be imported to deal with the date and time. This is inbuilt module available in the Python. This module consists different classes to work with date and time. These classes provide different functions to work with dates, times and time intervals.

There are main six classes in datetime module:

Data Type	Description
Date	It is a date type object. It manipulates only date (i.e. day, month and year).
Time	It is a time object class. It manipulates only time of the any specific day (i.e. hour, minute, second, microsecond).
Datetime	It manipulates the combination of both time and date (i.e. day, month, year, hour, second, microsecond).
timedelta	It manipulates the duration expressing the different between two dates, times or datetime values in milliseconds.
Tzinfo	It is an abstract base class which provides time zone information.

timezone	It is a class that implements tzinfo abstract base class as a fixed offset from the UTC.

There are different format codes is used to formatting the data and time.

The format codes are as follows:

Directive	Description	Example
%a	Day of Week, short version	Fri
%A	Day of Week, full version	Friday
%w	Day of Week as a number from 0 to 6, Here 0 is Sunday	4
%d	Day of Month from 01 to 31	25
%b	Name of Month, short version	Mar
%B	Name of Month, full version	March
%m	Month as a number from 01 to 12	11
%y	Year, short version (in two digit)	21
%Y	Year, full version (in four digit)	2021
%H	Hour from 00 to 23 (in 24 hr format)	16
%I	Hour from 00 to 12 (in 12 hr format)	07
%p	AM or PM	AM
%M	Minute from 00 to 59	35
%S	Second from 00 to 59	45
%f	Microsecond from 000000 to 999999	234567
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year from 001 to 365	325
%U	Week number of year, Sunday as the first day of week, from 00 to 53	40
%W	Week number of year, Monday as the first day of week, from 00 to 53	40
%c	Local version of date and time	Tue Mar 30 13:25:30 2021
%x	Local version of date (MM/DD/YY)	11/24/2021
%X	Local version of time (HH:MM:SS)	18:25:40

To get more insight and work with datetime modules, let's take few examples.

Example: To get current data and time.

```
# To get the current date and time
import datetime as dt
d = dt.datetime.now()
print(d)
```

The above code will give the following output.

```
2021-05-15 00:53:08.925167
```

Here, the now() function is used to display the current local date and time.

Example: To get the current date.

```
# To get the current date only
import datetime as dt
d = dt.date.today()
print(d)
```

The above code will give the following output.

```
2021-05-15
```

Here, the today() function is used to get the current local date.

Example: To get the todays date.

```
# To get the today's date, month and year separately
import datetime as dt
today = dt.date.today()
print(today)
print("Day :",today.day)
print("Month :",today.month)
print("Year :",today.year)
```

The above code will give the following output.

```
2021-05-15
Day : 15
Month : 5
Year : 2021
```

Here, the today() function is used to get the current local date and display day, month and year separately.

Example: To represent a date using date object.

```
# To represent a date using date object
import datetime as dt
d = dt.date(2021, 1, 26)
print(d)
```

The above code will give the following output.

```
2021-01-26
```

Here, the date is passed as an argument.

Example: To represent a date using timestamp.

```
# To get date from a timestamp
import datetime as dt
ts = dt.date.fromtimestamp(987654321)
print(ts)
```

The above code will give the following output.

```
2001-04-19
```

Here, the fromtimestamp() function is used converts seconds into equivalent date.

Example: To represent a time using time object.

```
# To represent a time using time object
import datetime as dt

# time(hour=0, minute=0, second=0)
t = dt.time()
print("Time :",t)

# time(hour, minute, second)
t = dt.time(10, 40, 55)
print("Time :",t)

# time(hour, minute, second)
t = dt.time(hour=10, minute=40, second=55)
print("Time :",t)

# time(hour, minute, second, microsecond)
t = dt.time(10, 40, 55, 123456)
print("Time :",t)

# time(hour, minute, second, microsecond)
t = dt.time(10, 40, 55, 123456)
```

```
print("Hour :",t.hour)
print("Minute :",t.minute)
print("Second :",t.second)
print("Microsecond :",t.microsecond)
```

The above code will give the following output.

```
Time : 00:00:00
Time : 10:40:55
Time : 10:40:55
Time : 10:40:55.123456
Hour : 10
Minute : 40
Second : 55
Microsecond : 123456
```

Here, the time() function with different arguments is used to get the time in different formats.

Example: To represent a datetime object.

```
# To represent a datetime using datetime object
import datetime as dt

# datetime(year, month, day)
dtformat = dt.datetime(2021, 5, 15)
print(dtformat)

# datetime(year,month,day,hour,minute,second,microsecond)
dtformat = dt.datetime(2021, 5, 15, 16, 35, 25, 234561)
print(dtformat)
```

The above code will give the following output.

```
2021-05-15 00:00:00
2021-05-15 16:35:25.234561
```

Here, the datetime() function is used to display the dates in different formats.

Example: To represent a datetime object using different format.

```
# To represent a datetime using datetime object
import datetime as dt

dtformat = dt.datetime(2021, 5, 15, 16, 35, 25, 234561)
print("Year : ",dtformat.year)
print("Month : ",dtformat.month)
```

```
print("Day : ",dtformat.day)
print("Hour : ",dtformat.hour)
print("Minute : ",dtformat.minute)
print("Timestamp : ",dtformat.timestamp())
```

The above code will give the following output.

```
Year :  2021
Month :  5
Day :  15
Hour :  16
Minute :  35
Timestamp :  1621076725.234561
```

Here, the datetime() function is used to display the dates separately in year, month, day, hours, minutes, seconds etc.

Example: To find the difference between to dates and times.

```
# Different between two dates and times
import datetime as dt

# date(year, month, day)
t1 = dt.date(year=2021, month=5, day=15)
t2 = dt.date(year=2020, month=7, day=25)
t3 = t1 - t2
print("Date Difference :",t3)
print("Type of t3 :",type(t3))

# date(year, month, day, hour, minute, second)
t1 = dt.datetime(year=2020, month=1, day=15, hour=8, minute=25, second=45)
t2 = dt.datetime(year=2021, month=4, day=25, hour=10, minute=30, second=50)
t3 = t1 - t2
print("Date Difference :",t3)
print("Type of t3 :",type(t3))
```

The above code will give the following output.

```
Date Difference : 294 days, 0:00:00
Type of t3 : <class 'datetime.timedelta'>
Date Difference : -467 days, 21:54:55
Type of t3 : <class 'datetime.timedelta'>
```

Here, the datetime() function is used to display the dates and perform the subtraction operation between two different dates.

Example: To use of timedelta object.

```
# Different between two timedelta objects
import datetime as dt

t1 = dt.timedelta(weeks=6, days=5, hours=9, minutes=45, seconds=10)
t2 = dt.timedelta(weeks=4, days=3, hours=5, minutes=25, seconds=35)
t3 = t1 - t2
print("Time Delta Difference : ",t3)

t1 = dt.timedelta(weeks=3, hours=10, minutes=45)
t2 = dt.timedelta(days=4, minutes=15, seconds=35)
t3 = t1 - t2
print("Time Delta Difference : ",t3)
```

The above code will give the following output.

```
Time Delta Difference :  16 days, 4:19:35
Time Delta Difference :  17 days, 10:29:25
```

Here, the timedelta() function is used to display the dates and perform the subtraction operation between two different dates.

Example: To represent the time in total seconds.

```
# Time duration in seconds
import datetime as dt
t = dt.timedelta(hours=9, minutes=35, seconds=15)
print("Time in Second :",t.total_seconds())
```

The above code will give the following output.

```
Time in Second : 34515.0
```

Here, the total_seconds() function is used to convert given times into second format.

Example: To use strftime() function for formatting.

```
# Use of strftime() function for date formatting
import datetime as dt

# current date and time
now = dt.datetime.now()
print("Current Date and Time :",now)
```

```
# time in HH:MM:SS format
f1 = now.strftime("%H:%M:%S")
print("Time format is :",f1)

# date and time format DD/MM/YY, HH:MM:SS format
f2 = now.strftime("%d/%m/%Y, %H:%M:%S")
print("Date :",f2)

# Date and time format MM/DD/YY, HH:MM:SS format
f3 = now.strftime("%m/%d/%Y, %H:%M:%S")
print("Date :",f3)
```

The above code will give the following output.

```
Current Date and Time : 2021-05-15 17:16:52.794301
Time format is : 17:16:52
Date : 15/05/2021, 17:16:52
Date : 05/15/2021, 17:16:52
```

Here, the strftime() function is used to display the dates in different format.

Example: To use strptime() function for formatting.

```
# Use of strptime() function for date formatting
import datetime as dt

# date in string format
dt_string = "15 May, 2021"
print("Date in string :",dt_string)

# date in object format
dt_object = dt.datetime.strptime(dt_string, "%d %B, %Y")
print("Date in object :",dt_object)
```

The above code will give the following output.

```
Date in string : 15 May, 2021
Date in object : 2021-05-15 00:00:00
```

Here, the strptime() function is used to string format date into object format date.

Example: To use different timezone.

```
# Use of time zone
import datetime as dt
import pytz

```

```
# Local time zone
local = dt.datetime.now()
print("Local Time Zone:",local.strftime("%m/%d/%y, %H:%M:%S"))

# London time zone
tz_London = pytz.timezone('Europe/London')
dt_London = dt.datetime.now(tz_London)
print("London Time Zone:",dt_London.strftime("%m/%d/%y, %H:%M:%S"))

# Newyork time zone
tz_NY = pytz.timezone('America/New_York')
dt_NY = dt.datetime.now(tz_NY)
print("New York Time Zone:",dt_NY.strftime("%m/%d/%y, %H:%M:%S"))
```

The above code will give the following output.

```
Local Time Zone: 05/15/21, 17:35:15
London Time Zone: 05/15/21, 13:05:15
New York Time Zone: 05/15/21, 08:05:15
```

Here, the pytz is used to set different timezone and strftime() function is used to display the dates in different format.

6.8 SUMMARY

The students will learn many things related to data aggregation and group wise operations in this module and they will be able to perform the various data science related operation using Python.

- Ability to perform the data aggregation using various functions such as min, max, sum, average, count etc.
- Ability to perform group wise operation on specific group such as splitting, applying and combining to calculate the mean, median, standard deviation group wise.
- Ability to do the statistical summary table using pivoting.
- Ability to work with cross tabulation using both two-way and three-way cross table.
- Ability to perform various operation on date and time data types.

REFERENCES

Books

5. Davy Cielen, Arno D. B. Meysman, Mohamed Ali : Introducing Data Science, Manning Publications Co.
6. Stephen Klosterman (2019) : Data Science Projects with Python, Packt Publishing
7. Jake VanderPlas (2017) : Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly

8. Wes McKinnery and Pandas Development Team (2021) : pandas : powerful Python data analysis toolkit, Release 1.2.3, 2021

Web References

6. https://www.geeksforgeeks.org
7. https://www.tutorialspoint.com
8. https://www.w3schools.com
9. https://pandas.pydata.org
10. https://pbpython.com

QUESTIONS

Short Answer:

6. What is data aggregation?
7. What is data splitting?
8. What is transformation?
9. What is pivot table?
10. What is cross tabulation?

Long Answer:

7. Explain data aggregation with different functions.
8. Explain groupby function with example.
9. Explain transform function with syntax and example.
10. Explain pivot table with example.
11. Explain cross tabulation with example.
12. Explain date and time data type with different format code.

PRACTICALS

1. Create a data frame and perform aggregation functions.
2. Create data frames and perform groupby function with different arguments.
3. Create data frames and perform transform function.
4. Create data frame and perform pivot table with different arguments.
5. Create data frame and prepare two-way and three-way cross table.
6. Perform various operation using date and time data types.

SEMESTER-I

INTRODUCTION TO DATA SCIENCE

UNIT VII: DATA MODELING

STRUCTURE

7.0 OBJECTIVES

The main goal of this module is to help students learn, understand and practice the data science approaches, which include the study of latest data science tools with latest programming languages. The main objectives of this module are data modeling which includes the basics of generative modeling and predictive modeling techniques and data visualization which include different types of charts and plots like histogram, scatter plot, time series plot etc.

7.1 INTRODUCTION

Data science become a buzzword that everyone talks about the data science. Data science is an interdisciplinary field that combines different domain expertise, computer programming skills, mathematics and statistical knowledge to find or extract the meaningful or unknown patterns from unstructured and structure dataset.

Data science is useful for extraction, preparation, analysis and visualization of various information. Various scientific methods can be applied to get insight in data.

Data science is all about using data to solve problems. Data has become the fuel of industries. It is most demandable field of 21^{st} century. Every industry require data to functioning, searching, marketing, growing, expanding their business.

The application of areas of data science are health care, fraud detection, disease predicting, real time shipping routes, speech recognition, targeting advertising, gaming and many more.

7.2 INTRODUCTION TO GENERATIVE MODELING

Generative models are the family of machine learning models that are used to describe how data is generated. There are mainly two different types of problems to work with machine learning or deep learning algorithms such as supervised learning and unsupervised learning.

In supervised learning problem, we have two variables such as independent variables (x) and the target variable (y). The examples of supervised learning are classification, regression, object detection etc.

In unsupervised learning problem, we have only independent variables (x). there are no target variable or label. It aims is to find some underlying patterns from the dataset. The examples of unsupervised learning are clustering, dimensionality reduction etc.

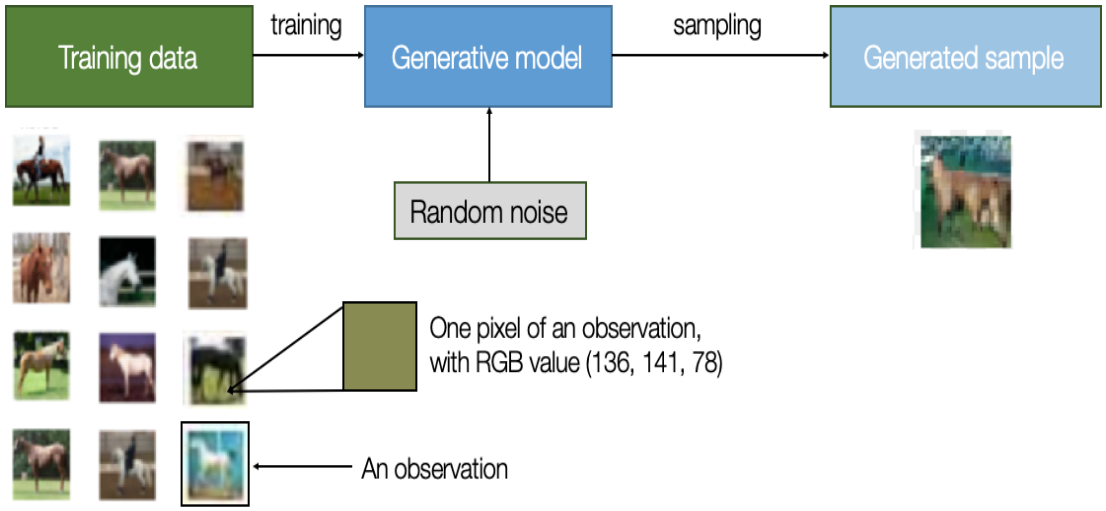
The generative model is an unsupervised learning problem in machine learning. It automatically discovers and learning the rules, regularities or patterns from the large input training dataset. This model learns to create a data that is look like as given. A generative model can be broadly defined as follows:

A generative model describes how a dataset is generated, in terms of a probabilistic model. By sampling from this model, we are able to generate new data.

Generative model can generate new data instances. If we have a dataset containing images of any animal and we may develop a model which can generate a new image of same animal that is never existing but still it looks like as real animal. This model has learned the general

rules that govern the appearance of a specific animal. A generative modelling is used to solve this kind of problems. A generative model process are as follows:

We require a dataset which consists many instants of the entity which we want to generate. This dataset is known as the training data and each data points is called as an observation. The following diagram represent a horse animal dataset.



Generative Model Process (Source: https://www.oreilly.com)

The existing dataset of horse images is used as a training dataset. Based on the training given to the existing dataset it built a generative model to create new images which look like as a real image.

7.3 INTRODUCTION TO PREDICTIVE MODELING

Predictive modeling is a mathematical approach to build models based on existing dataset, which will help to finding the future value or trend of a variable. The variety of statistical techniques including data mining and machine learning are used to estimate or predict the future outcomes.

The predictive modelling is used for every area such as

- Weather forecasting
- Price forecasting
- Demand forecasting
- Sales forecasting
- Customer targeting
- Financial modeling
- Risk assessment
- Market analysis

7.3.1 Types of Models

There are different predictive analytics models are developed for specific applications as follows:

- Classification Model
- Clustering Model
- Forecasting Model
- Time Series Model
- Outlier Model

- **Classification Model**

  This is a simplest and most commonly used predictive analytics model. It works on categorical information based on historical data.

  This model is used or apply in many industrial applications because it can easily retrain with new data as per the needs.

- **Clustering Model**

  This model is use to take the data and divide it into different nested smart groups based on some common attributes. It helps to divide or grouping things or data with shared characteristic or behaviors and take strategic decisions for each group. For example, the customers can be divided based on common attributes like purchasing methods, purchasing power, etc. for targeted marketing campaign to the customers.

- **Forecast Model**

  This is a very popular and most widely use model. It works with the metric value prediction, by estimating the value of new data based on learnings from historical data. It is also used to generate the numerical values and update where none or missing value found. This model can be applied wherever historical numerical data is available. It considers multiple input parameters.

  This model is used in many different business and industries. For example, the company's customer care department can predict how many supports calls they will receive per day.

- **Time Series Model**

  This model is focusses on data where time is an input parameter. This model is applied by using different data points which is taken from the previous year's data to develop a numerical metric that will used to predict the trends within a specified period of time.

  This model is used in many industries which want to see how a particular variable change over a time period. It also takes care about extraneous factors that might be affect the variable such as seasons or seasonal variable. For example, the shopping mall owners want to know the how many customers may visit the mall in week or month.

- **Outliers Model**

This model is work with anomalous data entries in a dataset. It works by finding unusual data, either in isolation or in relation with different categories and numbers. It is more useful in industries were identifying anomalies can save organization corers of rupees such as finance and retail. It is more effective in fraud detection because it can find the anomalies. Since an incidence of fraud is a deviation from the norm, this model is more likely to predict it before it occurs. For example, when identifying a fraud transaction, this model can assess the amount of money lost, purchase history, time, location etc.

7.3.2 Predictive Algorithms

The predictive analytics algorithms can be separated into two things: machine learning and deep learning. These both are subsets of artificial intelligence (AI).

Machine Learning: It deals structural data such as table or spreadsheets. It has both linear and non-linear algorithms. Linear algorithms are quickly train, while non-linear are better optimized for the problems they are to face.

Deep Learning: It is a subset of machine learning. It deals with unstructured data such as images, social media posts, text, audio and videos.

There are several algorithms can be used for machine learning predictive modelling. The most common algorithms are:

- Random Forest
- Generalized Linear Model (GLM) for Two Values
- Gradient Booster Model (GBM)
- K-Means
- Prophet

7.4 CHARTS

Charts is the representation of data in a graphical format. It helps to summarizing and presenting a large amount of data in a simple and easy to understandable formats. By placing the data in a visual context, we can easily detect the patterns, trends and correlations among them.

Python provides various easy to use multiple graphics libraries for data visualization with different features. These libraries are work with both small and large datasets.

Python has multiple graphics libraries with different features. Some of the most popular and commonly used Python data visualization libraries are :

- Matplotlib
- Pandas
- Seaborn
- ggplot
- Plotly

Matplotlib is a most popular, amazing and multi-platform data visualization library available in Python. Matplotlib consists a wide variety of plots like histogram, scatter plot, line or time series plot, bar chart etc.

7.4.1 Histogram

Histogram is a graphical representation of the distribution of numerical data. It contains a rectangular area to display the statistical information which is proportional to the frequency of a variable. It is an estimate of probability distribution of a continuous variable.

In a histogram, the data are binned and the count for each bin is represent. The number of bins is selected so that it is comparable to the typical number of samples in a bin. The bins are specified as consecutive and non-overlapping intervals of a variable. The numbers of bin can be customized also.

There are three basic steps to construct a histogram:

- Bin the range of values
- Distribute the range of values in a series of intervals
- Count the numbers of value into each interval

The ***hist()*** function is used to plot a histogram. It computes and draw the histogram of x values. There is some parameter to construct the histogram are as follows:

- ***bins:*** numbers of bin in the plot, optional
- ***range:*** lower and upper range of the bins.
- ***density***: density or count to populate the plot
- ***histtype:*** types of histogram plot such as bar, step and stepfilled, default is bar
- **align:** control to plot the histogram such as left, mid and right
- ***rwidth:*** relative width of a bar as a fraction of bin width
- ***color:*** it is a color spec or sequence of color specs
- ***orientation:*** horizontal or vertical representation, default is vertical

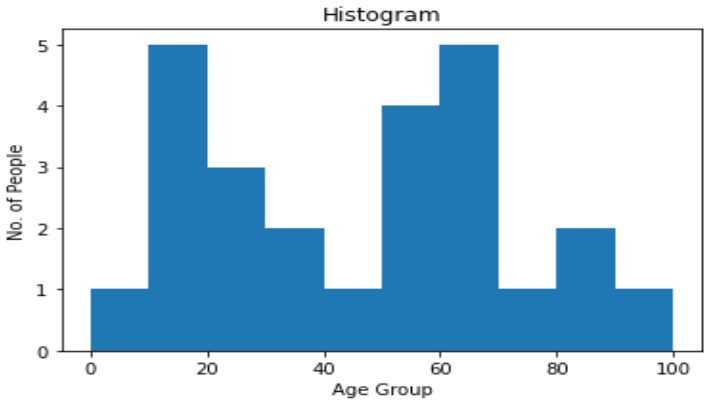
Example: Here we take an example of age of peoples.

The x-axis represents age group or bins and y-axis represents age.

```
# Importing library
import matplotlib.pyplot as plt

# Data values
age = [22, 55, 62, 45, 21, 22, 4, 12, 14, 64, 58, 68, 95, 85, 55, 38, 18, 37, 65, 59, 11, 15, 80, 75, 65]
bins = [0,10,20,30,40,50,60,70,80,90,100]



# Plotting the histogram with title and label
plt.hist(age, bins)
plt.xlabel("Age Group")
```

```
plt.ylabel("No. of People")
plt.title("Histogram")

# Show plot
plt.show()
```

The above code will plot histogram as follow:

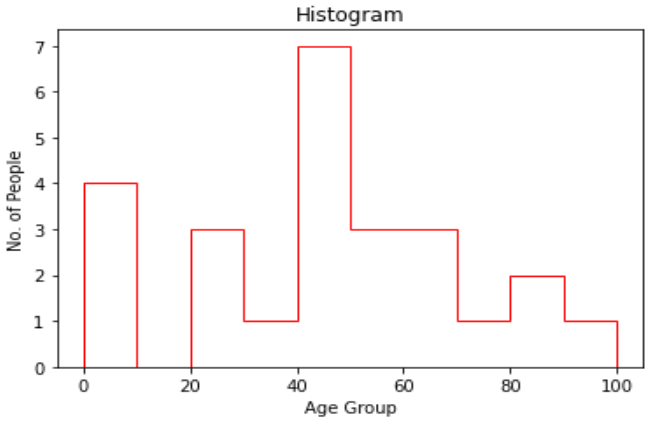


Here, the age is divided into different age group.

We can also set the type and color of histogram and using ***histtype*** and ***color*** as an argument.

```
import matplotlib.pyplot as plt

age = [22, 55, 62, 45, 21, 22, 34, 42, 42, 4, 2, 102, 95, 85, 55, 110, 120, 7, 65, 55,
111, 115, 80, 75, 65, 4, 44, 43, 42, 48]
bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

plt.hist(age, bins, histtype='step', rwidth=0.8, color="red")
plt.xlabel("Age Group")
plt.ylabel("No. of People")
plt.title("Histogram")
plt.show()
```

The above code will plot histogram as follow:



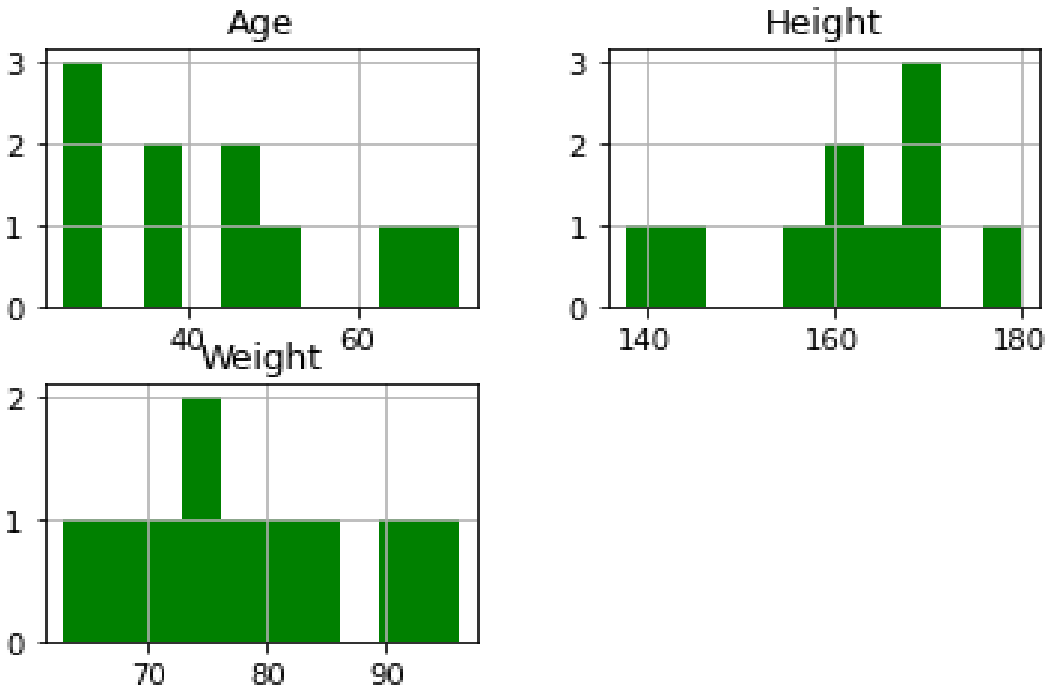
Here, the type of histogram is set to ***step*** and color is set to ***red***.

We can also create the multiple histogram of different columns as follows:

Here, we create a data frame with three different columns such as **"Age", "Height"** and **"Weight".**

```
import pandas as pd

df = pd.DataFrame({
    "Age":[25,38,45,29,65,52,46,72,28,35],
    "Height": [145,138,160,180,165,170,158,162,171,168],
    "Weight": [75,90,85,72,68,76,82,96,63,79]})

hist = df.hist(bins=10)
```

The above code will plot histogram as follow:



7.4.2 Scatter Plot

Scatter plot is diagram where each value in the dataset represent by a dot. It is set of dotted points to represent the individual data on both horizontal and vertical axis to reveal the distribution trends of data.

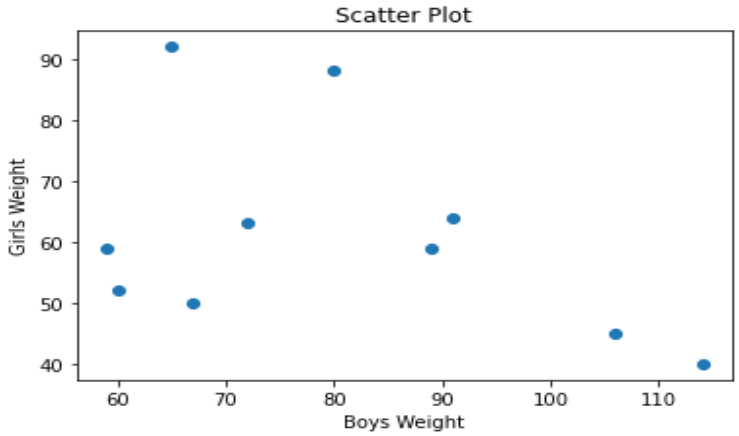
This plot is mostly used for large dataset to highlight the similarities in the dataset. It also shows the outliers and distribution of data.

The **scatter()** function is used to draw the scatter plot. This function plots one dot for each observation. It requires two different arrays of same length for both the x-axis and y-axis. we can also set the scatter plot title and labels on both the axis.

Example: Here we take an example of boys weight and girls weight. The x-axis represents **"Boys_Weight"** and y-axis represents **"Girls_Weight"**.

```
# Importing library
import matplotlib.pyplot as plt

# Data values
Boys_Weight = [67, 89, 72, 114, 65, 80, 91, 106, 60, 59]
Girls_Weight = [50, 59, 63, 40, 92, 88, 64, 45, 52, 59]

# Plotting scatter plot with title and label
plt.scatter(Boys_Weight, Girls_Weight)
plt.title("Scatter Plot")
plt.xlabel("Boys Weight")
plt.ylabel("Girls Weight")

# Show plot
plt.show()
```

The above code will create scatter plot as follow:

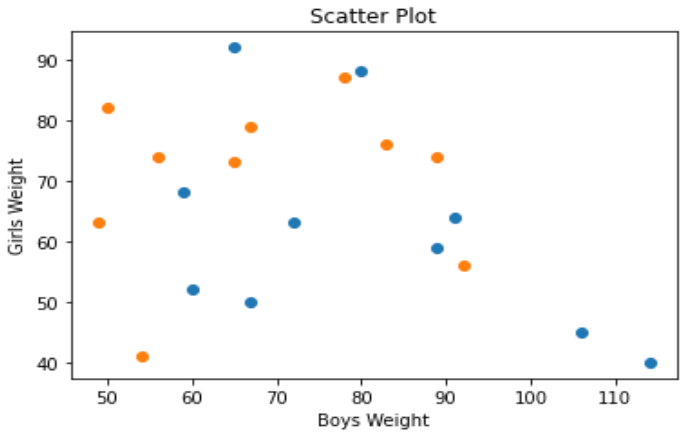


In above plot, we can see the relationship between boys and girls weight.

We can also compare the boys weight and girls weight of one class with another class.

```
import matplotlib.pyplot as plt
import numpy as np

Boys_Weight = [67, 89, 72, 114, 65, 80, 91, 106, 60, 59]
Girls_Weight = [50, 59, 63, 40, 92, 88, 64, 45, 52, 68]
plt.scatter(Boys_Weight, Girls_Weight)

Boys_Weight = [54, 67, 92, 56, 83, 65, 89, 78, 50, 49]
Girls_Weight = [41, 79, 56, 74, 76, 73, 74, 87, 82, 63]
plt.scatter(Boys_Weight, Girls_Weight)

plt.title("Scatter Plot")
plt.xlabel("Boys Weight")
plt.ylabel("Girls Weight")

plt.show()
```

The above code will create scatter plot as follow:



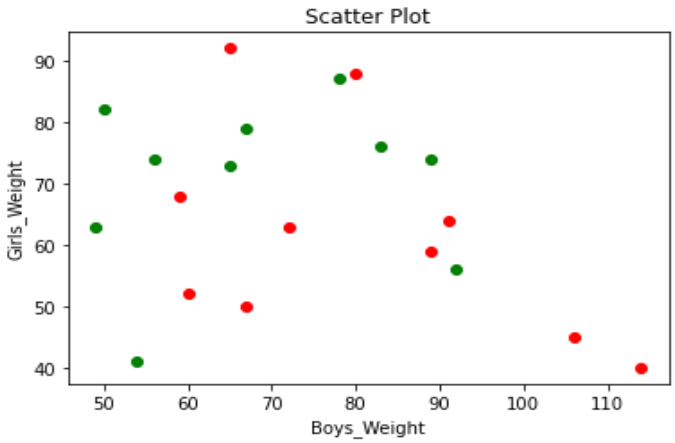
In above plot, we can see the relationship between boys and girls weight of one class with another class by using different colors.

We can also set or change the color of both the classes of data using ***color*** as an argument. Here we set the ***red*** color for first class students and ***green*** color for second class students.

```
import matplotlib.pyplot as plt
import numpy as np

Boys_Weight = [67, 89, 72, 114, 65, 80, 91, 106, 60, 59]
```

```
Girls_Weight = [50, 59, 63, 40, 92, 88, 64, 45, 52, 68]
plt.scatter(Boys_Weight, Girls_Weight, color="red")

Boys_Weight = [54, 67, 92, 56, 83, 65, 89, 78, 50, 49]
Girls_Weight = [41, 79, 56, 74, 76, 73, 74, 87, 82, 63]
plt.scatter(Boys_Weight, Girls_Weight, color="Green")

plt.title("Scatter Plot")
plt.xlabel("Boys_Weight")
plt.ylabel("Girls_Weight")

plt.show()
```

The above code will create scatter plot as follow:

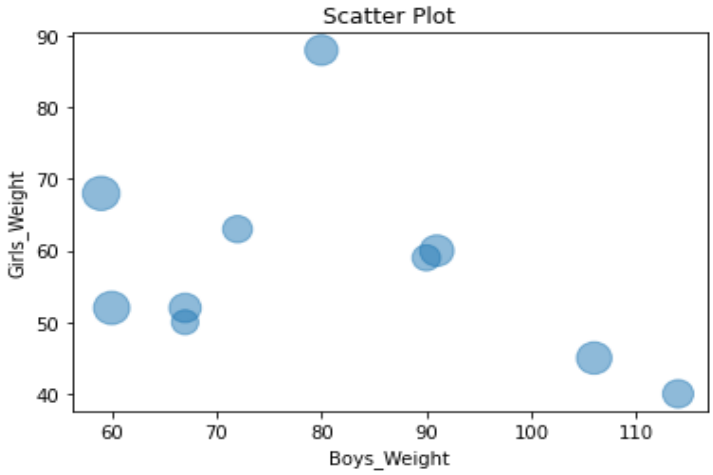


In above plot, we can see the relationship between both classes. Here, red color is used for first class students and green color is used for second class students.

We can also set or change the size of dots using ***s*** as an argument in scatter plot.

```
import matplotlib.pyplot as plt
import numpy as np

Boys_Weight = [67, 90, 72, 114, 67, 80, 91, 106, 60, 59]
Girls_Weight = [50, 59, 63, 40, 52, 88, 60, 45, 52, 68]
size = [200, 220, 240, 260, 280, 300, 320, 340, 360, 380]
plt.scatter(Boys_Weight, Girls_Weight, s=size, alpha=0.5)

plt.title("Scatter Plot")
plt.xlabel("Boys_Weight")
plt.ylabel("Girls_Weight")

plt.show()
```

The above code will create scatter plot as follow:

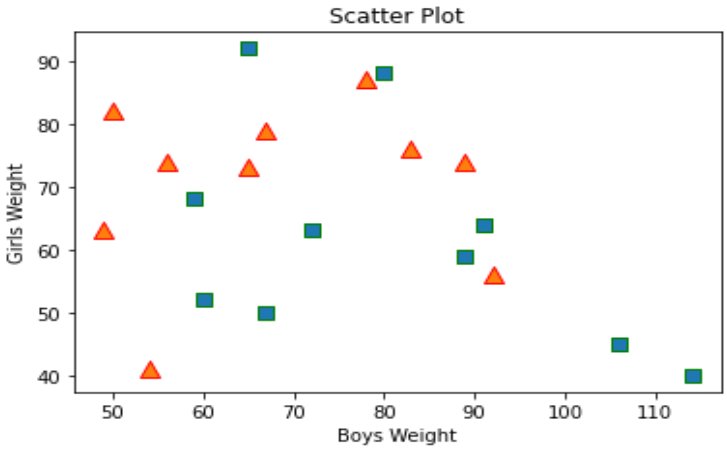


In above plot, we can see the size of each dots.

We can also set the shape instead of dots in scatter plot. Here we set the ***marker*** and ***edgecolor*** as an argument in scatter function to set the shape with edge color in scatter plot.

```
import matplotlib.pyplot as plt
import numpy as np

Boys_Weight = [67, 89, 72, 114, 65, 80, 91, 106, 60, 59]
Girls_Weight = [50, 59, 63, 40, 92, 88, 64, 45, 52, 68]
plt.scatter(Boys_Weight, Girls_Weight, marker ="s", edgecolor ="green", s=50)

Boys_Weight = [54, 67, 92, 56, 83, 65, 89, 78, 50, 49]
Girls_Weight = [41, 79, 56, 74, 76, 73, 74, 87, 82, 63]
plt.scatter(Boys_Weight, Girls_Weight, marker ="^", edgecolor ="red", s=100)

plt.title("Scatter Plot")
plt.xlabel("Boys Weight")
plt.ylabel("Girls Weight")

plt.show()
```

The above code will create scatter plot as follow:



In above plot, we can see the shape with edge color.

7.4.3 Line Chart

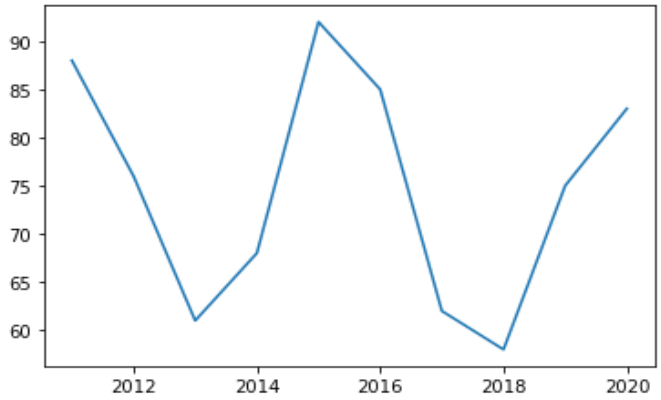
Line chart is used to shows the relation between two datasets on a different axis. There are multiple features available such as line color, line style, line width etc. It is also known as time series plot.

Matplotlib is most popular library for plotting different chart. Line chart is one of them. The ***plot()*** function is used to create a line chart. Here we will see some examples of line chart in Python.

Example: Here we take an example of numbers of students enroll in specific course in different year. The x-axis represents **"Year"** values and y-axis represents **"Student"**.

```
# Importing library
from matplotlib import pyplot as plt

# Data values
Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

# Plotting the line
plt.plot(Year, Student)

# Show plot
plt.show()
```

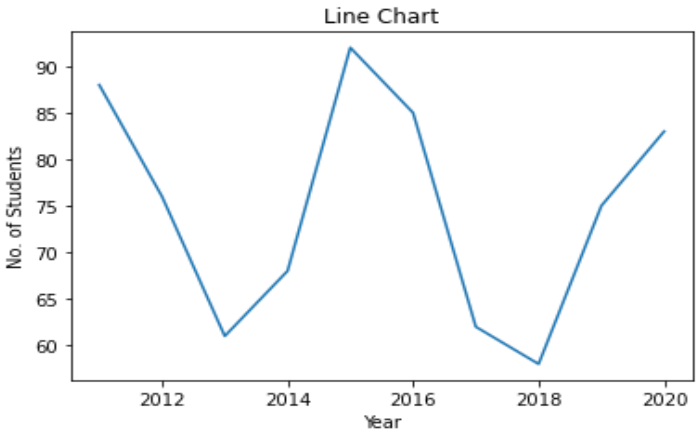
The above code will create line chart as follow:



In this chart, there is no label on both the axis and title of chart. Label is required to understand the dimensions of chart. The following code will create the line chart with ***title*** and ***labeled*** on both axes.

```
from matplotlib import pyplot as plt

Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")

plt.plot(Year, Student)

plt.show()
```

The above code will create line chart as follow:

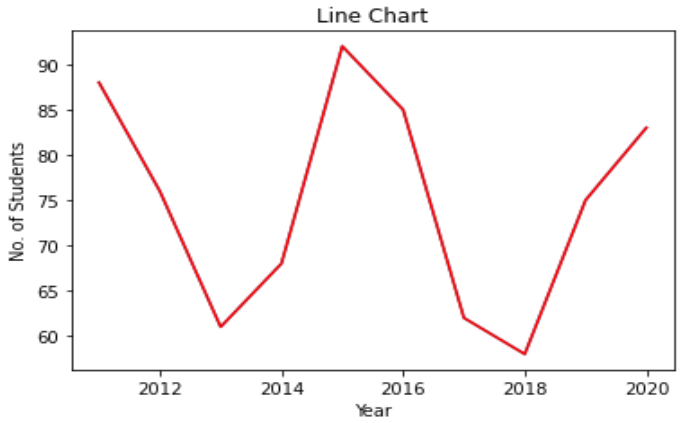


We can set the line color also using ***color*** as an argument. Here we set ***red*** color to the line.

```
from matplotlib import pyplot as plt
```

```
Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")

plt.plot(Year, Student)

plt.plot(Year, Student, 'red')

plt.show()
```

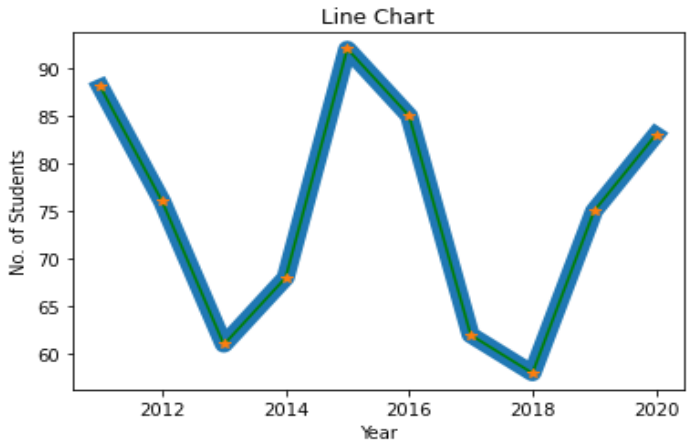
The above code will create line chart as follow:



We can set the line width also using ***linewidth*** or ***lw*** as an argument. Here we set ***10*** to the linewidth.

```
from matplotlib import pyplot as plt

Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")

plt.plot(Year, Student)

plt.plot(Year, Student, 'green', linewidth=10)
plt.plot(Year, Student,'*')
plt.show()
```

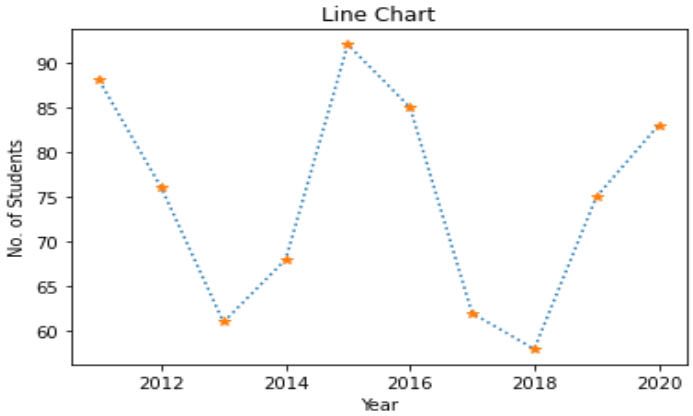
The above code will create line chart as follow:



We can set the line style also using ***linestyle*** or ***ls*** as an argument. There are various types of style available such as solid, dotted, dashed and dashdot. Here we set ***dotted*** as a linestyle in line chart.

```
from matplotlib import pyplot as plt

Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")

plt.plot(Year, Student, linestyle = 'dotted')

plt.plot(Year, Student,'*')

plt.show()
```

The above code will create line chart as follow:

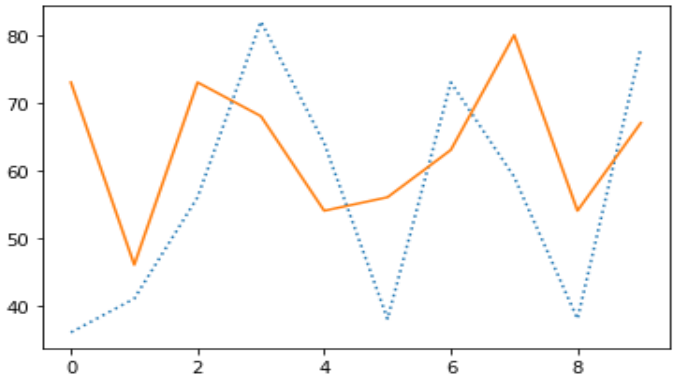


We can set the multiple lines in a single line chart. Here x-axis and y-axis represent the different values. We plot the line chart separately for both the axis. Here we set the ***dotted*** as a linestyle in x-axis.

```
from matplotlib import pyplot as plt

X = [36,41,56,82,64,38,73,59,38,78]
Y = [73,46,73,68,54,56,63,80,54,67]

plt.plot(X, linestyle = 'dotted')

plt.plot(Y)

plt.show()
```

The above code will create line chart as follow:

Here the dotted line represents the x-axis and solid line represent the y-axis.



7.4.4 Bar Chart

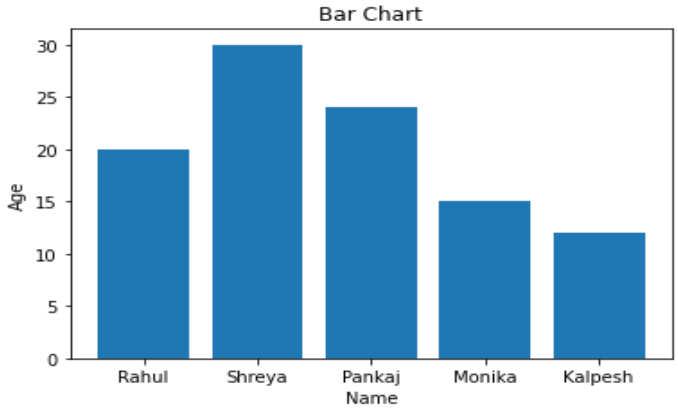
Bar chart or bar plot is representing the category of data with rectangular bars with different heights and lengths with reference to the values that they present. The ***bar()*** function is used to create a bar chart. The bar chart can be plotted both horizontally and vertically.

The bar chart describes the comparisons between distinct categories. One axis represents the particular categories being compared and another axis represent the measured values respected to those categories. The numerical values of variables in a dataset represent the height or length of bar.

Example: Here we take an example of students name and age. The x-axis represents **"Name"** and y-axis represents **"Age"**. Here we also set the chart title and labels on both the axis.

```
# Importing library
from matplotlib import pyplot as plt
import numpy as np

Name = np.array(["Rahul", "Shreya", "Pankaj", "Monika", "Kalpesh"])
Age = np.array([20, 30, 24, 15, 12])

# Labelling the axes and title
plt.title("Bar Chart")
plt.xlabel("Name")
plt.ylabel("Age")

# Plotting the bar
plt.bar(Name, Age)
```

The above code will create bar chart as follow:

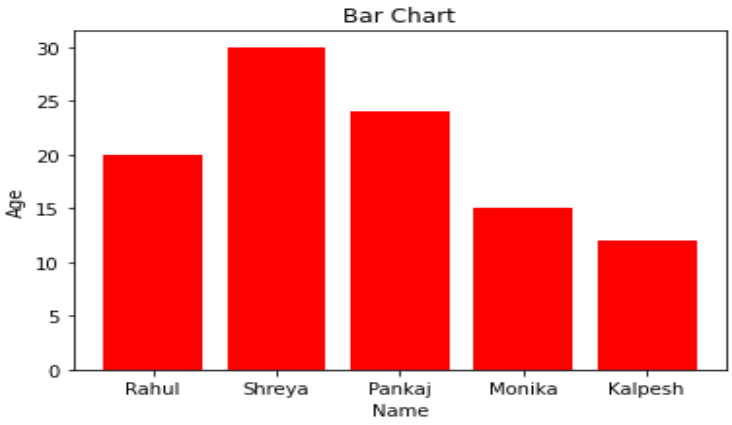


We can change the bar color also. We set ***color*** as an argument to change the color of bar. Here we set the ***red*** as a color of bar.

```
from matplotlib import pyplot as plt
import numpy as np

Name = np.array(["Rahul", "Shreya", "Pankaj", "Monika", "Kalpesh"])
Age = np.array([20, 30, 24, 15, 12])
```

```
plt.title("Bar Chart")
plt.xlabel("Name")
plt.ylabel("Age")

plt.bar(Name, Age, color="red")
```

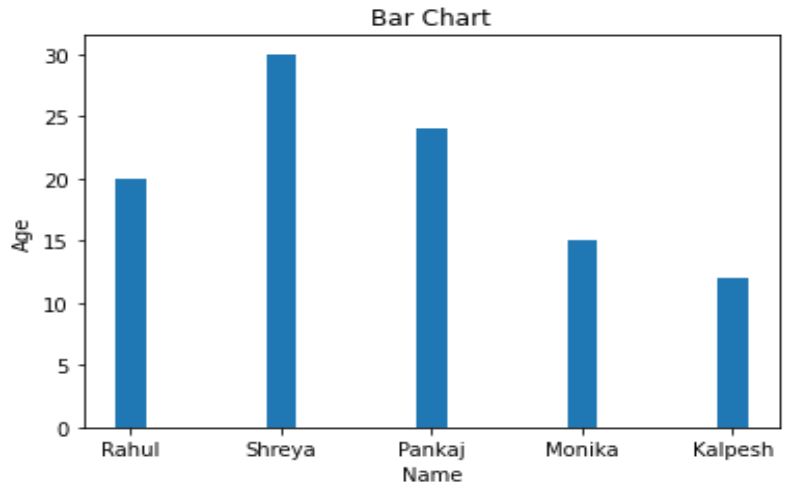
The above code will create bar chart as follow:



We can set the bar width also. We set ***width*** as an argument to set the width of bar. Here we set ***0.2*** as a width of bar.

```
from matplotlib import pyplot as plt
import numpy as np

Name = np.array(["Rahul", "Shreya", "Pankaj", "Monika", "Kalpesh"])
Age = np.array([20, 30, 24, 15, 12])

plt.title("Bar Chart")
plt.xlabel("Name")
plt.ylabel("Age")

plt.bar(Name, Age, width=0.2)
```

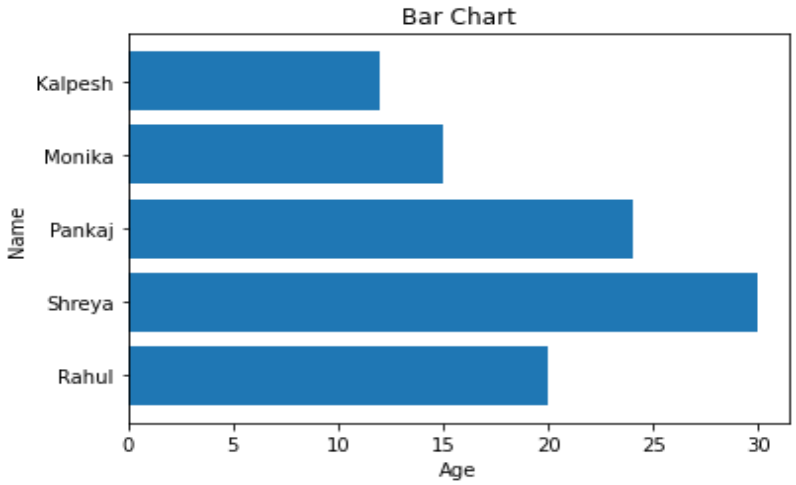
The above code will create bar chart as follow:



We can display bar horizontally also instead of vertically. We set the bar horizontally by using ***barh()*** function.

```
from matplotlib import pyplot as plt
import numpy as np

Name = np.array(["Rahul", "Shreya", "Pankaj", "Monika", "Kalpesh"])
Age = np.array([20, 30, 24, 15, 12])

plt.title("Bar Chart")
plt.xlabel("Age")
plt.ylabel("Name")

plt.barh(Name, Age)
```

The above code will create bar chart as follow:



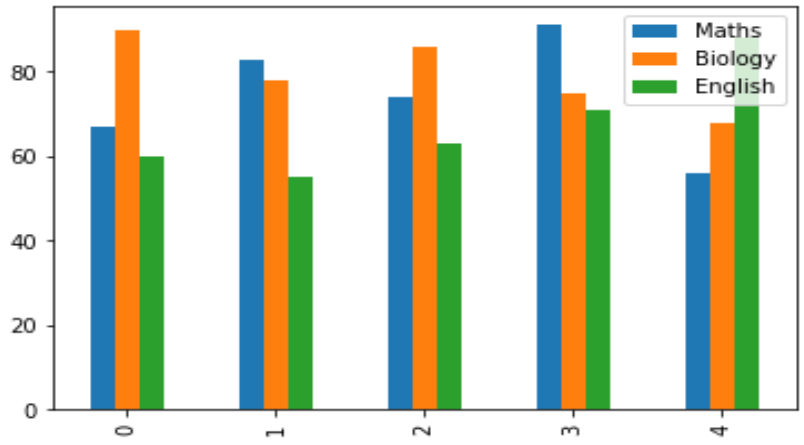
The multiple bar chart is used to represent the comparison among the different variables in a dataset. We can set the thickness and positions of bars also.

Here, we take an example of marks of different subject with the name of students. X-axis represents the students and y-axis represents the marks of different subjects.

```
from matplotlib import pyplot as plt
import pandas as pd

df = pd.DataFrame({
"Name":["Rahul","Shreya","Pankaj","Monika","Kalpesh"],
"Maths":[67,83,74,91,56],
"Biology":[90,78,86,75,68],
"English":[60,55,63,71,88]})

df.plot.bar()
```

The above code will create bar chart as follow:

Here, we show the comparisons of marks of different subjects of the students.



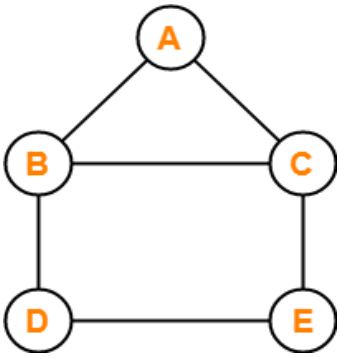
7.5 GRAPH

Graph is a pictorial representation of a set of objects. Some pairs of objects are connected through links. The interaction of link is denoted by points which is known as ***vertices***. The link which is used to connect the vertices is called ***edges***. We can perform some operation on graph such as:

- Display vertices
- Display edges
- Add new vertex
- Add new edge
- Create graph

The dictionary data type is used to present a graph in Python. The vertices of a graph are representing as the keys of dictionary and the links between the vertices also called edges which represent as the values of dictionary

Take the following graph as an example.



The above graph consists the following vertices (V) and edges (E).

```
V = {A, B, C, D, E}
E = {AB, AC, BC, BD, CE, DE}
```

The above graph represents using Python as below.

```
# Create the dictionary with graph elements
graph = { "a" : ["b","c"],
          "b" : ["a","c", "d"],
          "c" : ["a","b", "e"],
          "d" : ["b","e"],
          "e" : ["c","d"]
        }

# Print the graph
print(graph)
```

The code will give the following output.

```
{'a': ['b', 'c'], 'b': ['a', 'c', 'd'], 'c': ['a', 'b', 'e'], 'd': ['b', 'e'], 'e': ['c', 'd']}
```

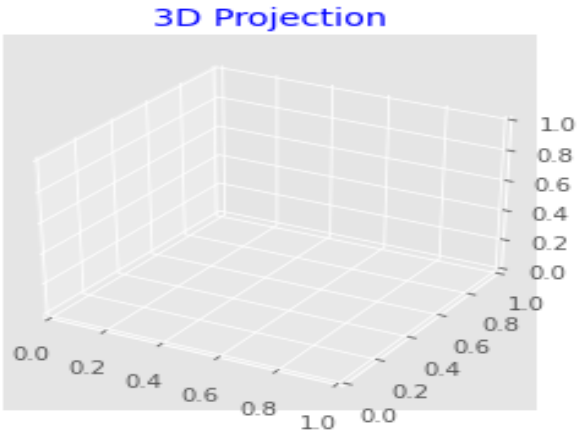
7.6 3D VISULIZATION AND PRESENTATION

The matplotlib library is most popular for data visualization in Python. It was initially designed for two-dimension plotting, but some three-dimension plotting utilities were built on top matplotlib's two-dimension display in later versions. Three dimensional plots are enabling by importing the mplot3d toolkit, which included with the main matplotlib.

A three-dimensional axis can be created by using the keyword ***projection="3d"*** as follows.

```
# Importing library
import matplotlib.pyplot as plt

# 3D projection plot
fig = plt.figure()
ax = plt.axes(projection ="3d")
plt.title("3D Projection", color="blue")
```

The above code will plot as follow:



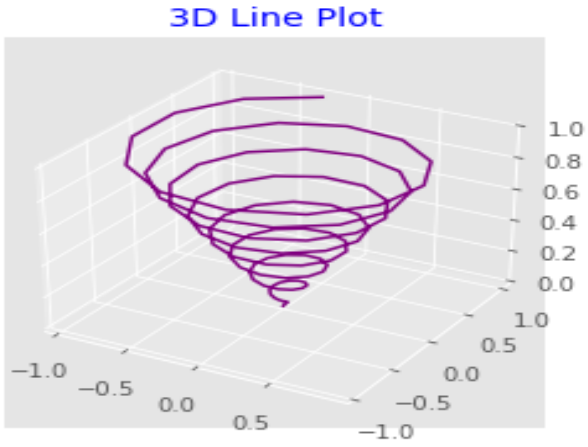
7.6.1 3D Line Plot

This is a most basic three-dimensional plot created using set of (x, y, z) triples. It is also known as time series plot. This plot is plotted using ***ax.plot3D*** function as follow:

```
# Importing library
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

# 3D projection
fig = plt.figure()
ax = plt.axes(projection = "3d")

# All three axis
z = np.linspace(0, 1, 100)
x = z * np.sin(50 * z)
y = z * np.cos(50 * z)

# 3D Line plotting
ax.plot3D(x, y, z, "purple")
ax.set_title("3D Line Plot", color="blue")
```

```
plt.show()
```

The above code will plot as follow:

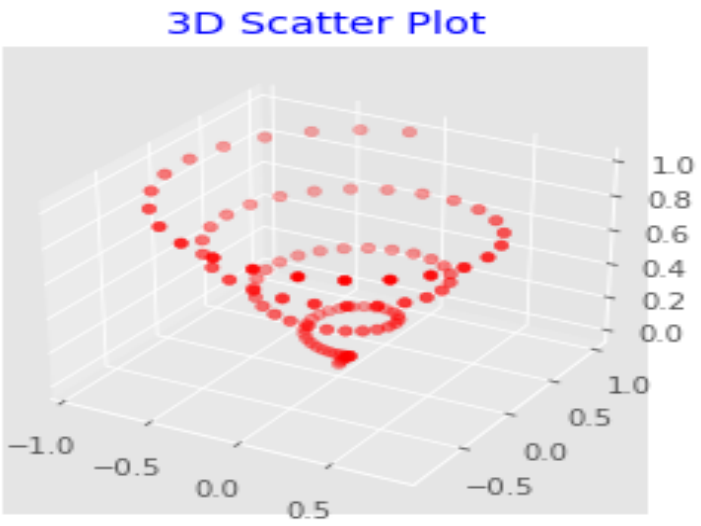


7.6.2 3D Scatter Plot

This is a basic three-dimensional plot created using set of (x, y, z) triples. It represents the data points on three axes to show the relationship between three variables. This plot is plotted using ***ax.scatter3D*** function as follow:

```
# Importing library
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

# 3D projection
fig = plt.figure()
ax = plt.axes(projection = "3d")

# All three axis
z = np.linspace(0, 1, 100)
x = z * np.sin(25 * z)
y = z * np.cos(25 * z)

# 3D scatter plotting
ax.scatter3D(x, y, z, color="red")
ax.set_title("3D Scatter Plot", color="blue")
plt.show()
```

The above code will plot as follow:

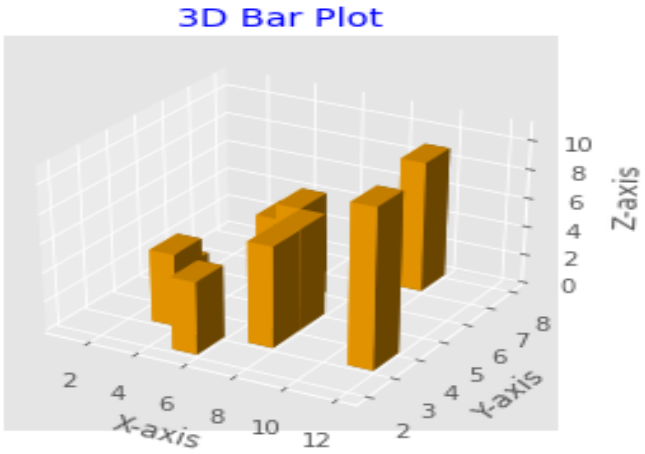


7.6.3 3D Bar Plot

The three-dimensional bar plot is used to compare the relationship between three variables. This plot is plotted using ***ax.bar3d*** function as follow:

```
# Importing library
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import style

# 3D projection
fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")

# All three axis
x = [1,3,5,7,9,11,7,3,5,6]
y = [5,7,2,6,4,6,5,3,6,7]
z = np.zeros(10)

dx = np.ones(10)
dy = np.ones(10)
dz = [1,3,5,7,9,11,7,5,3,7]

# 3D Bar plotting
ax.bar3d(x, y, z, dx, dy, dz, color="orange")
ax.set_title("3D Bar Plot", color="blue")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")
plt.show()
```

The above code will plot as follow:

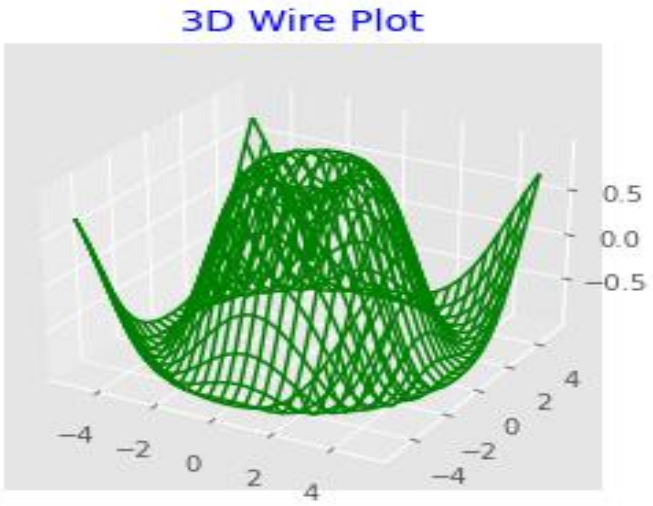


7.6.4 3D Wire Plot

This plot takes a grid of values and draws the lines between nearby points on three-dimensional surface. This plot is plotted using ***ax.plot_wireframe*** method as follow:

```
# Importing library
import numpy as np
import matplotlib.pyplot as plt

# 3D projection
fig = plt.figure()
ax = plt.axes(projection="3d")

# Function
def func(x, y):
    return np.sin(np.sqrt(x * x + y * y))

# All three axis
x = np.linspace(-5, 5, 25)
y = np.linspace(-5, 5, 25)
X, Y = np.meshgrid(x, y)
Z = func(X, Y)

# 3D wireframe plotting
ax.plot_wireframe(X, Y, Z, color="Green")
ax.set_title("3D Wire Plot", color="blue")
plt.show()
```

The above code will plot as follow:

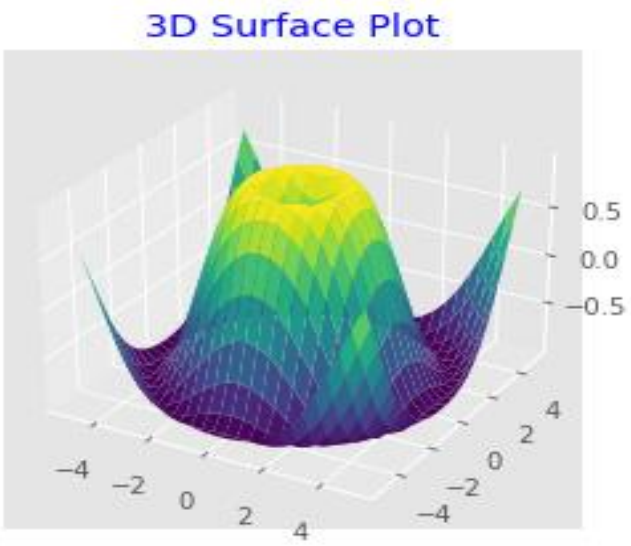


7.6.5 3D Surface Plot

This plot is like as wireframe plot, but each face of the wireframe is filled polygon. This plot shows the functional relationship between one dependent variable and two independent variables. This plot is plotted using ***ax.plot_surface*** method as follow:

```
# Importing library
import numpy as np
import matplotlib.pyplot as plt

# 3D projection
fig = plt.figure()
ax = plt.axes(projection="3d")

# Function
def func(x, y):
    return np.sin(np.sqrt(x * x + y * y))

# All three axis
x = np.linspace(-5, 5, 25)
y = np.linspace(-5, 5, 25)
X, Y = np.meshgrid(x, y)
Z = func(X, Y)

# 3D surface plotting
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,cmap='viridis', edgecolor='none')
ax.set_title("3D Surface Plot", color="blue")
plt.show()
```

The above code will plot as follow:



7.7 SUMMARY

The students will learn many things in this module and they will be able to perform the various data science related operation using Python.

- Ability to do understand the generative and predictive modeling.
- Ability to plot the various types of charts including histogram, scatter plot, line or timeseries plot, bar plot from the dataset.
- Ability to prepare a graph.
- Ability to plot the various types of three-dimension graph.

REFERENCES

Books

11. Davy Cielen, Arno D. B. Meysman, Mohamed Ali: Introducing Data Science, Manning Publications Co.
12. Stephen Klosterman (2019) : Data Science Projects with Python, Packt Publishing
13. Jake VanderPlas (2017) : Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly
14. Wes McKinnery and Pandas Development Team (2021) : pandas : powerful Python data analysis toolkit, Release 1.2.3

Web References

1. https://www.oreilly.com
2. https://www.geeksforgeeks.org
3. https://www.tutorialspoint.com
4. https://www.w3schools.com
5. https://pandas.pydata.org
6. https://pbpython.com

7. https://jakevdp.github.io
8. https://matplotlib.org
9. https://www.logianalytics.com
10. https://seleritysas.com

QUESTIONS

Short Answer:

11. What is generative modeling?
12. What is predictive modeling?
13. List the types of predictive modeling.
14. What is chart?
15. What is histogram?
16. List types of charts.
17. What is graph?
18. List the 3D plots.

Long Answer:

13. Explain predictive modeling in details.
14. Explain histogram with example.
15. Explain scatter plot with example.
16. Explain line chart or time series plot with example.
17. Explain bar plot with example.
18. Explain three-dimensional plot with example.

PRACTICALS

7. Create and display histogram with different arguments.
8. Create and display scatter plot with different arguments.
9. Create and display time series plot with different arguments.
10. Create and display bar plot with different arguments.
11. Create a simple graph.
12. Create and display various 3D plots.

B.Sc.(DATA SCIENCE)

SEMESTER-I

INTRODUCTION TO DATA SCIENCE

UNIT VIII: APPLICATION OF DATA SCIENCE

STRUCTURE

8.0 INTRODUCTION

Today, Data Science has influenced practically all companies by its advanced usage. Each company has used the data for its future goals. Thus, data science has made a bridge between the company and the user. All over the world, there are a couple of industries that use data science such as, business, insurance, energy, health care, biotechnology, telecommunication, travel, etc. At last, there are several data science applications, and these applications are given in this unit.

8.1 OBJECTIVES

The objective of this unit is to illustrate the applications of Data Science. After completing this unit, learners will be able to understand the fundamental applications of the latest technology. This Unit will cover the applications in the following areas:

- Data Science in Clean Energy, Health Care
- Insurance
- Travel
- Transport
- Gaming, Supply Chain Management

This unit deals with the basic applications of Data Science and how these applications are helpful in the implementation of the latest technology.

Figure 1: Applications of Data Science

8.2 DATA SCIENCE IN BUSINESS

In the modern era, various companies are using Data science to ease their regular processing. Most companies use data to make better decisions and that data will be implemented for their growth.

There are many different ways by which Data Science is helping businesses to run in a better way:

1. Making best decisions:

The traditional way of business was more detailed and fixed in nature but that data cannot be used in business operations and decision-making strategies. In this processing, various factors are involved to get the best outcome such as:

1. Aware of the context and behavior of the problem.
2. Elaborate and measuring the quality of data.
3. Best algorithms and tools will be used to find the best solutions.
4. Using stories about success for the understanding of teams.

With the help of these steps, businesses need data science to speed up the decision-making process.

2. **Manufacturing Better Products**

Most companies should be able to engage their consumers towards products. They find the best alternates for their customers and assure them related to their products. Thus, companies need data to develop their product in the best possible way. The process involves such as feedback, use advanced analytical tools, market trends, and innovative ideas.

3. **Efficiently Manage the Business**

Modern business is fully based on data and that will help to make meaningful analysis with events of prediction. With the use of data science, a businessman can manage their business more efficiently either business is on a large scale and a small startup. Moreover, companies can predict the growth rate of their implementation. Thus, it can help in summarizing the overall performance of the company and the distribution of the products worldwide. Data science helps in managing business efficiency by some factors such as tracking the performance, the success rate of the product, and other important metrics.

4. Predict Outcomes for Future Outcome

Prediction factors are the most crucial part of businesses. With the advanced tools and technologies, industries have elaborated their capability to provide the best training to their employee. In simple terms, predictive analytics is that which involves several machine learning algorithms and tools for future outcomes using previous year's data and these tools are SAS,

IBM, SPSS, etc.

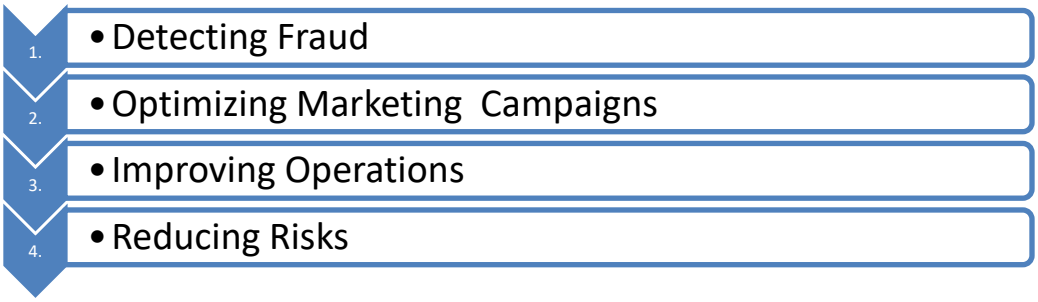


Figure 2: Predictive Processes

5. Assessing Business Decisions

After prediction, it is important to understand how these predictions will affect the performance of the business and growth. If the decision leads to face any consequences, then analyze the decision and eliminate the problems. Thus, business can make profits with the help of data science.

8.3 DATA SCIENCE IN CLEAN ENERGY

Clean energy is also known as renewable energy. It continuously grows with the advancement of the Internet of Things (IoT). The connectivity and sensor advancement collect more innumerable data from outside resources. Earlier energy companies did not work with a large amount of data. But, with the approach of data science, one can derive important insight from the excessive data. Data science to be used in the industry of clean energy to improve and optimizes the processes.

One of the significant approaches is to improve the day-to-day operations in the energy renewable sector. Data plays an efficient role in the management and clean energy regulation. Data science is to be used in many ways in clean energy. One of the principal examples is the solar plant that collects data for optimizing power performance, also predicts the maintenance required after the particular interval, etc. These applications include the widespread collection of data and its analysis.

There exist several ways of using clean energy with data science. For example, the data to be collected from solar to optimize the performance, reduce the maintenance times, analyze the expected maintenance time, and make solar implementation more compact. These applications of clean energy are extensively used for collection and data analysis.

8.3.1. Clean Energy for Most Suitable Environment and Economy

It can be a great asset to the enterprise that makes optimizations in the day-to-day enforcements in the wind or solar farm. In this way, renewable energy is an environmentally friendly and cost-effective option to fossil fuels that improve efficiency. With the advancement of renewable

energy sources, it is possible to get more mileage from wind farms or solar plants in a desirable manner.

8.4 DATA SCIENCE IN HEALTH CARE

Data science plays a vital role in all industries and it will help to transform the health care sector also. Medicine and healthcare facilities both are a crucial part of everyone's life and the treatment and the medicine are fully relied on doctor prescription but this wasn't always right. On the other hand, data science will help them with their diagnoses. In the healthcare sector, several fields such as medical imaging, drug discovery, genetics, predictive diagnosis, and some other areas where data science has been used.

a) **Medical Imaging**: Technology has improved the health sector day by day and there are various imaging techniques to visualize the internal parts of the human body such as X-Ray, MRI, and CT-Scan. In the past few years, doctors treat patients manually and this technique creates a problem for both of them, however, it has been possible through deep learning technologies in data science. In the past few years ago, doctors check the patients manually but due to deep learning of data science, the trend has to change day by day.

b) **Genomics**: It is the study of sequencing and analysis of genomes. A genome is made of DNA. It is a time-consuming and expensive process to analyze the sequence of genomes. However, with the advanced tools of data science, it is possible to analyze and understanding human genes at a low cost. The Gene system has helped to find the connections between genetics and the health of the person. Moreover, it also involves finding the proper drug and how it will affect the particular genetic structure and this combination is called Bioinformatics. Several tools are related to data science such as MapReduce, SQL, Galaxy, etc.

c) **Drug discovery**: Drug Discovery is a highly complicated task. Thus, the pharmaceutical industries are based on data science to solve their problems and create the best outcome for humanity. For example: In Covid -19 researcher of various countries share their data so they produce the best vaccine for the social cause. Drug discovery is a time-consuming process that involves heavy finance and heavy testing. Companies use the patient's information such as mutation profiles from the previous year's data and they can design drugs that address the key mutations in the genetic sequences. Furthermore, researchers study chemical compounds and test the combinations of different cells and genetic mutations. After that, they will produce the drug to solve the health issues.

d) **Monitoring patient health**: Data science plays a major role in IoT devices and these devices assist to check the heartbeat, temperature and check other medical parameters of the person. The data is collected with the help of analytical tools and it helps the doctors track the patient biological rhythm. Furthermore, it helps the doctors to take compulsory decisions for their patients.

e) **Tracking & Preventing Diseases**: Data science plays a significant role in monitoring

patient's health and takes important steps to detect chronic diseases at an early level with the best use of Artificial Intelligence.

8.5 INSURANCE

Insurance is one of the most competitive and unpredictable industry and it has been dependent on statistics however data science has changed the scenario. These days, insurance companies have a large range of information sources for covering the risk assessment. Big data helps to find risks and maintain effective strategies for customers.

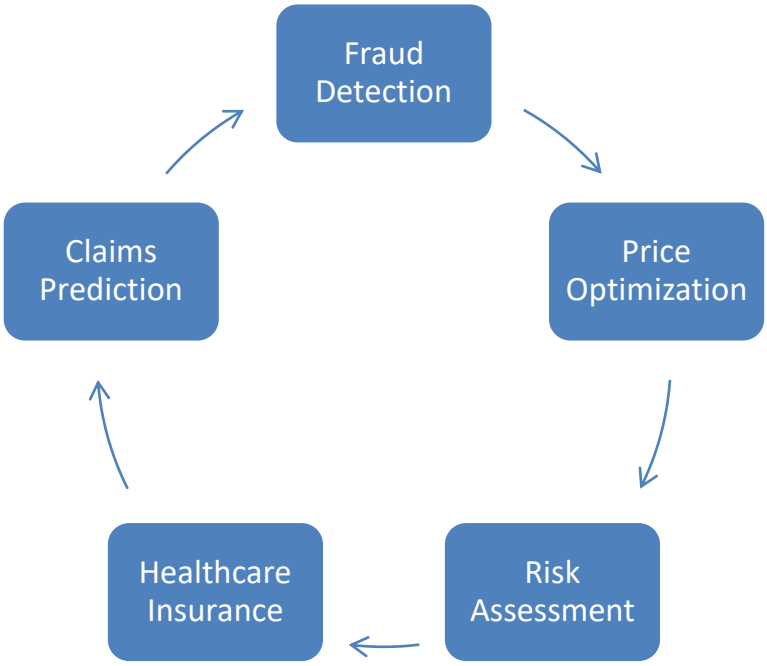


Figure 3: Effective strategies for customer

a) **Fraud detection:** Insurance fraud is faced by insurance companies every year. Data science is the best way to detect fraudulent activity with the help of software and various techniques. This software relies on the previous history of such activities and uses the sampling method to consider solving this issue.

b) **Price optimization:** It is a complex approach thus it can handle combinations of various methods and algorithms. Insurance companies use these algorithms to compare the changes between previous years and customer policy, because price optimization is closely related to the customer's price awareness. Price optimization assists to improve the customer's bond with the company for a long period.

c) **Risk assessment:** Risk assessment tools in the insurance industry encourage the forecasting of risks. Risks are of two major types: pure and speculative. A risk assessment identifies the risk amount and various reasons for the risk and these are the basis for data analysis and

calculations.

d) **Healthcare insurance:** Health insurance is the best insurance throughout the world and it covers the costs caused by any major diseases, accidents, disability, or death. In most countries, health policies are strongly recommended by governments. Healthcare facilities are improving rapidly due to which companies face to provide better services and reduce the customer's costs. As a result, companies store a wide range of data includes claims data, membership information, medical records, etc. so, companies will provide quality of care, fraud detection, and prevention and consumers get better facilities from the companies.

e) **Claims prediction:** Insurance companies are interested in the prediction of the coming future, so these companies predict the financial loss earlier by the use of models such as a decision tree, a random forest, a binary logistic regression, and a support vector machine. Forecasting, the demand of the future to charge premiums that are not too high and not low. Price models also contribute to the improvement, which helps the company to be one step beyond its opponents.

8.6 TRAVEL

Data Science is growing with multiple paradigms, tools, and machine learning approaches to acquire predictive, descriptive, and perspective goals to derive hidden data from the raw facts.
Nowadays, traveling is a thriving industry that regularly increases the number of consumers and requires the highly processing of data. This approach is highly adapting the data science algorithms. Travel industries such as hotel, reservation, airline, booking sites, etc. are the broad areas that manage multiple requests from customers every day. Data Science plays an efficient role and is used for hospitality and marketable uses in travel.

a) **Viewpoint Analysis:** The area where data science has proved valuable is viewpoint analysis. This analysis depends on the reviews of customers on the website. Organizations also track the remarks and emojis that are used by the customers for brands. It is critical to understand the view of the customer towards the brand and how they recommend the services to others. In this, Data science provides the resolution to do so.

b) **Tour Planning:** Data Science helps in building agendas and planning that assist in time management, cost-cutting, and quick decision making. The new trends in AI are trained by data science that requires constant learning about customer requests and requirements. It helps the customers by spending less time and numerous cost-effective plans. Every customer's history is considered by the AI machine learning tools to discover the most suitable plans.

8.7 TRANSPORT

Data science is also used in transportation actively making its mark in construct safer ways for

drivers. In this sector, data science introduced a self-driving car for the new generation. It also analysis of fuel consumption patterns, driver behavior and actively monitor the vehicle has established a strong bond between the company and the user. Driverless cars are the most advanced topic currently available in the market. Same as the driverless metro even cycle also has invented by Google engineers. Moreover, industries can also create the best logical routes and this is possible through data science. For example, most of the transportation companies follow data science for providing a better experience to their customers so that customers give the best feedback for their company and they also predict the best way for the customers without inconveniences like traffic and road conjunction.

8.8 MANUFACTURING

Manufacturing is the pillar of every other industry because it helps to analyze the performance, reduction of faults in the production adapts to the changes in the market trends, and upgrades the production system by the use of new technologies. The trend should be changed with the help of data science in manufacturing companies to boost production and generate annual revenue. To illustrate: the car industry is the real cause of a manufacturing company. In the last year, 2020 major food companies are using AI machines and this technology is capable of performing the given tasks such as:

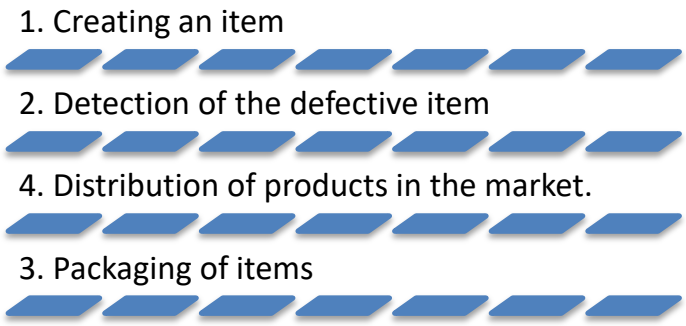


Figure 3: Steps

Product efficiency and performance of the machines are calculated by data science techniques such as visualization, which investigates the number of products per day as well as the number of defective product items and reasons for the defects. Data science helps to predict the annual growth on the behalf of current production systems.

Applications of Data Science in Manufacturing

The major application of data science in manufacturing industries is given below:

- Price Optimization
- Predictive Analytics
- Demand Forecasting and Inventory Management
- Supply Chain Management

a. **Price Optimization:** The main factor of Competition between the companies is the price of the product and numerous factors on which the end price of a product depends. These factors are raw materials, machines, labor cost, and electricity. All these affect the price of the final product. If the price of the product is high then it will affect the demand for the product in the market. At this stage, techniques of data science help to understand the requirement of products and identify the unnecessary costs of the product. It helps to optimize the cost of the product that would be responsible for their customers as well as, this cost helps them to in the highly corporative world with affordable cost. By this, the companies can groom their profitability for their future betterment.

b. **Predictive Analysis:** The overall growth of the company depends on its command of the trend of the market, the need of customers, and business rivals. Predictive analysis is one of the major factors that can help the organization's goals as per the customers' needs. Data science predicts the growth, customer demands, and future requirements due to these predictions', organizations set their goals for future manufacturing. During this process, data science can protect from errors in the products and develop more secure technologies that can help to increase production. Data science tools help to figure out the business and make changes in the product according to requirements. Production staff can build planning to cover uncertain situations using predictive analysis tools.
c. **Demand Forecasting and Inventory Management:** One major key factor for successful manufacturing is on-time production. Manufacturing companies face high-priority tasks in packaging and supplies. In a competitive market, it is important to forecast the demand of the customers in advance. Moreover, data science can help to analyze and predict customer demands. It helps the companies to manage the production and supply chain. In addition, it prevents unneeded production. At last, this feature is to manage inventory management, customer requirements, and business needs.

Some key factors about demand forecasting and inventory management:

1. It helps to overcome the need for irrelevant storage.
2. It controls inventory management.
3. It covers the gap between the suppliers and the production companies.
4. It manages the supply process.

In this way, production companies can perform well in the corporate world and create new strategies for future development.

8.9 TELECOMMUNICATION

Telecommunication companies are using data science applications that help in profit maximization. These companies play an efficient role in building an effective marketing strategy

and business strategy. It analyzes and visualizes data that performs transferring and many other tasks. The basic functionality of these companies is strongly related to transfer, import, and exchange. The older techniques are not considered these days, because the amount of data that passes through the communications links are growing larger with every passing minute. The most relevant and efficient use cases of data science in the field of telecommunication.

a. **Fraud Detection:** The telecommunication industry is affected by frauds, illegal access, theft, fake policies, authorization, cloning, etc. Frauds directly influence the relationship between the user and the company. Thus, the approaches of fraud detection, tools, and techniques are widely available. By using machine learning algorithms, the operator can identify and categorize normal traffic and fraud. If it identifies any fraud, it generates an alert and informs the system administrator. This technique is efficient because it provides a real-time response to anomalous activities.

b. **Predictive analytics:** It is applied in telecommunication companies for predicting valuable information faster, better and helps in making decisions. This approach uses historical data to build predictions. The better the quality and for a long time in history, the better will be predictability.

c. **Consumer Segmentation:** The success of telecommunication companies is to create market segments and content can be divided according to each group. There exist four segmentation schemes such as customer value segmentation, customer lifecycle, behavior, migration segmentation.

8.10 SUPPLY CHAIN MANAGEMENT

The supply chain process involved in manufacturing has always been complicated. From start to end risk has been involved in all stages. Some complicated stages are a collection of requirements, gathering raw data, market data, analysis of the various resources, well-trained staff and machines, quality and distribution of the product in the market. While production, some other factors are involved during company running time. Supply chain impact on the business loss due to organization faces several problems. Data science helps to manage the supply chain, detects the events where overloading is happening, predict the future possibilities of delay in manufacturing and supply. By this, manufacturers store the data in the form of backup for future emergencies and it will help all the businesses.

Finally, the applications of Data Science in manufacturing will change the current scenario of older methods used by the production companies. It will enhance the revenue generation of companies and contribute to the overall economic growth of industrialization.

8.11 GAMING

The gaming industry is one of the greatest industries in modern days. These days approximately two billion users all over the world have been involved in this industry due to more users data to

processed is enhancing day by day. Data science helps to improve the business of gaming with an understanding of the data.

a. **Game development**: Most important task in gaming is developing the game. In this step, the developer considers the idea behind the game, its functionality, design plays an analytical part of gaming and interesting features in-game playing. In which performance should be measured, result, and it may be adapted according to the need of the customer. In which data science is used to develop models, to analyze and identify expansion points, make predictions, algorithms, pattern identification, maps in which players should move.

b. **Game monetization**: Another main factor of game planning is linking with companies' growth. Well, design games will help the developer to make money, thus the developer will concentrate on the growth of the company. Big data is used to predict the behavior so that users will come to play the same game again and again and will be ready to pay money for playing.

c. **Game design:** Game designing is an art and it will be the best career for developers. It is a complex task requiring several programming languages. Developers show their creativity to help to create interactive and complex sides for the games. Gaming analytics are used to obtain about user's requirements, to understand the various barriers, reasoning, and timing. Advanced gaming concepts, storylines, and create a strategy to find the previous data.

d. **Object identification:** Latest graphics features, artificial intelligence applications are the key factors of developers and creative designers. Image recognition technologies are remodeling the gaming industry. Developer's uses object detection models, so they create the natural scenes with the perfect movement of the objects. For example, these models are used to differentiate between teams, commands, obstacles, and figures become easier and much faster for interactive games.

e. **Visual effects and graphics:** During video game development, there were various graphic techniques invented. Due to these advanced techniques, gaming effects have to change such as motion capture in games, real-time rendering, and many others. While using these motion techniques allows the creation of characters with more human features and attributes. It helps in display facial emotions and expressions with natural movements. Animation's developers try to use the latest algorithms to push the video boundaries of the game reached up to one more advanced level. Real-time tools are used for this purpose.

f. **Personalized marketing:** Personalized marketing is applied by those companies which are linked with animations, video, and gaming. Companies avoid useless and unprofitable advertisements. In which marketers and game developers are interacted with customers

and lead by the creation of successful market messages and results will be provided to the exact people. Personalized marketing is helping to find the activities of the users and side by side attract new users.

g. **Fraud detection:** In Gaming, all the actions and recommendations in the world are fast. Companies face the problem of finding fraudulent activity, so it is a challenging task. To solve this problem, companies provide verifications of users by law, finding doubtful accounts, and stop these fraud accounts as soon as possible. Payment fraud is common in gaming due to which companies require a high level of security system for detection of such users. Machine learning algorithms come to the security of gaming industries by applying fraud detection methods.

h. **Social and customer analysis:** The gaming industry proves to be very innovative and successful in the modern era. Its growing popularity is depending upon, after a new minute a new user is attracting towards games. Millions of users take part in video games actively all over the world. Most youngsters use gaming as a platform and leave a significant amount of valuable data that are used by developers because data helps them to understand the customer's perception and develop the latest games with advanced features.

At last, data science is helping to improve in the gaming industry by using techniques and methods. Data science helps the developers to invent the latest games and companies gain profit from such games and users enjoy playing real games and reduced their stress level.

8.12 GOVERNANCE

Governance is the application of data science that consists of rules, policies, processes, and organizational structure to support the data management task of the enterprise. The hierarchy of the organization data provides guidance, understanding, security, and trust among its stakeholders. With the growth of companies' data, the organization needs to create an appropriate big data environment that makes them available across the organization. This integration is an essential part of deciding workflows and making decisions by various teams. Data governance is an essential task of the organization's data management. With the help of this governance, one can easily know the kind of data availability, where the data resides, and the method of data usage.

8.13 BIOTECHNOLOGY

Biotechnology is a term that represents the manipulation of crops, breed the animals to produce specific features or simply biotechnology modifies the products for specific use. In the modern era, our technological tools are mathematics, statistics, computational resources, availability of data sources, etc. In data science, biotechnology plays a vital role to improve medical treatment, agriculture, animal breeding, and industrialization and to solve in environmental issues.

Types of biotechnology

The basic term of biotechnology can be broken down into small chunks bases on common use and various applications.

1. **Red Biotechnology:** It is the latest technology that will use in medical processes such as getting an organism to produce new drugs and stem cells to revive damaged human tissues or regenerate entire organs.
2. **White Biotechnology:** It has to be used in industry such as the invention of new chemicals or fuels for vehicles.
3. **Green Biotechnology:** It is for living creatures means green biotechnology implement in agricultural processes such as removing pests, so everybody uses pest's free crops and creates environmentally friendly development.
4. **Gold Biotechnology:** It is also known as Bioinformatics and it will use in biological data.
5. **Blue Biotechnology:** It will help in defense such as encompasses processes in marine and aquatic environments.
6. **Yellow Biotechnology:** It is the oldest branch of biotechnology and is used to process more nutrition-rich food products.

Yellow Biotechnology can be said as the oldest branch of Biotechnology. It uses microorganisms or insects for the production of more nutrition-rich food products for us so Yellow Biotechnology is also called Nutritional Biotechnology or Insect Biotechnology.

Biotechnology and Data Science

Biotechnologists are researchers and they use statistical analyses to redesign based on experiments. Data scientists focus on math and stats background however biotechnologists work on biostatistics and they have to used quantitative data and shifting through which factors are more likely to produce a particular effect requires major computational effort.

8.14 PHARMACEUTICALS

The launching of new pharmaceutical products to market is a long process with many consequences. A thousand trials need to be taken regularly to meet the objective that delays getting the output and also increases the cost with this expensive process. There exist numerous data points, experiments, benefits, and risks that must be analyzed which help in making the industry logically fit for the big data.

The cost of clinical trials can also be reduced by the data scientist:

- **Patient selection based on data:** Multiple data sources including social media and public databases related to health that identifies the population best for the trials.
- **Monitoring:** The companies can monitor on a real-time basis to identify operational risk.

- **Safety Assurance:** The data scientist can also consider the side effects before actual trials.

8.15 GEOSPATIAL ANALYTICS AND MODELLING

Geospatial analytics helps in gathering, manipulating, and displaying geographic data that includes GPS and satellite information. This data relies on coordinates and specific terms like a street address, postal address, etc. These techniques help in creating geographic models and visualization patterns for accurate modeling and predictions. The data can be collected from different technologies such as GPS, social media, location sensors, mobile, satellite, etc. to build better visualization of data for understanding the complex relationships between places and people. The visualization includes the maps, graphs, cartograms, statistics, etc. that shows the history and current changes also. It helps in making predictions more accurate and easier.

Geospatial analytics also adds timing and location for the data that creates a complete picture of events. Geospatial analytics process a large amount of geographic data. It helps the users to interact with billions of mapped locations by looking at real-time geospatial visualizations. Users can traverse the data using time and space by instantly checking changes from days to years.

The benefits of geospatial analytics include:

- **Appealing insights:** The data displayed in the form of maps are easier to understand by any user.
- **Better vision:** With the help of visual images, one can understand how the spatial conditions are changing in real-time that helps an organization to understand the changes and determine actions for the future.
- **Targeted report:** The location-based data helps the organizations to understand, why some locations and countries, like the United States, are more prosperous for business than others.

Geospatial Analytics Use Cases

Telecommunications: It quickly visualizes the record related to call details and network records to fix the issues before the customer notices. This tool helps in the identification of anomalies for signal fluctuation and assists how to resolve them.

Military: In a military operation, logistics provides an aspect of situational awareness. This tool helps the military to optimize the placement of resources by predicting the infrastructure usage, and maintenance needs.

Weather: It helps in getting a quick response to weather by getting alerts. This data also helps the airlines with routing and insurance companies to better access the property risk.

Urban Planning/Development: It helps in determining the growing population's effect on energy, transportation, and other resources. This data helps to analyze the large datasets with high speed and scale.

8.16 SUMMARY

This unit covers the important and advanced applications of data science. Nowadays, data science plays an efficient role to achieve a company's goal. It is used by a couple of industries all over the world that uses data science such as business, health care, insurance, telecommunication, biotechnology, etc. To gather data for manipulation and to display geographic data such as GPS and satellite information, the Geospatial analytics application of data science is used. It also helps in making business related decisions in an easy way and quickly. Thus, data science includes a lot of different applications to achieve the industry goal and makes improvements in existing work.

ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਪੰਜਾਬ ਸਟੇਟ ਓਪਨ ਯੂਨੀਵਰਸਿਟੀ
ਪਟਿਆਲਾ

The Motto of Our University
(SEWA)
SKILL ENHANCEMENT
EMPLOYABILITY
WISDOM
ACCESSIBILITY

SELF-INSTRUCTIONAL STUDY MATERIAL FOR JGND PSOU

JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

B.Sc.(Data Science)

Semester I

BSDB31102T

Fundamental of IT

Head Quarter: C/28, The Lower Mall, Patiala-147001
Website: www.psou.ac.in

The Study Material has been prepared exclusively under the guidance of Jagat Guru Nanak Dev Punjab State Open University, Patiala, as per the syllabi prepared by Committee of experts and approved by the Academic Council.

COURSE COORDINATOR AND EDITOR:

Dr. Amitoj Singh
Associate Professor
School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University

LIST OF CONSULTANTS/ CONTRIBUTORS

Sr. No.	Name
1	Dr. Amitoj Singh
2	Dr. Anju Bala

JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA
(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

PREFACE

Jagat Guru Nanak Dev Punjab State Open University, Patiala was established in December 2019 by Act 19 of the Legislature of State of Punjab. It is the first and only Open University of the State, entrusted with the responsibility of making higher education accessible to all, especially to those sections of society who do not have the means, time or opportunity to pursue regular education.

In keeping with the nature of an Open University, this University provides a flexible education system to suit every need. The time given to complete a programme is double the duration of a regular mode programme. Well-designed study material has been prepared in consultation with experts in their respective fields.

The University offers programmes which have been designed to provide relevant, skill-based and employability-enhancing education. The study material provided in this booklet is self-instructional, with self-assessment exercises, and recommendations for further readings. The syllabus has been divided in sections, and provided as units for simplification.

The University has a network of 10 Learner Support Centres/Study Centres, to enable students to make use of reading facilities, and for curriculum-based counselling and practicals. We, at the University, welcome you to be a part of this instituition of knowledge.

Prof. Anita Gill
Dean Academic Affairs

B.Sc. (Data Science)
Core Course (CC)
Semester I
BSDB31102T: Fundamental of IT

Total Marks: 100
External Marks: 70
Internal Marks: 30
Credits: 4
Pass Percentage: 35%

Objective

This course introduces the concepts of computer basics and working with word processor, spread sheet, and presentation software packages. Basic concepts of information technology have also been explained in this course.

INSTRUCTIONS FOR THE PAPER SETTER/EXAMINER

1. The syllabus prescribed should be strictly adhered to.
2. The question paper will consist of three sections: A, B, and C. Sections A and B will have four questions from the respective sections of the syllabus and will carry 10 marks each. The candidates will attempt two questions from each section.
3. Section C will have fifteen short answer questions covering the entire syllabus. Each question will carry 3 marks. Candidates will attempt any ten questions from this section.
4. The examiner shall give a clear instruction to the candidates to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.
5. The duration of each paper will be three hours.

INSTRUCTIONS FOR THE CANDIDATES

Candidates are required to attempt any two questions each from the sections A and B of the question paper and any ten short questions from Section C. They have to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.

Section A

Unit I: Computer Fundamentals: Block diagram of a Computer, Characteristics of Computers, Hardware, Software, Machine Language, Assembly Language and Assembler, High Level Language and Compiler v/s Interpreter. Input Devices: Keyboard, Mouse, Joystick, Track Ball, Touch Screen, Light Pen, Digitizer, Scanners, Speech Recognition Devices, Optical Recognition devices – OMR, OBR, OCR. Output Devices: Monitors, Impact Printers - Dot matrix, Character and Line printer, Non Impact Printers – DeskJet and Laser printing, Plotter.

Unit II: Computer Memory: Representation of information: BIT, BYTE, Memory, Memory size; Units of measurement of storage; Main memory: main memory organization, RAM, ROM, PROM, EPROM, Computer languages: Machine language, assembly language, higher level language, 4GL. Introduction to Compiler, Interpreter, Assembler, System Software, Application Software. Introduction to Internet, WWW and Web Browsers

Unit III: **Basic of Computer networks;** LAN, WAN; Concept of Internet; Applications of Internet; connecting to internet; ISP; Knowing the Internet; Web Browsing software's, Search Engines; Understanding URL; Domain name; IP Address; Using e-governance website Basics of electronic mail; Getting an email account; Sending and receiving emails; Accessing sent emails; Using Emails;

Unit IV: Word Processing Package: Opening, saving and closing an existing document; renaming and deleting files; Using styles and templates: Introduction to templates and styles; applying, modifying; using a template to create a document, creating a template, editing a template, organizing templates, examples of style use, Changing document views

Section B

Unit V: Working with text: select, cut, copy, paste, find and replace, inserting special characters, setting tab stops and indents, Formatting text, formatting paragraphs, Formatting pages: Using layout methods, creating headers and footers, Numbering pages, Changing page margins, Adding comments to a document, Creating a table of contents, Creating indexes and bibliographies, Printing a document, Tracking changes to a document.

Unit VI: Making Small Presentation: Basics of presentation software; Creating Presentation: Entering and Editing Text, Inserting And Deleting Slides in a Presentation, Inserting Word Table or An spreadsheet Worksheet, Adding Clip Art Pictures, Inserting Other Objects, Slide Show: Running a Slide Show, Transition and Slide Timings, Automating a Slide Show,

Unit VII: Using Spreadsheet Statistical functions: SUM, COUNT, AVERAGE, MAX, MIN, MEDIAN, MODE PRODUCT SQRT, STDEV.S, ABS, QUARTILE, PERCENTILE, AVERAGEIF, COUNTA, COUNTBLANK, CORREL, Logical operation IF, SUMIF, AVERAGEIF, COUNTIF,

Unit VIII: Formatting Text: Using RIGHT, LEFT, and MID functions; format text by using UPPER, LOWER, and PROPER functions; format text by using the CONCATENATE function, Generating inference from Data: Pivot Table, Creating Charts, Data Cleaning: Removing duplicate values, Text to Columns,

Suggested Readings

1. Nortorn, P. Introduction to Computers, 7th Edition, 2017
2 .Rajaraman, V., Fundamentals of Computers, PHI, 2014
3. Larry E. Long and Nancy Long, Computers: Information Technology in Perspective, PHI, 2001
4. Andy Channelle, Beginning OpenOffice 3, Apress, 2009

JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA
(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

BSDB31102T: FUNDAMENTAL OF IT

COURSE COORDINATOR AND EDITOR: DR. AMITOJ SINGH

UNIT NO.	UNIT NAME
UNIT 1	COMPUTER FUNDAMENTALS
UNIT 2	COMPUTER MEMORY
UNIT 3	BASIC OF COMPUTER NETWORKS
UNIT 4	WORD PROCESSING PACKAGE
UNIT 5	WORKING WITH TEXT
UNIT 6	MAKING SMALL PRESENTATION
UNIT 7	USING SPREADSHEET STATISTICAL FUNCTIONS
UNIT 8	FORMATTING TEXT

`

B.Sc.(DATA SCIENCE)

SEMESTER-I

FUNDAMENTAL OF IT

UNIT I: COMPUTER FUNDAMENTALS

STRUCTURE

1.0 OBJECTIVES

- To understand computer organization and bock diagram of computer.
- To gain knowledge about input, output and other peripheral devices.

1.1 COMPUTER SYSTEM

Today, computers are not only used in commerce and business but also in various fields like medicine, research, educational institutions, launching a satellite, etc. computers are available in different sizes and have different capabilities of processing commonly known as configurations. We need different types of hardware to construct a computer. Computer hardware is a collective term used to explain various tangible components of computers. Each hardware component has its functionality and is attached in a specific manner to form a computer system. A typical block diagram of a computer system is shown in Fig below. Now we will discuss the basic components that make up a computer system.

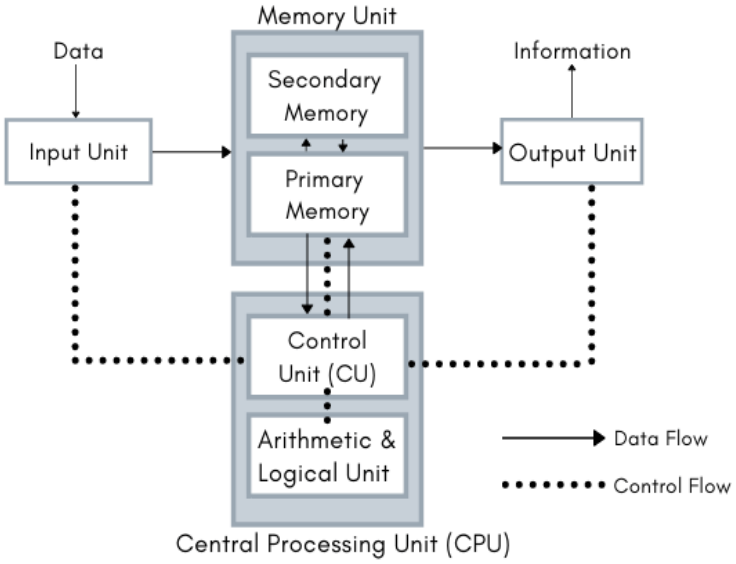


Fig 1.1 CPU

1.2 COMPUTER SYSTEM BLOCK DIAGRAM

1.2.1 Central Processing Unit (CPU)

The CPU consists of arithmetic and logical unit (ALU), control unit (CU) and memory storage unit. Collectively they form the brain of the computer. CPU is the place where the actual processing of data takes place on the execution of the program. The CPU is responsible for processing most of the data, turning input data into output data. The CPU is one of the main components that will improve the performance of your computer. CPU is being used in almost all kinds of digital processing equipment like desktops, laptops, tablet computers, smartphones, even in our television set and many more devices. Colloquially it is also termed as a processor, microprocessor or central processor. The two main companies that manufacture the desktop CPU are AMD and Intel.

1.2.2 Microprocessor

It is a silicon chip with ALU, register circuits and control circuits. The microprocessor is capable of carrying out a large number of functions like receiving data, processing and storing the results and outputting the required results on a single integrated circuit. It has the responsibility to perform ALU operations and control the components connected to it like memory, input output devices, etc. Thus, it is a programmable device that takes binary data as input, performs processing as per instructions loaded in memory and generates results in binary form.

1.2.3 Arithmetic and Logical Unit (ALU)

An ALU *is* a major component of the CPU of a computer system. It performs all the arithmetic & logical operations for the computer system e.g., addition, subtraction, compare, complement, shift, etc. It is a combinational digital circuit and an ALU can be designed by engineers to calculate any operation. As the operations become more complex, the ALU also becomes more expensive and it will take up more space in the

1.2.4 Control Unit (CU)

This is an important part of the CPU which supervises all the operations taking place in it. Its main aim to send and receive control signals to all parts of a computer system. The control signals are helpful in the smooth execution of instructions in the CPU, communication over buses to memory and IO devices. Via control signals, CU facilitates that all tasks in computer performed at right time and in the correct order. It also directs other units of the system to carry out their respective function. Thus, CU regulates and integrates the operations of the computer. It fetches an instruction from a program stored at main memory, decodes it and sends control signals to other units of the computer system.

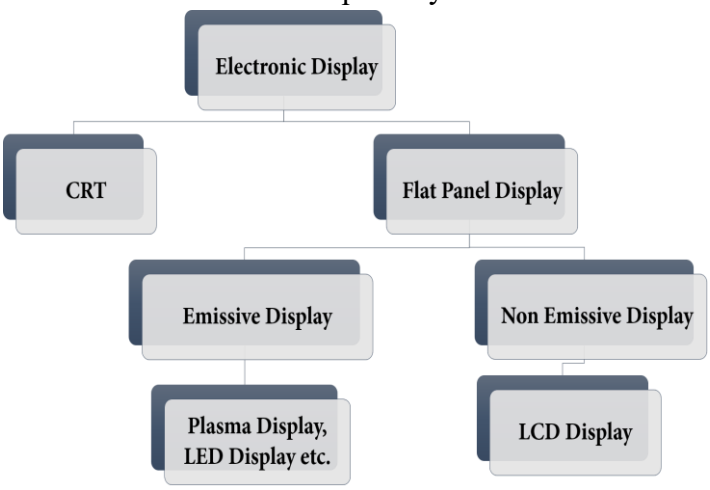


Fig. 1.2: Types of Display

1.3 DISPLAY

A display is an output device to present information in visual form. It may be an external monitor or built-in screen with the digital device e.g., computer, mobile, ATM, advertising boards, etc. A computer display is simply the screen that will give you, your video output from the computer. A computer monitor works with the video card located inside the

computer case, to display images and text on a screen. Most monitors have some control buttons that allow us to change the display settings of the monitor. With the change in display screen technology, different types of monitors are available in the market.

CRT (cathode ray tube) Monitors: These are older computer monitors built using cathode ray tubes (CRTs). A typical CRT monitor is shown in Fig below. The monitors employ CRT technology, which was used most commonly in the manufacturing of television screens. A cathode ray tube is a vacuum tube containing an electron gun at one end and a fluorescent screen at another end. The use of CRT made them heavy and caused them to take up a lot of desk space.

CRT Monitor **LED Monitor** **DLP Projector**

Fig. 1.3.: Display Devices **(a) CRT Monitor** (b) LED Monitor (c) DLP Projector

LCD (liquid crystal display) Monitors: Most modern monitors are built using LCD technology and are commonly referred to as flat screen displays. These thin monitors are light weighted, electricity saviour and take up much less space than the older CRT displays. An LCD is composed of two pieces of glasses with a thin layer of liquid crystals in between. When a voltage is applied to the glasses, the orientation of liquid crystals will be changed. This change in the crystal's orientation (called polarization) will make either a dark or a light area, creating a character or image on the display.
A TFT monitor uses thin-film transistor technology in an LCD. It is a variant of LCD monitors and is dominantly being used in current monitors.

LED (light-emitting diodes) Monitors: LED monitors are the latest types of monitors in the market today. These are flat panel displays that make use of light-emitting diodes for back-lighting, instead of cold cathode fluorescent (CCFL) backlighting used in LCDs. The advantages of LED monitors are that they produce images with higher contrast, have a less negative environmental impact when disposed of. Modern electronic devices such as mobile phones, TVs, tablets, computer monitors, laptops screens, etc., use a LED display to display their output.

DLP Monitors: DLP stands for Digital Light Processing, developed by Texas Instruments. It is a technology, which is used for presentations by projecting images from a monitor onto a big screen. It gives better quality pictures that can also be visible in a lit room normally.
Plasma Monitors: The plasma monitor has a flat screen, and it has small fluorescent lights with colour that are lit up to form images on the screen. plasma monitors have a very wide screen using very thin materials.

OLED Monitors: OLED stands for organic light-emitting diode. This type of monitor is thinner and lighter, and it offers incredible contrast and colour. It works without a backlight as it transmits visible light. Flexible and transparent displays are also possible using OLED.
Touch Screen Monitors: These monitors perform both input and output functions. It enables users to interact with the computer by using a finger or stylus instead of using a mouse or keyboard. When users touch the screen with their finger, it occurs an event and forwards it to the controller for processing. It takes input from the users by touching menus or icons presented on the screen.

1.4 KEYBOARDS

A keyboard is the primary input device used with the computer similar to an electronic typewriter. It is used to input data and instructions of a user in a computer system. A keyboard is composed of buttons used to create letters, numbers, symbols, and perform functions. A keyboard is connected to a computer system using a cable or wireless connection. Some keyboards also have additional functions like volume control buttons to power down or sleep the device even a built-in trackball mouse intended to provide the easiest way to use both the keyboard and the mouse. The various types of keyboards are used by users for associated purposes like a qwerty keyboard for general purpose uses, a gaming keyboard for game lovers, a virtual keyboard for software inputting, an ergonomic keyboard for physiological consideration and a multimedia keyboard for convenient web surfing and music play, etc. A keyboard has various keys some are logically grouped and a name is assigned to them. A QWERTY layout keyboard and its various keys are depicted in Fig. below.

Fig. 1.4: Keys on keyboard

Function keys: The topmost row of the keyboard have function keys. These are twelve keys F1 to F12. Each of these keys is used for a special purpose and assigns some system commands to them. For different software, these can be customized to perform a specific task. e.g., some shortcut of command can be assigned.

Character keys: These are keys which also present in a traditional typewriter too e.g. A-Z, a-z, 0-9, Tabs, Caps, are those keys. The keys are used to type letters, punctuation and other characters.

Modifier keys: These keys do nothing by themselves but with help of these keys, the functions of other keys are modified. Ctrl, Alt, Shift, Alt Gr comes under this group.

Navigation keys: These are also termed cursor control keys. Used to navigate cursor in any direction e.g., left □, right➡, up⬆and down⬇, beginning of line or screen (Home), end of line or screen (End).

Numeric Keypad: Numeric keys of keypad i.e., 0-9, NumLock, -, +, /, * and Del keys forms this group.

System command keys: some keys have important functions other than printing characters, depending on the type of application where they are being used. These can be interpreted by the computer system as formatting or important commands to the system.

PrtSc: Print Screen key is used to capture the entire screen and send it to the clipboard.

Break/Pause: Not being used for predefined purpose nowadays. We can still use it to assign other tasks like terminate a program.

Esc: Escape key is used to quit a dialogue box, as a quit or stop signal.

Enter: In a text editor window, it is used to terminate a paragraph and request for the next newline. For a command line, enter key is a signal to process the command.

Shift: The shift key is used to type more symbols than visible on the keyboard. e.g when we press the 'a' key +⇧ Shift key it will produce 'A'.

Window: Window key (□) is used to open the start application menu on the windows operating system.

Space Bar: It is used to provide space between words during typing. In physical appearance, it is a wide key on the keyboard.

Backspace: The backspace key erases the text to the left of the cursor's position. It is generally useful for correcting typos.

1.5 MOUSE

A mouse is a handheld input device that controls the pointer in a GUI (Graphical User Interface). It is the most widely used pointing device and can move and select text, icons, files, and folders. In a desktop computer the mouse is placed on a flat surface like mouse pad or desk in front of computer. When we have a mouse in our palm and we move the palm in any direction, the mouse will convert the palm's movement into an equivalent digital signal. The digital signal is used to move the pointer on the computer screen. Some basic operations of a mouse are as below:

Point: to move your mouse pointer (↖) to a specific location on the screen.

Click: It is pointing to an item and then single time press and release of the mouse's main button i.e., left mouse button. Generally, this is done to select an item or menu command or to identify a location on a computer screen.

Right-Click: It is the press of right button of a mouse. Generally, to open a dropdown menu list to choose what we can do more with the item, like copy, paste, open, print, etc.

Double Click: It is pressing & releasing of the left mouse button on a spot twice, rapidly.

Drag & Drop: It is a process of pointing an icon on the screen, pressing the left mouse button (without releasing it), moving the mouse pointer to a different location and release the left button. To move an item is called dragging whereas placing it somewhere is dropping.

Scrolling: Mouse with single axis digital wheel is very common nowadays. It can be depressed and used as a third button for scrolling. Scrolling is a process to navigate a webpage or document with a given scroll button.
Some mouse has some extra buttons for performing other special tasks like webpage forward or backward, volume up or down. Mice are available in both wireless and physical wired connections. Fig. 1.5 shows the traditional mechanical mouse and the most popular wireless mouse of nowadays.

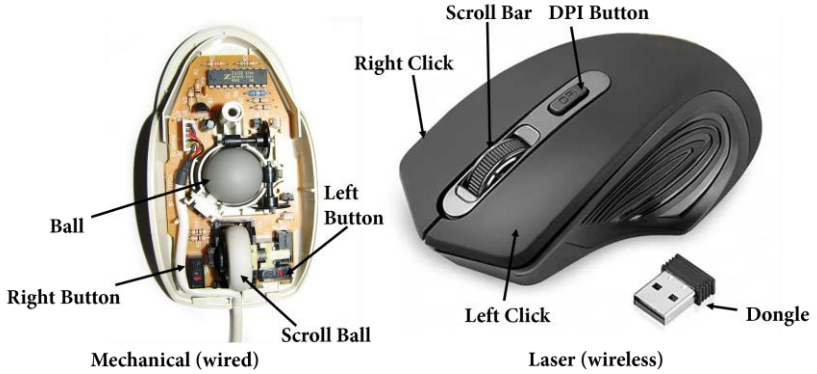


Fig.1.5: Common Parts of Computer Mice (a) Mechanical (b) Wireless

Mechanical mouse: As the name implies these mice have some mechanical structure with a hard rubber ball to detect the motion of the mouse. Sensors inside the assembly interpret the rubber ball movement into the equivalent electronic signal. Due to mechanical driven functionality, its parts like wheels and sensors will wear out over time.
Optical and laser mouse: Uses an LED sensor and imaging arrays of photodiodes to detect the relative movement on the underlying surface. Such mice are not able to work properly on surfaces which does not reflect light properly like glass, plastic, etc. A laser mouse is also an optical mouse having laser light for sensing mouse movement despite LED or photodiode.

1.6 Hard Disk Drive (HDD)

The standard hard disk drive (HDD) is a type of nonvolatile memory (NVM). HDD stores operating systems files, application problems, media and other documents. A hard drive uses a disk and magnets to write data on the disk permanently, even in the event of a power failure. HDD can be used to store and retrieve digital information using platters or rotating disks. Data can be read in a random access manner; means we can store and retrieve data in any order rather than sequentially. A hard device is also required for the installation of any program or files you want to keep on your computer. When we download the files they are

permanently saved on our hard disk. The cost per bit stored on the hard disk is very less compared to other storage media.
A hard disk is a magnetic storage medium for a microcomputer. A computer's hard drive consists of, various disks with read/write heads, a driver motor (used to spin the disks), and a small amount of circuitry which is sealed with a metal case to protect the disks from dust.
The hard disk drives are consists of four key components inside the casing:

Platters: A HDD consists circular disks called platters sealed with container which store data inside the hard disk in the form of 1s and 0s. To increase the overall capacity of the drive, several platters are used. Platter's speeds correlate with read/write rates.
Spindle: It is used to place the platters in position and rotate as it requires.

Read/Write arm: It is used to control the read /write heads. The actual work of reading/writing arm is to convert the magnetic surface into electric current.

Actuator: It is used to control the movement of the read/write arm and transfer data to and from the platters. An actuator is responsible for ensuring the exact position of the read/write arm.

HDD Size: The hard drive is mostly capable of storing more data than any other drive, but its size can change depending on the type of drive. Older hard drives had a storage size of several 100MB to several GB. Newer hard drives have a storage size of several hundred GBs to several TBs.

Advantages of HDD: It has a Fast start up and produces very little noise. HDDs are environment friendly and produce minimum heat on working. These are light weighted so Ideal for Laptops. As compared to other drives HDDs consume less power.

1.7 SOLID STATE DRIVE (SSD)

Modern computers are now using solid-state drives as the primary storage device, rather than HDD. HDDs are very slow as compare to SSDs, for reading and writing data. SSDs are replacing HDDs. Now the configuration is being made in such a way that SSD is used as the master drive for installing the operating system and other software on it, and HDD is being used as secondary storage to store documents, downloads, and audio or video files. HDDs are less expensive than SSDs. However, more and more laptops are beginning to utilize SSD over HDD, helping to improve the reliability and stability of laptops. Various components of HDDs and SSDs are depicted in Fig. 1.6.

Comparison between HDD and SSD
SSD and HDD both are Hard Disk Drive
SSD has high read/write performance for random and sequential data retrieval as compared to HDD.
SSD is now more popular as compared to HDD in desktops and Laptops.

SSD uses the newest way to read/write data using tacking chips in a grid whereas HDD uses magnetic properties to read/write data. Thus HDD has a frequent mechanical breakdown as compare to SSD.
SSD generates little to no noise and HDD can sometimes be one of the loudest components in a computer.

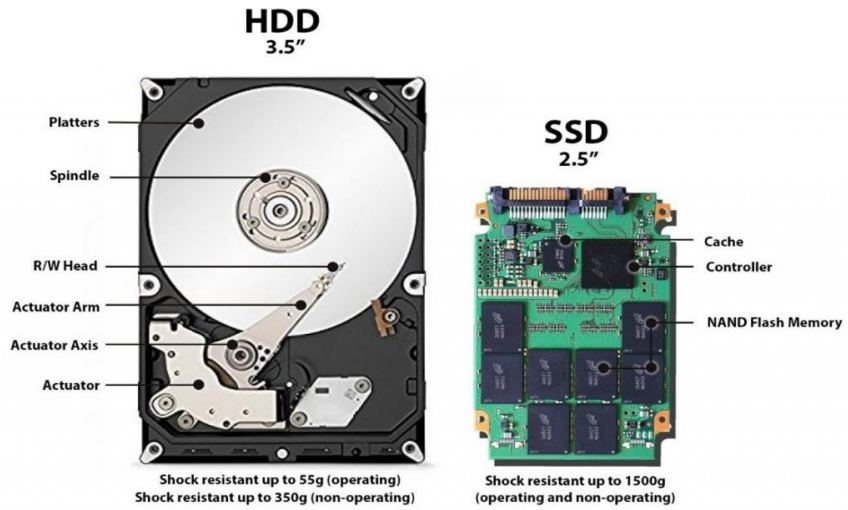


FIG 1.6: Parts of HDDs and SSDs

1.8 OTHER PERIPHERAL DEVICES

A peripheral device also called an auxiliary device is any connected device that provides a computer with additional features. Usually, the word peripheral is used to refer to a device external to the computer case, like a scanner, but the devices located inside the computer case are also technically peripherals. Peripheral is commonly divided into three kinds input device, output device, storage device. Let us understand various peripherals devices:

1.8.1 Input Devices

These are used to send data or commands to the computer system. Some commonly used devices are mouse, keyboard, scanner, barcode reader, webcam, microphone, Digital Camera, Light pen, joystick, stylus Graphic tablet, Touch screen. Some popular input devices are discussed and depicted in Fig. 1.6 & 1.7.

Scanner: Scanner is an input device that is used for direct data entry from the source document into the computer system. It converts document image into digital form and saves into the computer for future prospect.

Bar Code Reader: Barcode reader or barcode scanner is an optical device (scanner) that is used to read barcodes in the form of lines, decode the information contained in the barcode and send the information to a computer. It is being used commonly for automated, fast and reliable data entry operations. We can see its usage in the shopping market to track the price of goods, tracking parcel or postage, or in libraries where each book has a bar code to uniquely identify its details.

Webcam: It is used to capture image and video and convert it into digital form. It has no inbuilt memory so they require computer storage to save captured data.

Fig.1.7: Input Devices (a) Scanner (b) Barcode Reader (c)Webcam

Microphone: It is a voice input device that allows users to input audio into the computer system. It is used in a computer system for taking audio input for its various applications like online chatting, computer gaming, voice recording, voice recognition and many more.

Digital Camera: It is used to take pictures digitally. It allows the user to store the captured media files (audio, video) in a memory card and transfer them to a computer. Digital cameras have become very popular and inexpensive in recent years.

Light Pen: It looks like a pen. It is light sensitive device, made up of photocells and an optical system in a small tube. It is mainly used to select items on the computer screen, for drawing pictures and writing directly in a document file using a computer screen.

Joystick: It is generally used to play games conveniently on the computer or other gaming device. It controls the objects, players and vehicles of the computer game.

Fig. 1.8: Input Devices (a) Microphone (b) Digital Camera (c) Joystick

Graphic Tablet: It is also known as a digitizer. It is used to convert hand-drawn artwork into digital file formats e.g. png, jpeg, etc. Users use the stylus to draw graphics on a surface as we draw on paper using a pen or pencil.
Stylus: Using this device we can draw or write on the digitizer's surface and touchscreen.
Touch screen: Widely used for portable devices such as smartphones, tablets, laptops, notebooks. It allows users to input via gestures of hand or stylus.

1.8.2 Output Devices

It provides processed data saved on the computer as output to the users. Some output devices are monitors, projector, printers, speakers, Braille readers, plotters, Television, video card, sound card Radio. We have discussed the primary output device i.e. monitor in our earlier section. Now let us understand some others output devices.

Projector: It is an output device that projects an image, video onto a large surface, like a white screen or wall.

Printer: It is an output external hardware device that takes the electronic data or information on a computer or any other device and converts it into a hard copy. In other words, a printer is an output device that prints a paper document that includes text, images, or a combination of both. The printer output is called a hard copy, as it is a physical form of an electronic document. The quality of a printer depends on various factors like printing color, resolution, speed, etc.

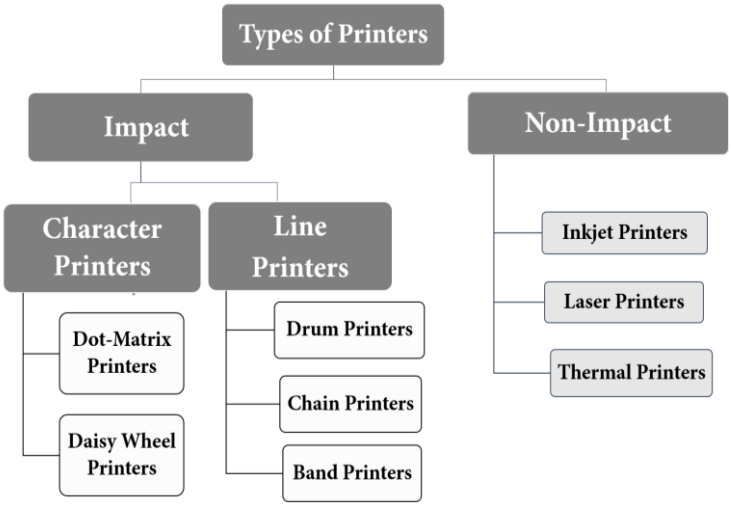


Fig.1.9. Classification of Printers

Nowadays two types of printers are mostly used, inkjet and laser printers. Types of printers are depicted in Fig. 1.9. Dot-matrix and daisy wheel printers are impact printers which print single character at a time whereas drum, chain and band printer prints complete line. Above discussed printers were impact printers that use mechanical moving components for printing. In the case of a nonimpact printer, there is no mechanical moving component used. Inkjet, laser and thermal printers lie in this category. Inkjet printers are commonly used by consumers or suppliers, while laser printers are used by the businesses world, where they require high resolution or highspeed printing. Dot matrix printers, which have become increasingly rare but are still used for basic text printing. Dot-matrix printer speed and resolution are less as compare to laser printer.

Speaker: A computer speaker is the primary output device for audio output. It is hardware devices that convert a computer's sound card signals into audio form. It was coming as separate external hardware in PCs but now in laptops and modern PCs it is coming as an onboard preassembled unit

Braille Reader: It is a peripheral device that is mainly used for a blind person to read text displayed on a computer screen. Braille readers are also called braille displays it is a separate device as a part of the keyboard.

Plotter: It is similar to printers but was used to produce vector graphics drawings. It uses various writing tools (e.g. pencil, pen, marker, etc.) instead of toner for printing. A

conventional printer draws series of dots whereas a plotter device draws multiple, continuous lines onto paper.

Graphics card: It is an expansion card that attaches to the slot residing on the motherboard. It is used to process the images and videos and enables higher resolution graphics to rapidly visualize on a display screen.

Sound Card: It controls the output of sound signals, which enable speaker and headphones to work.

1.9 SUMMARY

The proliferation of internet services requires basic internet skills. Browsers are user agents to send and receive information on the internet. Search engines are powerful tools to retrieve desired information in minimal time.

Good governance by e-Governance is achieved by the digital India program. It aims to provide minimal government and maximum governance. Computer hardware is various physical (tangible) components of computers. The CPU consists of ALU, Control Unit and Memory storage unit. The ALU can be called the brain of the computer without it the computer would be more or less useless. Input and Output devices are essential components of a computer system.

Primary memory is called a temporary or volatile memory as whatever data or instructions are stored in it, remain till power is ON. Secondary memory is mainly used to store data permanently and is also known as nonvolatile memory. A Computer monitor is simply the screen that will present you digital output from the computer. Various type of monitors are CRT, TFT, LCD, LED, Plasma, etc. The keyboard is used to type letters, numbers, symbols, and send instructions to a computer. The printer is an output device that prints document having text, images, or a combination of both. The output of the printer is called a hard copy.

HDD is a nonvolatile memory used to store operating systems files, application problems, media and other documents. Peripheral devices can be input/output, internal or external.

1.10 PRACTICE EXERCISES

Subjective Questions

- What is a web browser? Discuss common features of a web browser.
- How does a Search Engine work? Explain its searching process.
- What is DigiLocker? Explain its features.
- Explain types of printers.
- Write a Google search query to get:
- official websites of the Government of India.
- web results available under the name "India New Education Policy", get web results with pdf file.
- web results that give information about laptops in the range of Rs 25000 to Rs 35000. These results should come from the website of an anyone e-commerce company and should include laptops of the desired brand.

Further Readings

1. Nortorn, P. Introduction to Computers, 7th Edition, 2017

2 .Rajaraman, V., Fundamentals of Computers, PHI, 2014.

3. Larry E. Long and Nancy Long, Computers: Information Technology in Perspective, PHI, 2001

4. Andy Channelle, Beginning OpenOffice 3, Apress, 2009

B.Sc.(DATA SCIENCE)

SEMESTER-I

FUNDAMENTAL OF IT

UNIT II: COMPUTER MEMORY AND INTERNET

STRUCTURE

2.0 OBJECTIVE

- Learning representation of information: BIT, BYTE, Memory, Memory size;
- Understanding different types of memories like RAM, ROM, PROM, EPROM
- Knowing about Computer languages: Machine language, assembly language, higher level language, 4GL, Compiler, Interpreter, Assembler, System Software, Application Software.

2.1 MEMORY UNIT

The instructions and data given to the computer are stored in the memory or storage unit. This data along with the program instructions are used by the CU and ALU. It is also used to store intermittent results and information (final results). Types of memory are discussed in detail, in the next topic.

The smallest unit of memory is called a 'Bit'. A bit can have the value 1 or 0 which is known as binary values. Groups of eight bits form a Byte and similarly higher order units are formed. The Table 1.1 shows measurement units for digital data with their denoting symbol and corresponding capacity.

Table1.1: Measurement Units for Digital Data

Unit	Symbol	Capacity
Bit	B	1 or 0 (on or off)
Byte	B	8 bits
Kilobyte	KB	1024 Bytes
Megabyte	MB	1024 Kilobyte
Gigabyte	GB	1024 Gigabyte

Unit	Symbol	Capacity
Terabyte	TB	1024 Gigabytes
Petabyte	PB	1024 Terabyte
Exabyte	EB	1024 Petabytes
Zettabyte	ZB	1024 Exabytes
Yottabyte	YB	1024 Zettabytes

Computer memory is one of the most important components of the computer system. Computer memory is a vital resource that is managed by the operating system. When the data is sent to the memory it is kept at some particular location called to address. The data can be retrieved by the computer from this address as and when required.

2.2 TYPES OF COMPUTER MEMORY

The computer system makes use of different types of memory depending upon the functional requirement. Depending on the direct accessibility of memory by CPU, memories are classified as a primary and secondary type. Further, the main memory is divided into two types based on the data retention by the system memory, volatile and non-volatile.

2.2.1 Primary Memory

Primary memory is known as main memory or internal storage because it is directly accessible by the CPU. It is used to store program instructions, data and intermittent results. It is made of semiconductor devices. Due to its fast access rate and circuit complexity, it is expensive in comparison to secondary memory. A computer can't work if there is no primary memory installed into the system. RAM, ROM, Cache Memory are an example of primary memory.

A. Random Access Memory (RAM)

It is called Random access memory due to its feature that access time to any stored information is independent of the physical location of data. RAM is also known as a temporary or volatile memory because whatever data stored in it remains till the computer is switched ON. When the current is switched off, all stored data will be wiped out or lost. RAM is the most essential element of a computer system because, without it, the system cannot perform its tasks. RAM is further classified into two types (a) Static RAM (b) Dynamic RAM

Static Random Access Memory (SRAM)

The word static indicates that the memory retains its contents as long as power remains supplied. However, data is lost when the power gets down due to its volatile nature. SRAM is faster and much more expensive than DRAM.

Dynamic Randon Access Memory (DRAM): *DRAM* is constructed of tiny capacitors that leak electricity. Designers use DRAM because it is much denser (can store many bits per chip), uses less power, and generates less heat than SRAM. For these reasons, both technologies are often used in combination: DRAM for main memory and SRAM for the cache.

B. Read-only Memory (ROM)

The programs stored in ROM are permanent and are not lost or erased when the current is switched off. So, it is a nonvolatile memory type. The programs stored in ROM are generally of critical in nature & given by the manufacturer of the computer and includes operating system programs, booting program, etc. ROM is available in different types, including PROM, EPROM and EEPROM.

Programmable Read Only Memory (PROM): PROM is read-only memory that can be modified only once by a user. The user buys a blank PROM and enters the desired contents using a PROM programmer.

Erasable and Programmable Read Only Memory (EPROM): It is programmable with the added advantage of being reprogrammable (erasing an EPROM requires a special tool that emits ultraviolet light).To reprogram an EPROM, the entire chip must first be erased.

Electrically Erasable and Programmable Read Only Memory (EEPROM): The EEPROM is programmed and erased electrically. It can be erased and reprogrammed about ten thousand times. Both erasing and programming take about 4 to 10 ms (milli second). In EEPROM, any location can be selectively erased and programmed. EEPROMs can be erased one byte at a time, rather than erasing the entire chip.

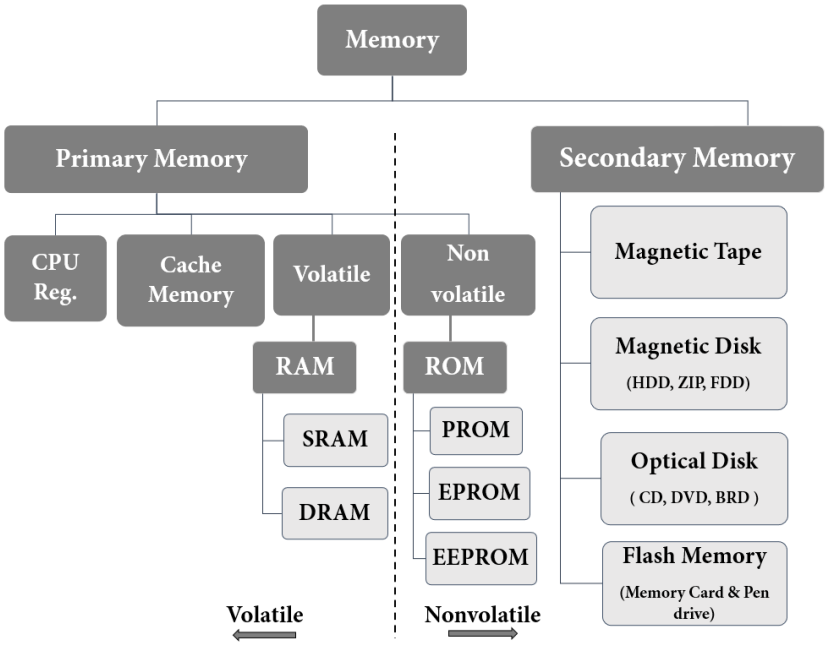


Figure 1: Classification of Memories

Flash memory: It is essentially EEPROM with the added benefit that data can be written or erased in blocks, removing the one-byte-at-a-time limitation. This makes flash memory faster than EEPROM.

2.2.2 Secondary Memory

Generally, the amount of storage available in the main/primary storage unit becomes insufficient when loading large programs or simultaneous processing of programs e.g. complex business problems. In such situations, it is necessary to use external or auxiliary memory for storing data. Here the secondary memory comes into the picture and it is mainly used to store data permanently. It is also termed as 'external memory' due to lack of direct access between CPU and the memory. It is a nonvolatile memory; data retains even after the computer system is switched off or electric power is disconnected.

A. Hard Disk Drive (HDD)

The standard hard disk drive (HDD) is a type of nonvolatile memory (NVM). HDD stores operating systems files, application problems, media and other documents. A hard drive uses a disk and magnets to write data on the disk permanently, even in the event of a power failure. HDD can be used to store and retrieve digital information using platters or rotating disks. Data can be read in a random access manner; means we can store and retrieve data in any order rather than sequentially. A hard device is also required for the installation of any program or files you want to keep on your computer. When we download the files they are permanently saved on our hard disk. The cost per bit stored on the hard disk is very less compared to other storage media.

A hard disk is a magnetic storage medium for a microcomputer. A computer's hard drive consists of, various disks with read/write heads, a driver motor (used to spin the disks), and a small amount of circuitry which is sealed with a metal case to protect the disks from dust.

The hard disk drives are consists of four key components inside the casing:

Platters: A HDD consists circular disks called platters sealed with container which store data inside the hard disk in the form of 1s and 0s. To increase the overall capacity of the drive, several platters are used. Platter's speeds correlate with read/write rates.

Spindle: It is used to place the platters in position and rotate as it requires.

Read/Write arm: It is used to control the read /write heads. The actual work of reading/writing arm is to convert the magnetic surface into electric current.

Actuator: It is used to control the movement of the read/write arm and transfer data to and from the platters. An actuator is responsible for ensuring the exact position of the read/write arm.

HDD Size: The hard drive is mostly capable of storing more data than any other drive, but its size can change depending on the type of drive. Older hard drives had a storage size of several 100MB to several GB. Newer hard drives have a storage size of several hundred GBs to several TBs.

Advantages of HDD: It has a Fast start up and produces very little noise. HDDs are environment friendly and produce minimum heat on working. These are light weighted so Ideal for Laptops. As compared to other drives HDDs consume less power.

B. Solid State Drive (SSD)

Modern computers are now using solid-state drives as the primary storage device, rather than HDD. HDDs are very slow as compare to SSDs, for reading and writing data. SSDs are replacing HDDs. Now the configuration is being made in such a way that SSD is used as the master drive for installing the operating system and other software on it, and HDD is being used as secondary storage to store documents, downloads, and audio or video files. HDDs are less expensive than SSDs. However, more and more laptops are beginning to utilize SSD over HDD, helping to improve the reliability and stability of laptops.

Comparison between HDD and SSD

- SSD and HDD both are Hard Disk Drive
- SSD has high read/write performance for random and sequential data retrieval as compared to HDD.
- SSD is now more popular as compared to HDD in desktops and Laptops.
- SSD uses the newest way to read/write data using tacking chips in a grid whereas HDD uses magnetic properties to read/write data. Thus HDD has a frequent mechanical breakdown as compare to SSD.
- SSD generates little to no noise and HDD can sometimes be one of the loudest components in a computer.

2.3 SOFTWARE AND HARDWARE

Hardware is the term given to the physical components of a computer: e.g. keyboard, monitor, system box or floppy disk drive. Software, on the other hand, is electronic information: files, operating system, graphics, computer programs are all example of software. The difference between hardware and software reflects the duality between the physical and mental worlds: for example, your brain is hardware, whereas your mind is software. Software is the stuff that makes your computer do things for you. The computer without software would be like a home entertainment system with no tapes, CDís, or movies - you have the machine, but thereís nothing to play on it. Software is continually developed. Each time the software maker (Microsoft, Adobe, Corel, etc) develops a new version of their software they assign it a version number. Before Microsoft Word 7, there was Microsoft Word 6.0.1, and before that Word 6.0. The larger the developments made to the software, the larger the version number changes. Usually a large change will result in a whole number upgrade; a small change may result in a tenth of a decimal place. Hardware are those components or physical pieces (things you can touch) that make up the computer. The different pieces of the computerís hardware are monitor, speakers, mouse, CDROM, floppy drive, hard drive, keyboard, CPU, RAM, Processor, etc. Each piece plays a role in the operation of a computer.

2.3.1 System Software

System Software is the type of software which is the interface between application software and system. Low level languages are used to write the system software. System Software maintains the system resources and gives the path for application software to run. An important thing is that without system software, system can not run. It is a general purpose software.

2.3.2 Application Software

Application Software is the type of software that runs as per user request. It runs on the platform which is provided by system software. High level languages are used to write the application software. Its a specific purpose software.

2.4 MACHINE LANGUAGE

Sometimes referred to as machine code or object code, machine language is a collection of binary digits or bits that the computer reads and interprets. Machine language is the only language a computer is capable of understanding. Therefore all instructions and data should be written using binary codes 1 and 0. The binary code is called the machine code or machine language. The exact machine language for a program or action can differ by operating system on the computer. Computer programs are written in one or more programming languages, like C++, Java, or Visual Basic.

A computer cannot directly understand the programming languages used to create computer programs, so the program code must be compiled. Once a program's code is compiled, the computer can understand it because the program's code has been turned into machine language. A compiler checks the entire user-written programme (known as source

programme) and produces a complete programme in machine language (known as object programme).

The source programme is retained for possible modifications and corrections and the object programme is loaded into the computer for execution.

For example, a program instruction may look like this:

1011000111101 Typical Machine language Instruction format

1. OPCODE (Operation code) OPCODE tells the computer which operation to perform from the instruction set of the computer.

2. OPERAND (Address/Location) OPERAND tells the address of the data on which the operation is to be performed.

Advantage of Machine Language

The only advantage is that program of machine language run very fast because no translation program is required for the CPU.

Disadvantages of Machine language

- It is machine dependent i.e. it differs from computer to computer.
- It is difficult to program and write.
- It is prone to errors
- It is difficult to modify.

2.5 ASSEMBLE LANGUAGE

It is a low level programming language that allows a user to write a program using alphanumeric mnemonic of instructions. It requires a translator as assembler to convert language into machine language so that it can be understood by the computer. It is easier to remember and write than machine language.

Using alphanumeric mnemonic codes instead of numeric cods for the instruction in the instruction set e.g. using ADD instead of 1110 (binary) or 14 (decimal) for instruction to add. Allowing storage location to be represented in form of alphanumeric address instead of numeric address e.g. representing memory locations 1000, 1001 etc.

Advantages of Assembly Language

1. The symbolic programming of Assembly Language is easier to understand and saves a lot of time and effort of the programmer.
2. It is easier to correct errors and modify program instructions.
3. Assembly Language has the same efficiency of execution as the machine level language. Because this is one-to-one translator between assembly language program and its corresponding machine language program.

Disadvantages Assembly Language

1. One of the major disadvantages is that assembly language is machine dependent. A program written for one computer might not run in other computers with different hardware configuration. Long programs written in such languages cannot be executed on small sized computers.

2. It takes lot of time to code or write the program, as it is more complex in nature.

2.6 HIGH LEVEL LANGUAGES

High level language is abbreviated as HLL. High level languages are similar to the human language. Unlike low level languages, high level languages are programmers friendly, easy to code, debug and maintain.

High level language provides higher level of abstraction from machine language. They do not interact directly with the hardware. Rather, they focus more on the complex arithmetic operations, optimal program efficiency and easiness in coding.

Low level programming uses machine friendly language. Programmers writes code either in binary or assembly language. Writing programs in binary is complex and cumbersome process. Hence, to make programming more programmers friendly. Programs in high level language is written using English statements.

High level programs require compilers/interpreters to translate source code to machine language. We can compile the source code written in high level language to multiple machine languages. Thus, they are machine independent language.

Today almost all programs are developed using a high level programming language. We can develop a variety of applications using high level language. They are used to develop desktop applications, websites, system software's, utility software's and many more.

High level languages are grouped in two categories based on execution model – compiled or interpreted languages.

Advantages of High level language

1. High level languages are programmer friendly. They are easy to write, debug and maintain.
2. It provide higher level of abstraction from machine languages.
3. It is machine independent language.
4. Easy to learn.
5. Less error prone, easy to find and debug errors.
6. High level programming results in better programming productivity.

Disadvantages of High level language

1. It takes additional translation times to translate the source to machine code.
2. High level programs are comparatively slower than low level programs.
3. Compared to low level programs, they are generally less memory efficient.
4. Cannot communicate directly with the hardware

2.7 LANGUAGE PROCESSORS

Compilers, interpreters, translate programs written in high-level languages into machine code that a computer understands. And assemblers translate programs written in low-level or assembly language into machine code. In the compilation process, there are several stages. To help programmers write error-free code, tools are available.

Compiler: the language processor that reads the complete source program written in high-level language as a whole in one go and translates it into an equivalent program in machine language is called a compiler. Example: c, c++, c#, java.

In a compiler, the source code is translated to object code successfully if it is free of errors. The compiler specifies the errors at the end of the compilation with line numbers when there are any errors in the source code. The errors must be removed before the compiler can successfully recompile the source code again

Assembler: the assembler is used to translate the program written in assembly language into machine code. The source program is an input of an assembler that contains assembly language instructions. The output generated by the assembler is the object code or machine code understandable by the computer. Assembler is basically the 1st interface that is able to communicate humans with the machine. We need an assembler to fill the gap between human and machine so that they can communicate with each other. Code written in assembly language is some sort of mnemonics(instructions) like add, mul, mux, sub, div, mov and so on. And the assembler is basically able to convert these mnemonics in binary code. Here, these mnemonics also depend upon the architecture of the machine

Interpreter: the translation of a single statement of the source program into machine code is done by a language processor and executes immediately before moving on to the next line is called an interpreter. If there is an error in the statement, the interpreter terminates its translating process at that statement and displays an error message. The interpreter moves on to the next line for execution only after the removal of the error. An interpreter directly executes instructions written in a programming or scripting language without previously converting them to an object code or machine code

2.8 WORLD WIDE WEB

World Wide Web, which is also known as a Web, is a collection of websites or web pages stored in web servers and connected to local computers through the internet. These websites contain text pages, digital images, audios, videos, etc. Users can access the content of these sites from any part of the world over the internet using their devices such as computers, laptops, cell phones, etc. The WWW, along with internet, enables the retrieval and display of text and media to your device.

The building blocks of the Web are web pages which are formatted in HTML and connected by links called "hypertext" or hyperlinks and accessed by HTTP. These links are electronic connections that link related pieces of information so that users can access the desired information quickly. Hypertext offers the advantage to select a word or phrase from text and thus to access other pages that provide additional information related to that word or phrase.

A web page is given an online address called a Uniform Resource Locator (URL). A particular collection of web pages that belong to a specific URL is called a website, e.g., *www.facebook.com*, *www.google.com*, etc. So, the World Wide Web is like a huge electronic book whose pages are stored on multiple servers across the world.

Small websites store all of their WebPages on a single server, but big websites or organizations place their WebPages on different servers in different countries so that when users of a country search their site they could get the information quickly from the nearest server.

Difference between World Wide Web and Internet

Some people use the terms 'internet' and 'World Wide Web' interchangeably. They think they are the same thing, but it is not so. Internet is entirely different from WWW. It is a worldwide network of devices like computers, laptops, tablets, etc. It enables users to send emails to other users and chat with them online. For example, when you send an email or chatting with someone online, you are using the internet.

But, when you have opened a website like google.com for information, you are using the World Wide Web; a network of servers over the internet. You request a webpage from your computer using a browser, and the server renders that page to your browser. Your computer is called a client who runs a program (web browser), and asks the other computer (server) for the information it needs.

2.8.1 History of the World Wide Web

The World Wide Web was invented by a British scientist, Tim Berners-Lee in 1989. He was working at CERN at that time. Originally, it was developed by him to fulfill the need of automated information sharing between scientists across the world, so that they could easily share the data and results of their experiments and studies with each other. ERN, where Tim Berners worked, is a community of more than 1700 scientists from more than 100 countries. These scientists spend some time on CERN site, and rest of the time they work at their universities and national laboratories in their home countries, so there was a need for reliable communication tools so that they can exchange information.

Internet and Hypertext were available at this time, but no one thought how to use the internet to link or share one document to another. Tim focused on three main technologies that could make computers understand each other, HTML, URL, and HTTP. So, the objective behind the invention of WWW was to combine recent computer technologies, data networks, and hypertext into a user-friendly and effective global information system.

2.9 PRACTICE EXERCISES

- What is a web browser? Discuss common features of a web browser.
- How does a Search Engine work? Explain its searching process.
- What is DigiLocker? Explain its features.
- Explain types of printers.
- Write a Google search query to get: official websites of the Government of India.

.REFERENCES

1. Hunt, R., J. Shelley, Computers and Commonsense, Prentice Hall of India.
2. Sinha, Pradeep K. and Preeti Sinha, Foundation of Computing, BPB Publication.
3. Saxena, Sanjay, A First Course in Computers, Vikas Publishing House.

B.Sc.(DATA SCIENCE)

SEMESTER-I

FUNDAMENTAL OF IT

UNIT III: BASIC OF COMPUTER NETWORKS

STRUCTURE

3.0 OBJECTIVE

- Understanding Computer Networks and different kind of networks
- Understanding Internet and its applications
- Understanding web browser, search engines and internet basics

3.1 COMPUTER NETWORK

The network allows computers to connect and communicate with different computers via any medium. Lan, man, and wan are the three major types of networks designed to operate over the area they cover. There are some similarities and dissimilarities between them. One of the major differences is the geographical area they cover, i.e. LAN covers the smallest area; MAN covers an area larger than LAN and WAN comprises the largest of all.

3.1.1 Local Area Network (LAN)

LAN or local area network connects network devices in such a way that personal computers and workstations can share data, tools, and programs. The group of computers and devices are connected together by a switch, or stack of switches, using a private addressing scheme as defined by the TCP/IP protocol. Private addresses are unique in relation to other computers on the local network. Routers are found at the boundary of a LAN, connecting them to the larger wan.

3.1.2 Metropolitan Area Network (MAN)

Man or metropolitan area network covers a larger area than that of a LAN and smaller area as compared to wan. It connects two or more computers that are apart but reside in the same or different cities. It covers a large geographical area and may serve as an ISP (internet service provider). Man is designed for customers who need high-speed connectivity. Speeds of man range in terms of mbps. It's hard to design and maintain a metropolitan area network.

3.1.3 Wide Area Network (WAN)

Wan or wide area network is a computer network that extends over a large geographical area, although it might be confined within the bounds of a state or country. A wan could be a connection of LAN connecting to other LANs via telephone lines and radio waves and may be limited to an enterprise (a corporation or an organization) or accessible to the public. The technology is high speed and relatively expensive.

3.2 INTERNET SKILLS

Internet is a popular tool for accessing digital information & services across the globe. It uses digital devices like computers, mobiles, other gadgets and network devices to exchange information and services. Skills required to access various applications of the internet are commonly termed as "internet skills". These skills are very important nowadays and we will learn, some basic terms pertaining to the internet, how to access with web browsers and use of search engines to find relevant, reliable & precise information (like pearls) from the web (the information ocean).

Internet is an interconnected network of computers all over the world that connects people and information 24 hours a day. The internet is a computer network consisting of a worldwide network of servers that use the TCP/IP network protocols to facilitate data transmission and

exchange and to communicate between networks and devices. It is a network of networks that consists of private, public, academic, business, and government networks of local to global scope, linked by a broad array of electronic, wireless, and optical networking technologies. The internet carries a vast range of information resources and services, such as the inter-linked hypertext documents and applications of the world wide web (www), electronic mail, telephony, and file sharing.

Internet is nothing but a collection of various interconnected networks of heterogeneous types across the globe. It comprises of different kinds of devices, specifications for hardware & software to be connected in a global network and a variety of protocols with a common understanding between various countries, universities, companies and global agencies. It is also referred as a "network of networks". The purpose is to share resources over a global network, a resource can be a simple webpage having information, a network printer, or any digitally accessible service like email, e-commerce, real-time streaming, telnet, etc. It is a global network working over physical cables like traditional pots (plain old telephone system), TV cables, Fiber optic cables and even wireless mediums like wi-fi, 3g/4g or satellites communication depending on need and type of communication.

3.3 TYPES OF INTERNET SERVICE

The type of internet service you choose will largely depend on which internet service providers (ISPs) serve your area, along with the types of service they offer. Here are some common types of internet service.

Dial-up: this is generally the slowest type of internet connection, and you should probably avoid it unless it is the only service available in your area. Dial-up internet uses your phone line, so unless you have multiple phone lines you will not be able to use your landline and the internet at the same time.

DSL: DSL service uses a broadband connection, which makes it much faster than dial-up. DSL connects to the internet via a phone line but does not require you to have a landline at home. And unlike dial-up, you'll be able to use the internet and your phone line at the same time.

Cable: cable service connects to the internet via cable TV, although you do not necessarily need to have cable tv in order to get it. It uses a broadband connection and can be faster than both dial-up and DSL service; however, it is only available where cable TV is available.

Satellite: a satellite connection uses broadband but does not require cable or phone lines; it connects to the internet through satellites orbiting the earth. As a result, it can be used almost anywhere in the world, but the connection may be affected by weather patterns. Satellite connections are also usually slower than dsl or cable.

3G and 4G: 3G and 4G service is most commonly used with mobile phones, and it connects wirelessly through your ISP's network. However, these types of connections aren't always as fast as DSL or cable. They will also limit the amount of data you can use each month, which isn't the case with most broadband plans.

3.4 DOMAIN NAMES AND IP ADDRESSES

An internet protocol, or IP, address is different than a domain name. The IP address is an actual set of numerical instructions. It communicates exact information about the address in a way that is useful to the computer but makes no sense to humans. The domain name functions as a link to the IP address. Links do not contain actual information, but they do point to the place where the IP address information resides. It is convenient to think of IP addresses as the actual code and the domain name as a nickname for that code. A typical IP address looks like a string of numbers. It could be 232.17.43.22, for example. However, humans cannot understand or use that code.

3.5 APPLICATIONS OF THE INTERNET

Internet is being used in a variety of applications, few of them are as under:

1. Communication: Millions of e-mails are sent and received worldwide in a day for exchanging information. Online messengers are also popular for real-time communications. With help of VoIP (voice over the internet protocol) audio and video conversation also take place.
2. E-commerce: The internet provides an online market to sell/purchase various products and services globally. Now, we can purchase things from our neighbouring house to the farthest country as well. These online stores may work round the clock and enable consumers to purchase from home.
3. Storage & file transfer: The user can send and store files of different types. Cloud computing makes it possible to share it among various users with their associated roles to access such files.
4. Live streaming & podcasts: Users can send their live videos and audios to large groups of people in a real-time manner.
5. News, entertainment: Whatever is happening in the real world, it can be updated as a piece of information on the web. Now, it is possible to provide breaking news, entertainment, election results or about the sports activity of your favourite team or sportsperson.
6. Collaborative tasks: People can meet online, discuss things and work together using collaborative applications.
7. Research & learning activities: With the availability of information and online resources, researcher and learners can continually equip themselves.
8. Interactive gaming: We can play & interact online with another human being, a computer program having a real-time conversation. We can participate in the game across the world.
9. Social networking: Various online platform provides facility to connect the people of the same interest. Millions of people daily connect with their colleagues, family members and find new persons on such platforms.
10. Job hunting: Job providers can post various vacancies on the internet via their portal, third-party portals or social media. Jobseekers search for various jobs on portals, newspapers & apply their applications with resume to concerning human resource managers.

11. Navigation & tracking: Searching best routes on digital maps, tracking the live status of trains, cars, parcels are few widely used location-based services of the internet.

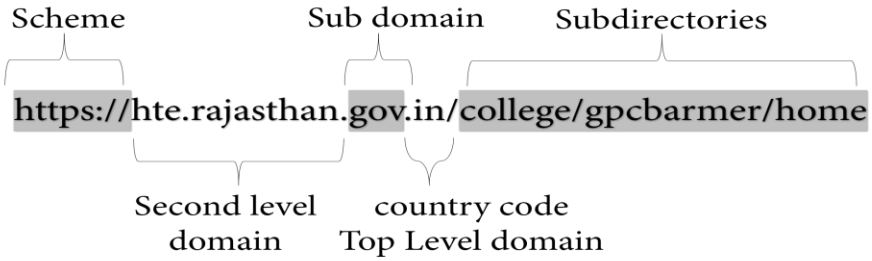
In addition to the above, real-time updates, cashless transactions, online booking, live trading and advertisements, etc. Are also some popular uses of the internet.

3.6 GLOSSARY FOR THE INTERNET

Few commonly used terms for the internet basics are:

www: world wide web or simply 'web' is the most popular use case of the internet. It is an information system where different kinds of files or resources are hosted and uniquely accessed via url address. These resources are interlinked with hyperlinks. The resources can be uploaded or downloaded with application software known as web browser using some standard protocols like http or ftp.

Domain Name: it is a human understandable unique name on the internet to identify a computer system or resource.



Url: uniform resource locator is known as the web address. It is a unique identifier of a web resource with a specification of how to access and from where to access it. Http: hypertext transfer protocol is a set of rules (protocol) that define the way how data transfer over the web. It is used to access websites.

Https: it is the secure version of http using ssl (secure socket layer) encryption.

Ftp: file transfer protocol is a set of rules (protocol) that define the way how data transfer over the internet. It is used to transfer a large file from one host to another.

Hyperlink: it is a word, phrase or image that refers to another data. Can be followed by the user by clicking or tapping. The reference may belong to another document or specific element of the same document.

Browser: it is an application program with a user interface to display and navigate web pages over www.

Webpage: it is a hypertext document designed to view on the web browser.

Website: it is a collection of web pages and related resources that is identified by a common domain name and hosted (published) on a web server.

Search engine: it is a web-based complex software that provides information searching services to its users. The search engine uses various algorithms to search its huge database and produce appropriate search results taking minimal time.

ISP: an internet service provider is a company that provides internet access to other companies or individuals.

Email: electronic mail is a method of exchanging digital messages from one electronic device to another device or to many recipients via a network.
Podcast: it is a web resource available on the internet that contains audio information.

Filetype: every information available on the internet have a certain format and type which is understood by their file type. Information can be in form of documents, audio, video, etc.

Download: it is the process of copying data over the internet from one device to another in direction of a server to a client machine.

Upload: it is also the process of transferring data from one device to another on the internet but from client to server-side.

DNS: the domain name system translates human understandable domain name (for example, www.ncs.gov.in) to machine readable ip address (for example, 203.129.202.69)

TCP/IP: transmission control protocol and internet protocol are set of rules that govern the linking of a computer system to the internet and similar computer networks.

Modem: it is the short name for hardware device modulator-demodulator. It is responsible to convert the digital data of a computer system to an analog signal which can travel over telephone lines (modulator) and vice-versa.

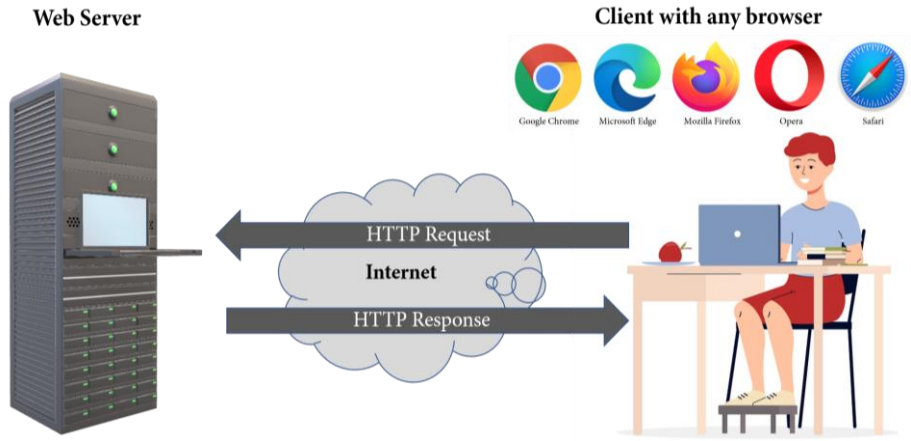
Network equipment: these are networking hardware required for the interconnection and communication in a computer network e.g., bridge, repeater, hub, switch, router, modem, etc.

Cloud computing: it provides computer resources over the internet as per the demand of the user. Resources can be computer infrastructure, computing power or data storage, etc.

3.7 UNDERSTANDING A BROWSER

Services & information provided by the internet follows the *client-server model*. In this model when a client machine seeks some service, it forms a request message (http request) and sends it via a client-side program to network towards the server machine. On the other

side, when a request approaches the server machine it grants or denies the requested service in form of the response message (http response). A browser is a client-side application program to search and retrieve information from the world wide web, available in the form of web pages and display it on the client's machine. It is also termed as "web browser" or "user agent". we will understand some trending variants of the browser, their comparison in terms of offered features and the basic architecture of a web browser for its better understanding.



Google chrome: it is the most widely used web browser developed by google. It is cross-platform web browser that was firstly released in 2008 for the windows operating system of microsoft. This browser is now a proprietary freeware based on google's free and open-source software (foss) project "chromium". It is widely used due to its speed & security capabilities. It is constantly updating and keeps us safe from phishing and malware scams. Web store keeps chrome customizable via its various themes, extension and web apps. It can translate a website in different languages.

Microsoft edge: it is developed by microsoft and was firstly released in 2015 for its proprietary operating system windows 10. It is the successor of the internet explorer web browser of the microsoft family. It is also integrated with microsoft's online platforms for providing voice control, searching functionality and dynamic content related to searches inside the address bar.

Mozilla firefox: it is a free and open-source browser developed by mozilla foundation and its subsidiary mozilla corporation. It was initially released in september 2002. In comparison to other browsers, firefox provides an extensive library of extensions & add-ons to its users for customizing their browser experiences & functionality.

Opera: it is a multi-platform browser developed by opera software. It was initially released in april 1995. It is also available for mobile devices and these mobile versions are known as opera mini & opera mobile. With its artificial intelligence (ai) based platform opera browser supports a personalized news feed at the start page. It also supports sharing files, links and notes between user's different devices with the *opera flow* feature.

Safari: this browser was developed by apple inc. And it is not a complete open-source browser. It was initially released on january 2003 as a part of mac operating system. It is considered as faster browser with considerable high privacy features. Safari also implemented feature of cross site tracking. It natively supports web page translation and picture in picture functionality.

3.8 SEARCH ENGINE

The search engine is a generic name assigned to a software system whose purpose is to systematically search the web pages against supplied search terms, commonly known as 'keywords', 'search query' or 'search phrase'. The result of this search is presented in form of a listing technically referred to as search engine results pages (serp). The search engine is not a mere finding tool but a web service that performs the task of indexing, organizing, rating and reviewing websites too. There are many search engines available in the world of computer networking. Every search engine works in its way and that's why we get different search results for the same search query in different search engines. Some rely on users to maintain a catalogue of web pages where other use their automated advanced software to identify key information available in interlinked websites.

3.8.1 Some Popular Search Engines

Google: it is the most trusted search engine worldwide. It was developed by larry page and sergey brin in 1996 for their academic research project. It was initially known as backrub. It is written in c, c++ and python programming languages. It is being used as a default search engine for various web browsers e.g., chrome, safari and mozilla firefox, etc. Google is using emerging technologies like artificial intelligence (ai), machine learning (ml) to recognize user behaviour, likings and other contextual information and produce better results for its users.

Microsoft bing: it is owned and being maintained by microsoft. It is the successor of previous search engines of microsoft e.g., msn search & windows live search. It was launched in june 2009 and written in asp.net. It provides a variety of search services like web, image, video and map. Unlike google its home page provides various links to current news, weather and links to other information like "on this day in history"

Yahoo: this is the oldest search engine available to internet users. It is founded by jerry yang and david filo in january 1994 as "jerry and david's guide to the world wide web". This search engine is owned by yahoo and originally written in general-purpose scripting language- php.

Baidu: it is among the top performer in the market share of search engines worldwide. It is owned by chinese company baidu, inc. Which is one of the largest artificial intelligence and internet companies in the world. It was incorporated in january 2000 by robin li and eric xu. This search engine holds more than 72% of the chinese search engine market as of june 2021. It offers various services like maps, image search, video search, patent search, legal search, games, etc.

Yandex: it is a search engine prevalently used in russia and was launched in september 1997. It is owned by yandex n.v., a russian-dutch domiciled multinational. Apart from image searching, video searching and web searching, it also provides other services like online text and website translator, maps, email, app analytics and marketing platform.

Duckduckgo: last in our list of the search engine but not the least, duckduckgo(ddg) is a favorite search engine for millions of users (mine too), especially who cares their privacy and want to keep their searching history anonymous. It is created by gabriel weinberg and owned by duck duck go inc. It was launched in september 2008 and its code is written in perl, javascript and python. Many search engines record the search history of their users and profile their surfing, searching habits by giving an excuse for better-personalized search results. In contrast, duckduckgo respects the privacy of its users and displays the same search results to its all users for a given search query. It is against the online tracking of user's data and believes that "your personal data is nobody's business".

3.8.2 Types of Web Searches

In the case of web information retrieval, the intention or need behind searching query is not always informational in nature. Web search queries are classified into three types according to the intention behind the search.

1. Navigational: users may input some search keywords not for seeking direct information about the entered text, in spite they intent to navigate to some website. The purpose is to navigate a website that is in the mind of the user or he/she think that such website should be there or they have visited the website in past. E.g. Users may simply type "jim corbett national park" with intention (not known to search engine) to navigate the official website. The search result will show different search results having beautiful pictures of the national park, weblinks to 'top stories' and videos of the national park and many other related results. Here, by examining the website domain name, a user may navigate to any of the above websites, *https://uttarakhandtourism.gov.in/* or *https://www.corbettonline.uk.gov.in/*
2. Informational: in such web queries user's intention is to find related information about the given search term which may be on some interlinked static webpages. The purpose is to read those pages to acquire facts about the input text. E.g. If a user has entered the search term "jim corbett national park" with an intention to find its history, geography or climate then the user may open any suitable results available infront of him like wikipedia pages or another website.
3. Transactional: the purpose of such web queries to reach a website where further interaction or web-mediated activity is performed. E.g. If the user enters search term "jim corbett national park booking" then out of available search result user will select suitable web result to navigate a website where he will perform booking operation for his intended visit to the national park. User may visit *https://www.corbettonline.uk.gov.in/* and finalize a visit via booking.

Such web queries are also performed widely for shopping, downloading content from the web, etc.

3.9 E-MAIL

E-mail, short for Electronic Mail, consists of messages which are sent and received using the Internet. While there are many different e-mail services available that allow you to create an e-mail account and send and receive e-mail and attachments, we have chosen to focus this class on the services available through Gmail because it is free and one of the more popular e-mail services available.

The Pros

- It's fast. Most messages are delivered within minutes – sometimes seconds – around the world without the inconvenience and cost of using a postal service. In fact, postal service is often referred to as "snail mail" by e-mail users.
- It's personal. While the nature of e-mail is informal, its efficiency is an excellent substitute for telephone conversations.
- You can think through your response. Like a letter, you can type your reply and make changes before sending.
- The sender and the receiver don't have to be working at the same time. E-mail avoids problems such as telephone tag or tying to contact someone in a different time zone.
- E-mail makes it easy to keep a record of your communication. You can save and refer to later copies of the e-mails you send as well as those you receive.
- You can reach a lot of people at once. It is possible to send one message to hundreds of recipients at once, or you can send a private message to one individual.

The Cons

- Junk Mail (also referred to as spam). This is as annoying in e-mail as it is with traditional mail. Most e-mail services now filter incoming mail and sort e-mail messages that are most likely advertisements or scams into a folder called “spam.”
- Friendly spam. Try not to forward unnecessary messages to friends who may not appreciate hearing the latest list of “Top Ten Things…” • Ads. The reason you can get free e-mail services like Gmail is because of advertisements. You pay the price of having to click around them to read your mail.
- Misinterpretation. E-mail arrives without tone or facial expressions, which can lead to misunderstanding.
- E-mail messages can be passed on to others. You should always count on the possibility of your message ending up in the inbox of someone it wasn't intended for.
- You can hide behind e-mail. It's tempting to use e-mail instead of facing a person when you have to deal with an unpleasant situation. It's best to talk to a person face-to-face under these circumstances.

3.10 PRACTICE EXERCISES

Searching on various web browsers on a particular search engine:

1. Open Mozilla Firefox web browser and go to google search engine website i.e., *www.google.com*

Insert any of your favorite search terms (like “nep 2020”) in the search bar

1. What is the way to view a Wikipedia page in regional languages?
2. Explain in detail about the components given in the table, such as the names of the

manufacturers of the components, their purpose, estimated price, etc.

REFERENCES

- D. Anfinson and D. Quammen, *IT Essentials PC Hardware and Software Companion Guide*. Madrid:
    - CISC Press.Pearson Education., 2009.
- M. Meyers, *Mike Meyers' CompTIA A+ guide : essentials : exam 220-701*. New York: Mcgraw-Hill,
    - 2010.
- Computer Rear Panel Connectors pinouts diagrams @ pinouts.ru," *pinouts.ru*. https://pinouts.ru/comp.php (accessed Sep. 19, 2021).

B.Sc.(DATA SCIENCE)

SEMESTER-I

FUNDAMENTAL OF IT

UNIT IV: WORD PROCESSING PACKAGE

STRUCTURE

4.0 OBJECTIVES

- To know the basics of Word Processor Components
- To open, close, save, delete and rename the document
- To use the templates and styles
- To design a table of contents
- To design the resume
- To create the document views

4.1 INTRODUCTION

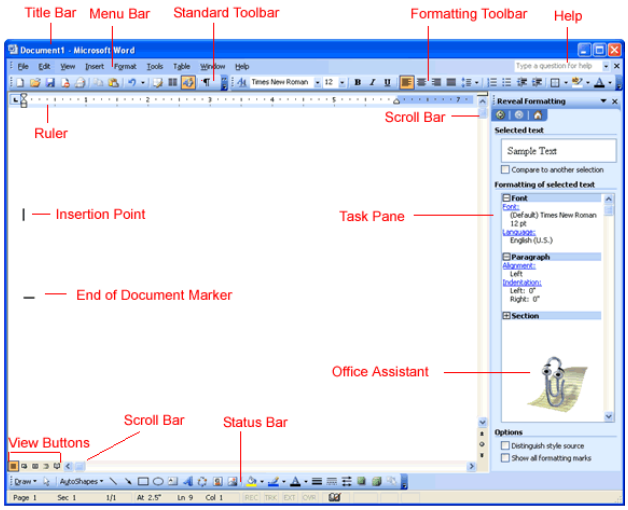
Word Processing Package is a computer application software package which is designed by Microsoft Co-operation. It is used to process and edit the words and also known as computerized typewriter software. Computer and Typewriter have QERTY type keyboard and all the concepts are in Word Processor which are in Typewriter. As compared to typewriter, Word Processor package has many advantages:

- The content of the text can be seen on to the screen rather than directly printing.
- It is easier to modify and edit the information through computer
- Word Processing Package is most popular used package. It is used for every application like industrial, commercial, business, administration, hospital etc.
- You can edit, copy, save, open, and print the document.
- It enables the user to create Graphics, Tables, and Images etc. into the document file.
- It can work in every environment like DOS, WINDOWS, and LINUX etc.

In this chapter, firstly the components of MS-WORD 2007 will be discussed. Then, the basics of word processor like Opening, Closing an existing document and the use of templates have been discussed [13].

4.2 COMPONENTS OF WORD PROCESSOR

The different components of MS-WORD is shown as below:



Title Bar

A horizontal bar is present at the top of an active document. This bar shows the name of the currently open document and application. At the right end, it contains control buttons like Minimize, Maximize, Restore and Close.

Quick Access Toolbar

A customizable toolbar present at the top of an active document. By default, this Toolbar displays the Save, Repeat and Undo buttons and mostly used for accessing frequently used commands. Any of the commands can be added by clicking the dropdown arrow [4].

Ribbon

The Ribbon changes the menus and toolbars into the previous versions. The Ribbon depicts various features that used to be hidden in the File menus. It becomes easier to find and see all commands for document formatting. CTRL + F1. Is used to reduce the Ribbon to a single line of tabs.

Status Bar

A horizontal bar present at the bottom of an active window. It provides the information like number of pages present in the current document, current position of the cursor, current page number etc.

View Toolbar

A toolbar which enables, modifies, and shows different views of an active document's content.

Zoom Button

A button that expands or reduces the document contents in the document window.

Ruler

It helps to adjust the text along with the document. The ruler may set the ruler according to the requirement of the setting. It is used to make the text more presentable.

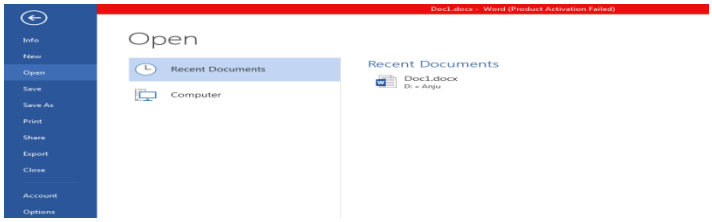
Task Bar

It shows the currently open document along with other settings such as volume control, printer, Clock and CPU information etc.

4.2.1 Opening a Document

When you need to modify the existing document, then document will be opened that allows to make changes in the document like adding, modifying or deleting content. To open an existing word document created in the Microsoft Word, the following steps are taken:

1. Click the File menu.
2. Click on the Open submenu which makes a dialog box to appear.

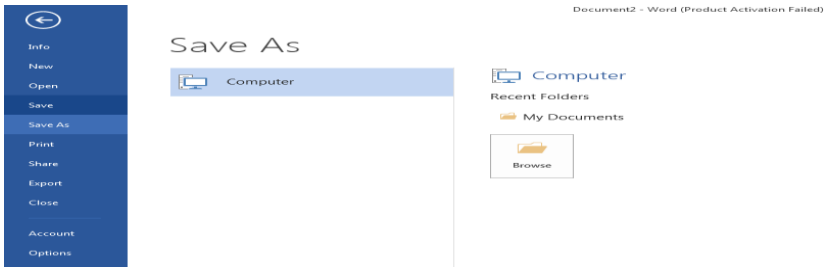


3. In the recent documents, it show the list of all latest documents opened previously from which one can choose the file you want to open.
4. On clicking the Computer icon, locate and open the drive and folder the contains the file.
5. Click the file which one want to open.

4.2.2 Saving a Document

It is necessary to save the newly created files in order to access them later. We need to give some name to the file which we want to save. To save the created file, the following steps are taken:

1. Choose the File menu.
2. Click on Save submenu from the File menu. From keyboard, the combination of Ctrl+S can be pressed. The Save dialog box will appear as



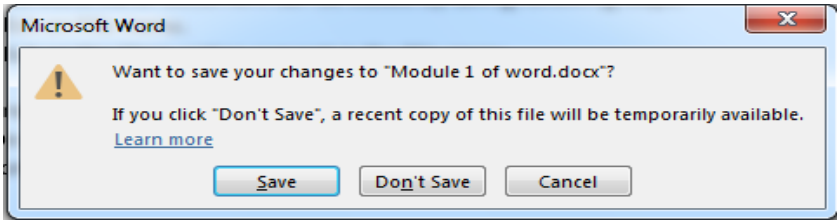
3. Select the required folder from the Browse folder where we want to save the document.
4. In the File name box, write the name of the file by which we want to save the created document.
5. From the Save as type field, we need to specify the format of the file in which we want save our file. The default file format is ".docx".
6. Click on the Save button [16].

4.2.3 Closing a Document

The opened word file or document can be closed by taking following steps:

1. Click on File menu.
2. Click on the Close option present in the File menu.

   OR
3. Press Ctrl+F4 key combination from the keyboard.
4. On clicking the Close option for the already existing opened document, the following dialog box will appear on the screen

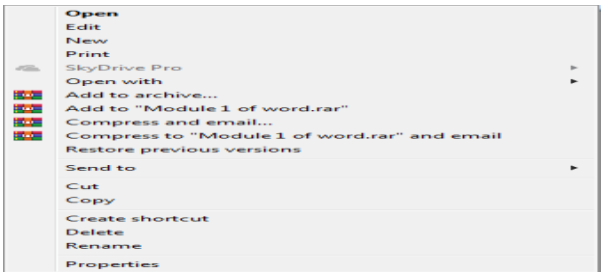


5. To save the document, click on Save otherwise click on Don't Save.
6. On the press of Save button, the document will be saved to the already specified location.
7. On the press of Don't Save, the specified changes in the document will not be reflected back to the specified location.

4.2.4 Renaming the Document

To change the name of the existing document is done using the following steps:

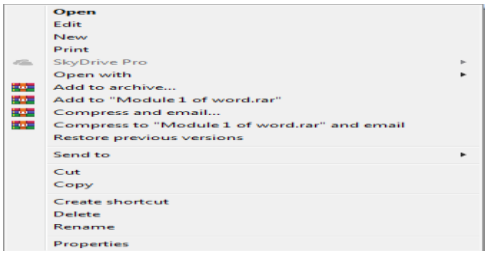
1. Go to the specified location where the document is saved.
2. Right click on the document which we want to rename. The following menu will appear on doing so [8].
3. Click on the Rename option. This will provide the facility to type the new name of the document instead of old name.



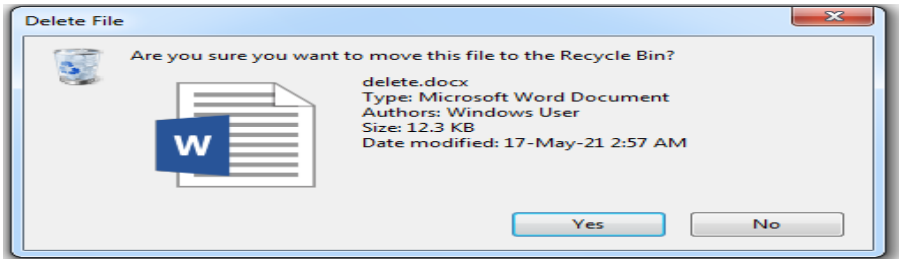
4.2.5 Deleting the Document

To delete the existing document the following steps are followed:

1. Go to the specified location where the document is saved.
2. Right click on the document which we want to delete. The following menu will appear on doing so.



3. Click on the Delete option. The following dialog box appears on the screen asking for the assurance of the user for the deletion of the file.



4. On clicking Yes button, the file gets deleted and move to the recycle bin.
5. On clicking No button, the process gets reverted back and nothing happens.

4.3 USE OF THE TEMPLATES, THEMES AND STYLES

Templates are the files that help to design interesting, attractive, and professional-looking documents. Template contains content and design based elements that are useful in the starting point while creating a document.

A template may be defined as a predefined file with predefined structure, style and look. On the basis of this file, a new document can be created having same structure, style and look as that of the template. The size with orientation of the page, specific margins, face type, font style and line spacing, have to be specified [17]. This collection of specifications that determine the appearance of a document is known as template. User can change structure, style and look of the document depending upon the requirement.

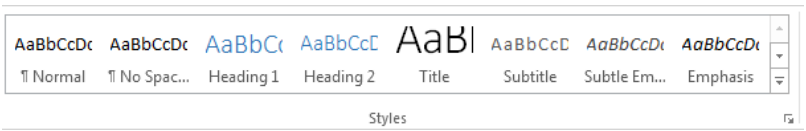
Examples: Examples are resumes, invitations, and newsletters.

Theme

It is used to provide a designer look with different theme colors and fonts. Themes can be shared among the Office for various applications that support themes, such as Excel, Word and PowerPoint. For example, we can create or customize a theme in either Excel, MS-Word PowerPoint, and then apply it anywhere.

Style

It is one of the important feature of the Microsoft Word. Style is basically the predefined instructions used for formatting throughout the document. The style gallery is available in the standard toolbar under the home menu as shown below.



When we create a document in the Microsoft Word, the new blank document uses the Normal template and the written text will use the Normal style. The typed text in the newly created document uses the font type, font size, indentation, line spacing, paragraph spacing, text alignment and other specifications defined under the Normal style.

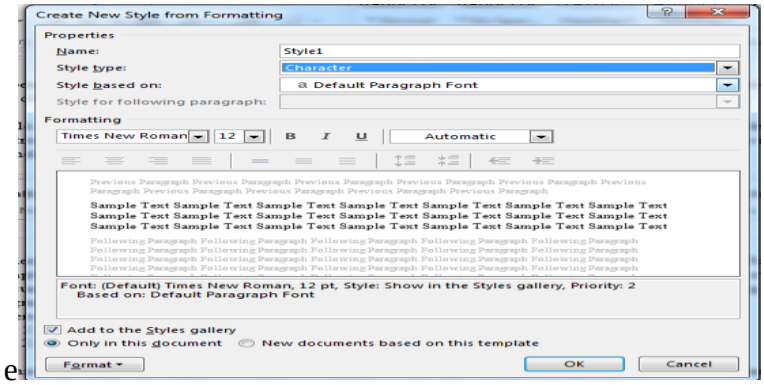
There are two types of styles.

- Character style

- Paragraph style

Character Style

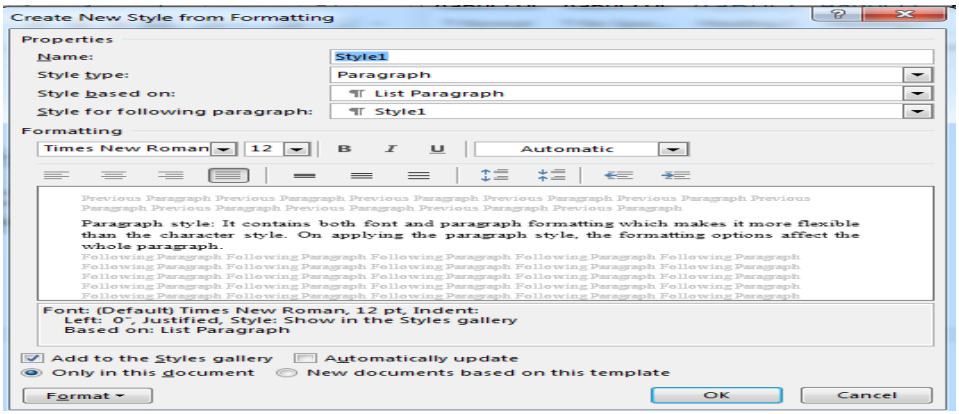
It is applied to typed words or even individual characters. Character formatting is done from the formatting options available under the create new style from formatting option. The style type chosen is character.



e

Paragraph Style

It contains both font and paragraph formatting which makes it more flexible than the character style. On applying the paragraph style, the formatting options affect the whole paragraph.



After Microsoft Word 2007 version, there is a provision of Linked styles which can be used for character formatting or paragraph formatting. When the formatting options are used on the text written in the particular paragraph, the linked styles act as character style. When formatting options are used on the paragraph, they act as paragraph style.

We can disable this option, by checking the 'Disable Linked Styles' option.

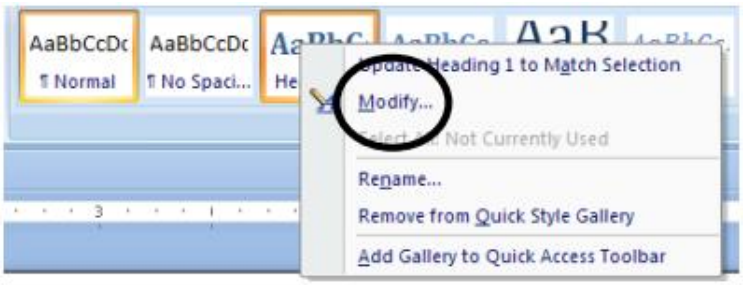
Steps to use and create the style

- Click the paragraph, Word, list, or table you want to format [3].
- Select the style that we want want to apply from the Styles group

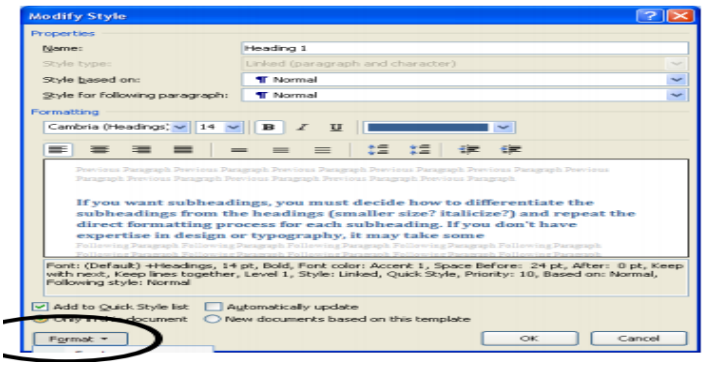
Steps to modify the style

For quickly modifying all the text that is formatted with a particular style, you can reformat the style just by changing its properties.

- Right Click on the styles option in the Styles group
- Click on Modify option



- Do the modifications by using either the Format button or icons

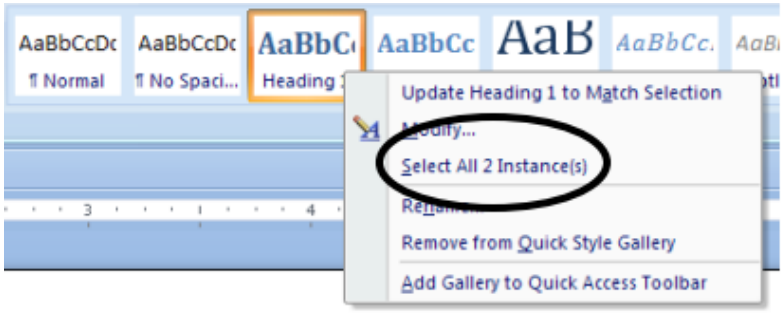


- Click on OK

To Select all Same Formatting using Styles,

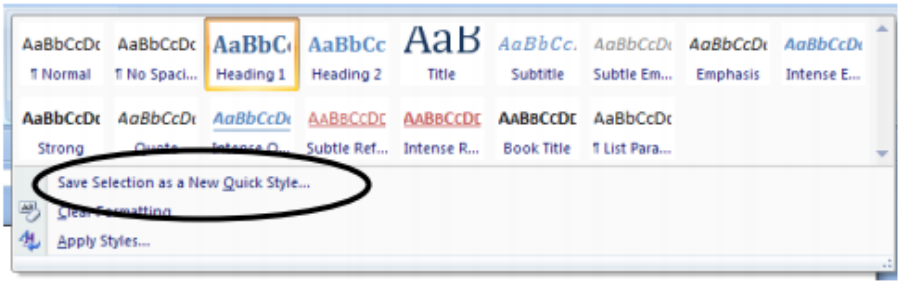
We can quickly see all the areas of our document that have been formatted with a selected style, it will highlight all the areas that are formatted using style.

- Right-click on the Style option
- Click on Select All 2 Instances

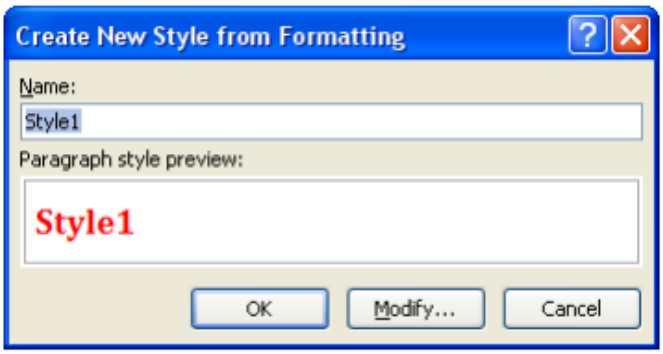


Steps to Create a New Style

- Highlight the part of the document to format
- Using the Font and Paragraph groups apply all formatting
- Click on the drop-down list in the Styles group
- Click Save Selection as a New Quick Style



- Type the name of the new style and click on OK.

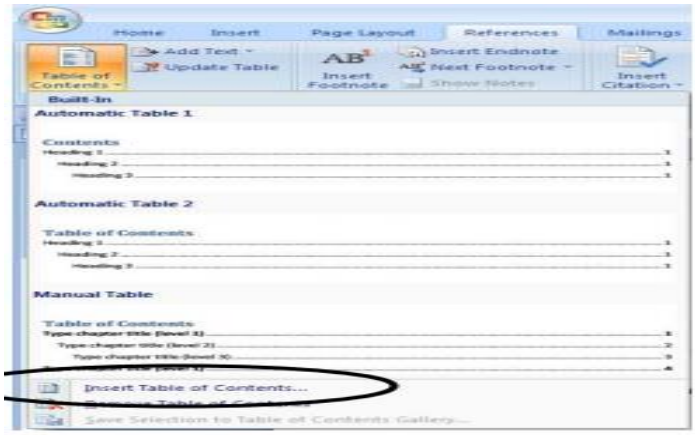


4.4 CREATE A TABLE OF CONTENTS

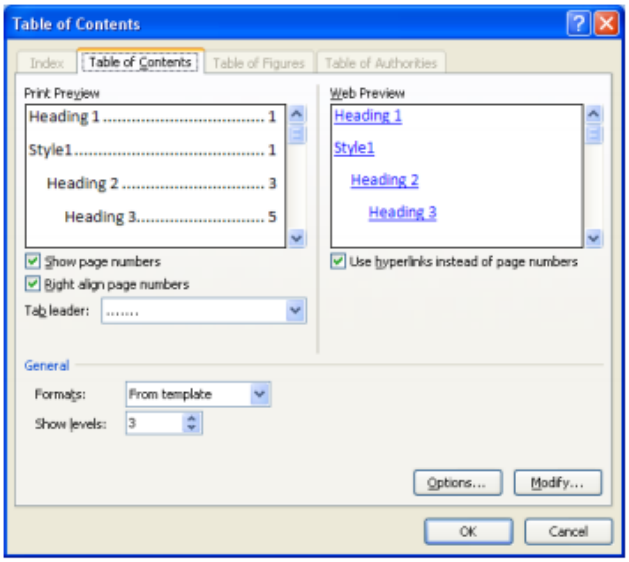
To get the overview of the topics in a document, Table of Contents is used to create using the heading styles from the Styles group.

- Apply the styles of headings to the areas of the document that are to be included in the Table of Contents [5].
- Use Heading 1 as the main Heading, Heading 2 for subtopics
- Select the References tab
- Select the Table of Contents

- Select either one of the built-in table of contents styles or click on Insert Table of Contents for a list of options

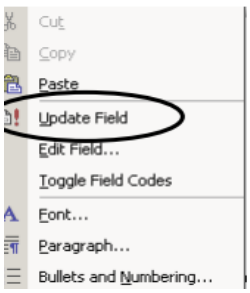


- Apply Changes
- Click on OK



4.4.1 Update the Table of Contents

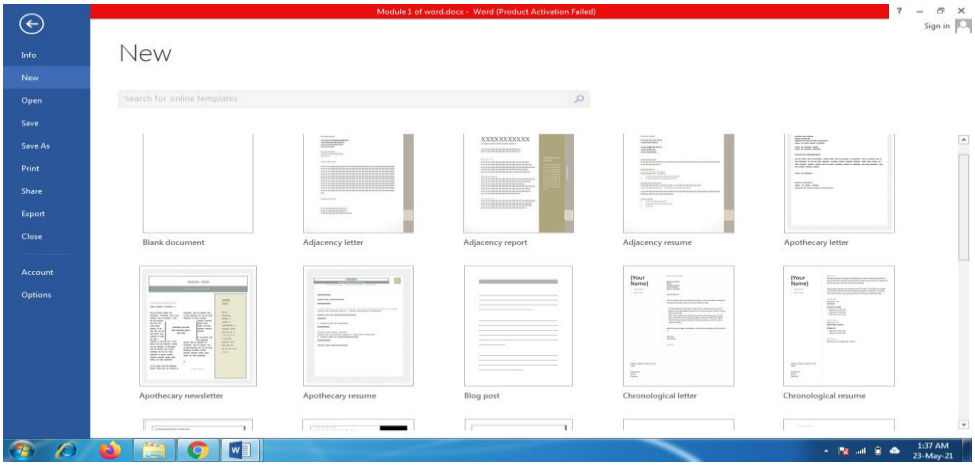
- If you want to modify the document, Table of contents need to be modified using the following steps
- Right-click where you need to update within the Table of Contents [3]
- Select Update Field
- Click on Update Entire Table
- Click OK



4.4.2 Using Template to Create a Document

There are number of existing templates available for the newly created word document. To select the already existing templates, following steps are used:

1. Choose the File tab
2. Click on the new option which will display all the existing templates.

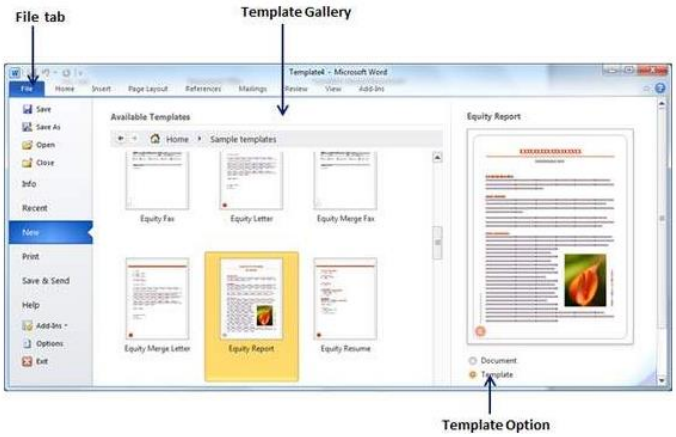


Microsoft Word provides a number of templates which one can use but it also provide the option to search for the templates online from the office.com.

4.4.3 Creating / Modifying a Template

Depending on the requirement of the user, the user can create his own new template. The template file has .dotx extension. The following steps are used to create the new template.

1. Select the File tab.
2. Click on the New option which display all the existing templates
3. Select any of the existing templates and open with template option as 'On'.



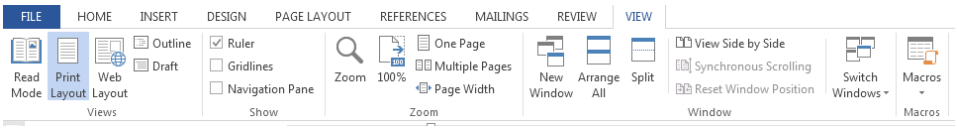
The user can modify the opened template as per the requirement and save it with .dotx extension.

Even, one can create the template from the created new document as well.

1. Click on File tab.
2. Click on New option where existing available templates will open.
3. Double click the Blank Document to create the new empty document template.
4. Save the template with the .dotx extension and a unique name.

4.5 DOCUMENT VIEWS

Depending upon the different aspects of the usage of the word document, Microsoft Word provides different views of the document. Instead of the default view, the user can find other views available to make one more productive. By default the word document opens in Print Layout, but other views can be selected by clicking the View tab.



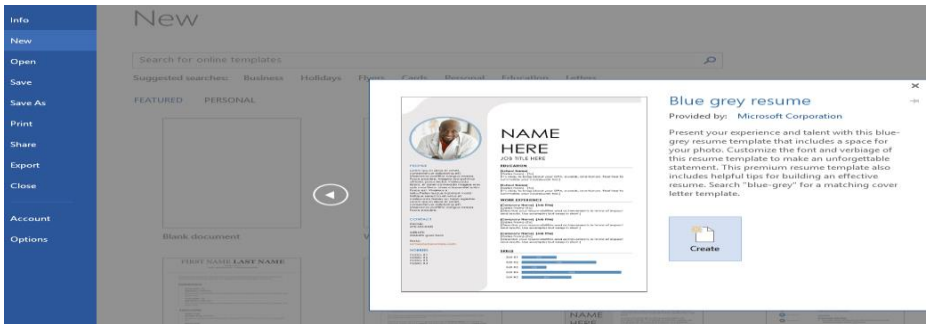
The other available view modes except the default 'Print Layout' are Web Layout, Read Mode, Outline and Draft.

- The Web Layout is appropriate when one want to view the document in the form of a web page. It is mostly used when the user is designing the web page in Microsoft Word.
- The Outline view is used for the navigation of a lengthy document which shows the outline form of the document. The user can also decide the number of levels that should be shown.
- Draft view works similar to the plain text editor which shows only the text without any formatting and graphics.
- Read mode displays only the pages of the document by hiding all the toolbars and menus which provides more space for the document text.

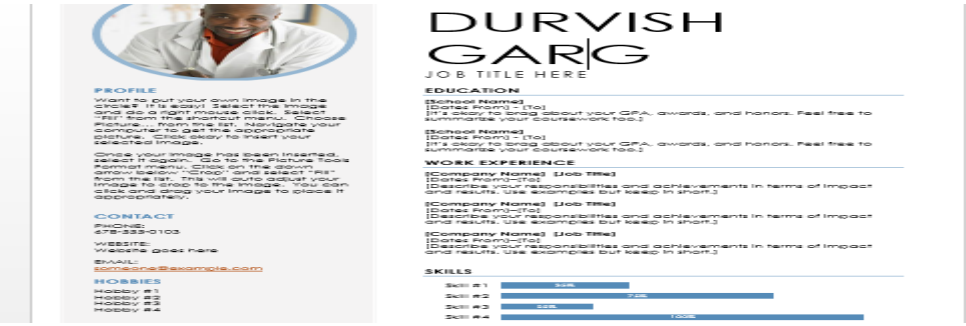
4.6 STEPS TO CREATE A RESUME

The following steps are used to create a Resume using MS-WORD

- Open MS WORD
- Click on CTRL+N or New from the File Menu
- Select the style of resume



- Click on Create
- Then Update the details and photographs as per the requiremnets



4.7 SUMMARY

- Basics of the Word Processor has been discussed
- Steps for Creating, closing, deleting and renaming the document file is also discussed
- Templates are the files that help to design interesting, attractive, and professional-looking documents.
- Theme is used to provide a designer look with different theme colors and fonts.
- **Style** is one of the important feature of the Microsoft Word. Style is basically the predefined instructions used for formatting throughout the document.
- Templates and Styles can be used for designing Resume, Invitation letter etc.

4.8 PRACTICE QUESTIONS

Q1. Differentiate between SAVE and SAVEAS
Q2. Design a template for typing the resume
Q3. Differentiate between Template, Style and Theme
Q4. Design the template for Invitation letter
Q5. What are the document views?

MCQ Type Questions

Q1. Which is the extension of templates [1]

a) Dotx
b) Doc
c) Doct
d) Dott

Q2. Which of the following statements regarding styles in MS Word are true[5]?

a) Styles can not to individual words or characters, only be applied to paragraphs,
b) All text in the document has assigned style even without assigning it
c) We cannot modify built-in styles
d) All of the above

Q3.Ali typed a letter to his son but did not apply a style. Which built-in style was assigned to the text in his letter?

a) Normal Style
b) Default Style

c) No Spacing Style
d) No style applied to the text

Q4. Which of the following statements about style deletion are true?
a) Both Custom and Built-in Styles can be deleted
b) Both Custom and Built-In Styles can be removed from the Styles Gallery
c) Styles cannot be deleted if they are assigned to text in the document, you first need to assign that text a new style
d) All of the above

Q5. Which of the following statements about the styles are false?
a) Styles help maintain consistent formatting within and between documents
b) Every aspect of text formatting can be specified for a style
c) Styles make it easier to change the formatting in large documents
d) Custom styles must be created as part of a template

Q6. The file extension _____ shows the file is a Word document.
a. .wor
b.. .wrd
c. .doc
d. None of these

Q7. How many number of different documents can be opened at the same time?
a. Maximum Three
b. One Only
c. As per the computer memory.
d. None of these

Q8 The _____ in the Resume Wizard dialog box shows the wizard is ready to be create the document [6]
a. Address panel
b. Start panel
c. Add or Sort Heading panel
d. Finish panel

Q9. _____is the default font size of a new Word document which is based on Normal template in Word 2007?
a. 12 pt
b. 11 pt
c. 14 pt
d. None of above

Q10. What do you call ‘a collection of character and paragraph formatting commands’?
a. defaults
b. template
c. documnet
d. a boilerplate

B.Sc.(DATA SCIENCE)

SEMESTER-I

FUNDAMENTAL OF IT

UNIT V: WORKING WITH TEXT

STRUCTURE

5.0 OBJECTIVES

To Edit and Format Text/Paragraph and Page

- To add comments in the document
- To insert the Table of Contents in the file
- To write the bibliography for writing content or research papers
- To track the document views

5.1 EDITING AND FORMATTING A DOCUMENT

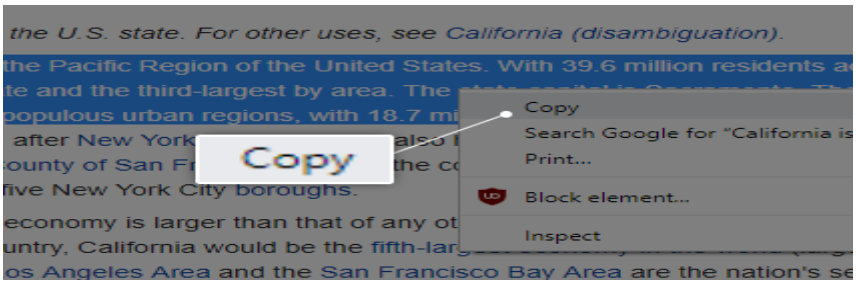
Editing means doing modifications in the document according to the requirements of the user. It is used for better look to your documents. You can select the text either by using Keyboard or mouse clicking.

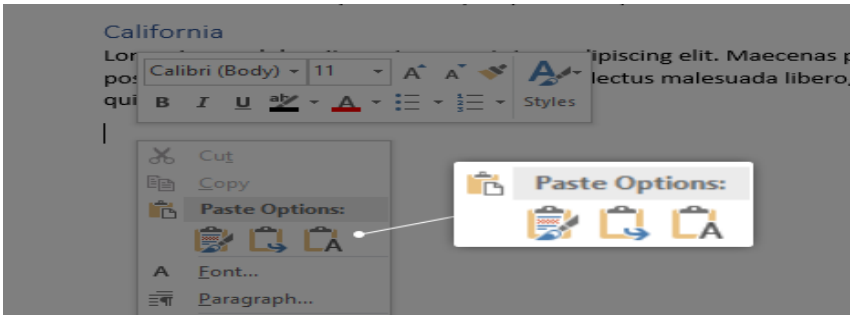
In this module, firstly editing of the text is discussed, then formatting through various options.

5.1.1 Select, Copy and Paste Text in Word

It is the important feature of the MS-WORD in which the part of the document can be available multiple times at the required location. The copy part of the content may be present in the same location and to other location. To perform a Copy, Paste the following steps need to follow:

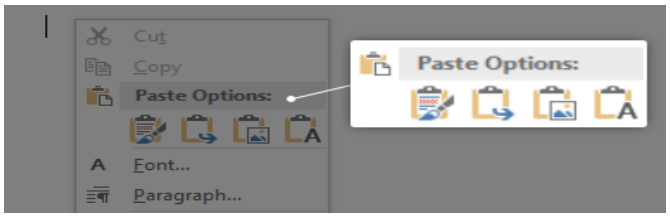
- Select the text that you need to copy.
- You can copy the text in one of two ways:
    - Right-click on the selected text, then select the **Copy** option. Or you can select the Copy option from Edit menu from the menu bar.





    - **Ctrl + C** Key can be used to copy also as a shortcut on your keyboard.
- Paste the text inside your document in a number of ways[2]:

- Put the cursor where you want to paste and right click onto it, It may be accessed the **Home** tab in the Ribbon. You can select any of the formatting like "**Source formatting", "merge formatting" or keep text only".**
- Use the **Ctrl + V** shortcut on your keyboard for pasting the text.

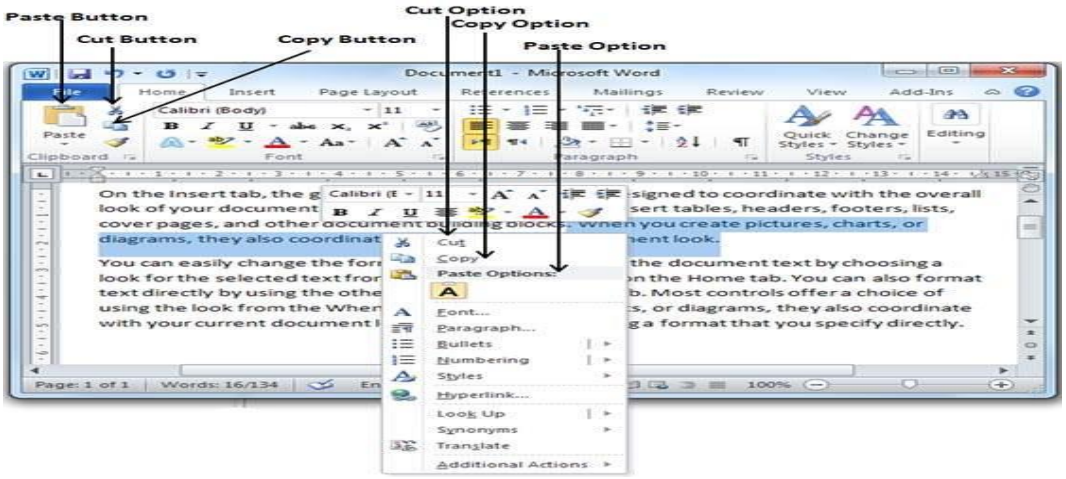


- The text you copied is now in your document!

5.1.2 Cut & Paste the Text

The Cut operation is used to remove the content from its original location and made available from its original location to a desired location. It can be used to move to the same document or to any other document. These are the following steps:

- Select a portion of the text which we want to cut
- Multiple options can be used to cut the content
- Using Right-Click – By pressing right-click on the selected text, cut option will be displayed and click on the option.
- Using Ribbon Cut Button is also available at the ribbon to cut the selected content
- Using Ctrl + X shortcut Keys to cut the selected text.

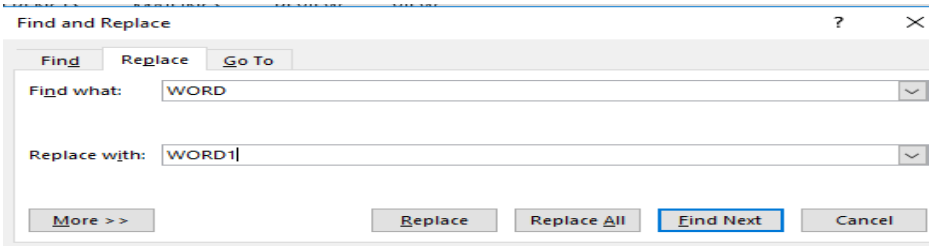


- Using Ctrl + V keys is used to paste the content at the desired location.

5.1.3 Find and Replace in the Word

This option is used to find a required word and replace a word with other word. To perform Find and Replace these are the steps:

- Click Edit Menu and select the option replace from the drop down menu, either press Ctrl+H.



- Type the word or text that you want to find and enter desired text in the Replace box.
- To update in all the places at once, choose Replace All.

5.1.4 Inserting Special Symbols and Characters

MS-WORD contains all the alphabets, numbers and some other symbols which are available on the keyboard. Sometimes, you need to insert some special symbols that are not available on the keyboard like some mathematical formulas, scientific equations etc.

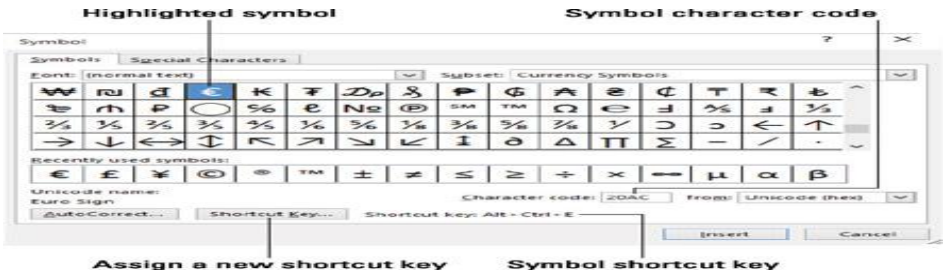
These are the following steps:

To insert a special symbol:

- Put the cursor where you need to insert a symbol
- From the Insert tab , select ‘Symbol’[14].
- Choose the symbol from the drop-down list.
- If the symbol is not in the current list, Select More Symbols. From the font box, select the font that you need to use and select Insert.

To insert a special character:

- Click on the Insert tab, choose the Special Characters tab.
- Select the character that you need to insert, and then select Insert.



5.1.5 Set Tabs and Indenting

Tab stops can be sued to create uniformly spaced text. Word has by default left tab stops set after every half-inch, but it can be created using own tab stops for a specific position.

5.1.5.1 Steps to Set the Tabs

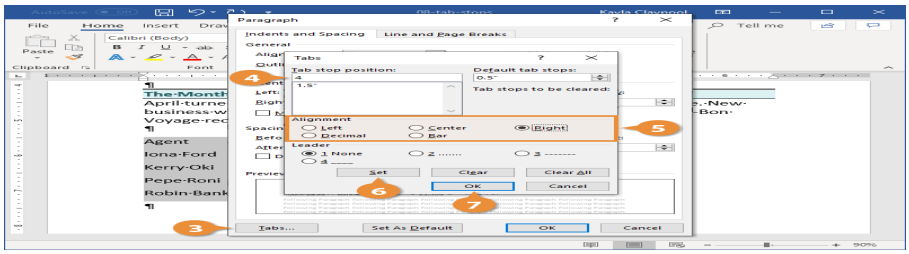
1. Select the **Show/Hide ¶** button from the Home tab.

2. Select the View tab.

3. Select the **Ruler** checkbox to Show the group.

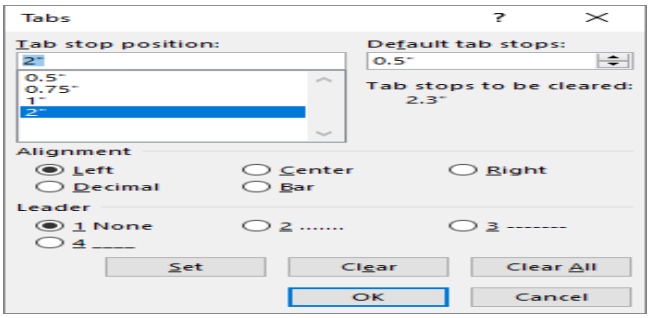
5.1.5.2 Steps to Set a Custom Tab Stop

- Select the **Home** tab.
- Select the **Paragraph** dialog box launcher.
- Select Tabs.
- Choose the type of tab stop which you need to set.
- Click **Set**.
- Click on **OK**.



5.1.5.3 Steps to Remove a Tab Stop

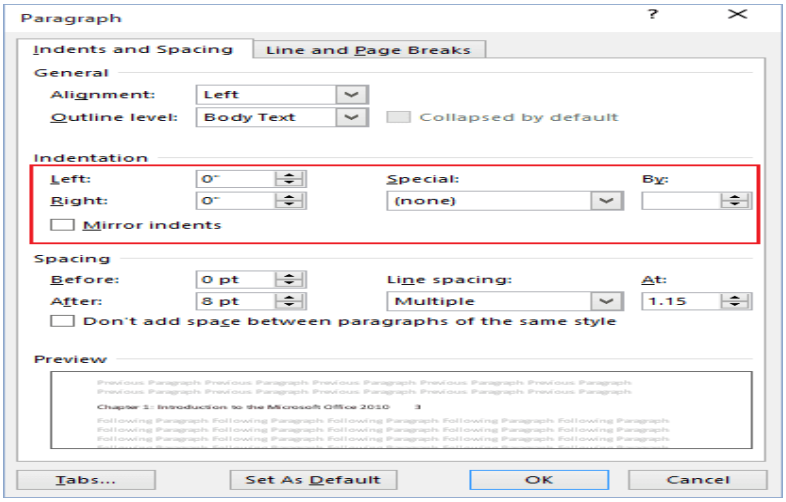
- Select the **Clear** button in the Tabs dialog box to delete a single tab stop
- Select the **Clear All** button to delete all tab stops.
- Alignment can also be done in the same way.



5.1.5.4 Indenting Text

It is used to provide the extra space to the paragraph or the text. The distance between the page margin and the boundaries of the Text is called an Indent and the process is known is as indentation. There are four types of indents such as left, right, hanging and first line indent

- Steps to indent the paragraph are:
- Select the paragraph or text which you need to be indented
- Select the Format menu and select the paragraph option
- Click the mouse on the indent and spacing tab option.
- Select any of the option to set left, right, hanging or first line indent.
- Click on OK.

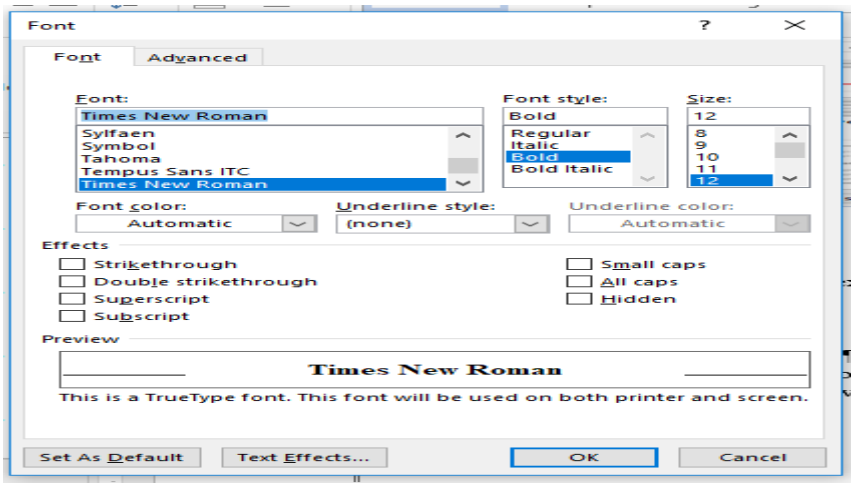


5.1.6 Formatting Text

Formatting text means to display the text in the better way using various font, font size, font styles and font colors.

Steps to Format the text

1. Select the text or the paragraph
2. Select the Format menu and drop down menu appears.
3. Select the Font option. A font window appears on the screen with different items
4. Font: Select the type of the Font like Times New Roman or any other.
5. Font Style: Select the Font Style like regular, bold, italic etc.
6. Select the Font Size any like 10,12,8 etc.
7. Select the Font color from the set of colors
8. Click OK

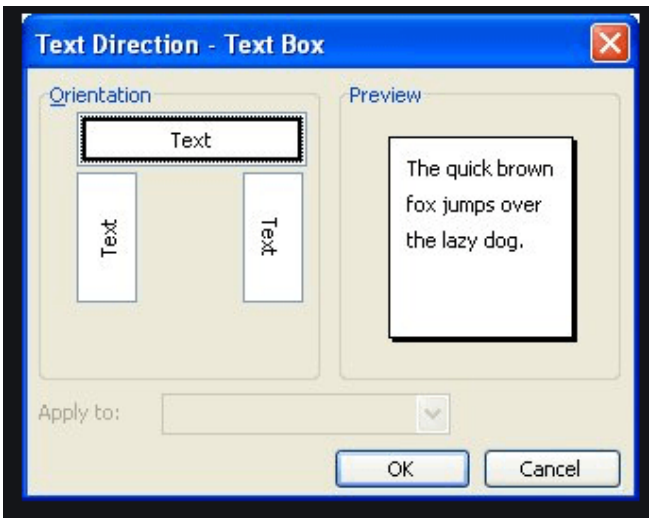


5.1.6.1 Setting Text Direction

Text direction can be changed from bottom to top or top to the bottom. This option is useful for printing name or headings in a envelope on A4 sheet

Steps to Set the Text Direction:

- Click on the Insert Tab and select the Text Box from the drop down list
- Then Text box will be available
- Enter the required Text in the text box
- Select the Format and select the Text Direction.
- Select the required Text direction
- Click on OK



5.1.6.2 Auto-Correct

Auto Correct option converts the large Strings to short form. These are the following steps

- Click on the Tools and Click on Auto Correct Option from the drop down menu'
- Enter the short form MS-WORD like MW
- Click on add and ok

5.1.6.3 Bullets and Numbering

Graphical symbols bullets can be used to represent each line and Numbering can be used to represent the items into numbers and alphabets. These are the steps to insert bullets and numbering

- Set the cursor where you need to use bullets/numbering
- Select any of the bullet button from formatting tool bar
- Press the enter key or Okay

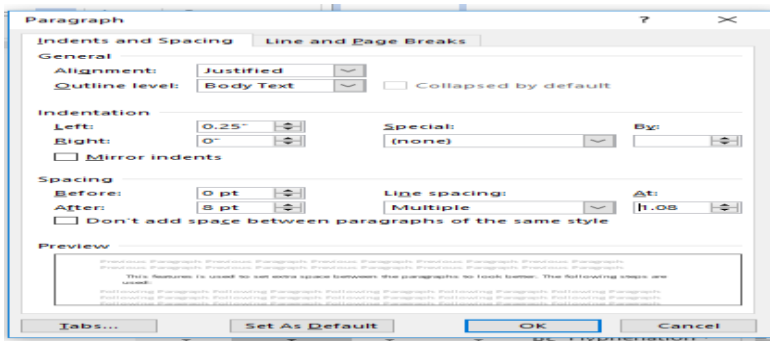
5.1.7 Formatting Paragraphs

MS-WORD provides many features for formatting whole paragraph in your document. The following features are being discussed.

5.1.7.1 Paragraph Spacing

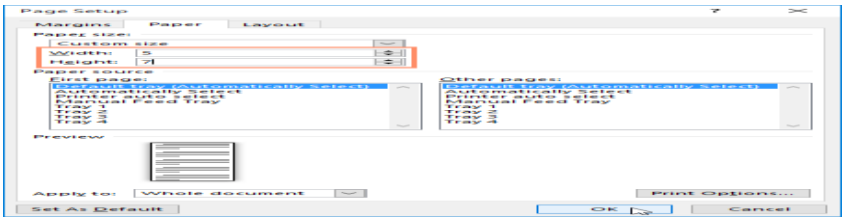
This features is used to set extra space between the paragraphs to look better. The following steps are used:

- Choose the paragraph which you need to format
- Click on the Format me menu and select the paragraph option



5.1.7.2 Page Setting

- Click on the File option and select the page setup
- Select the Margin, Paper or layout option

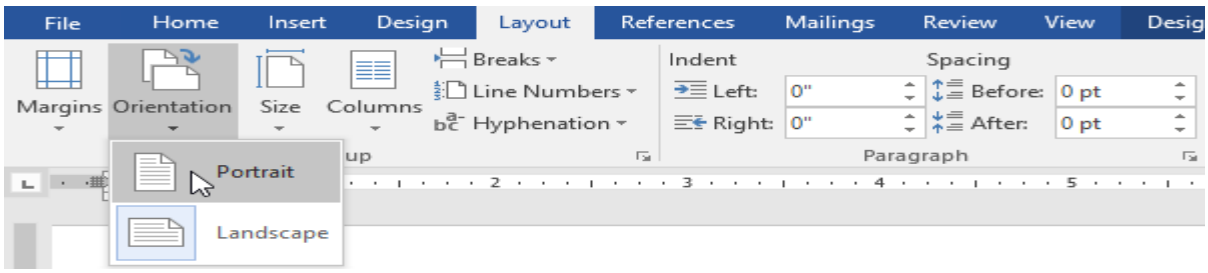


5.1.7.3 Page Layout

Sometimes you need to take a print on landscape, then the page layout can be used for taking the page in portrait (length wise) or landscape (width wise)

- Click on orientation tab
- Select the orientation either portrait or landscape

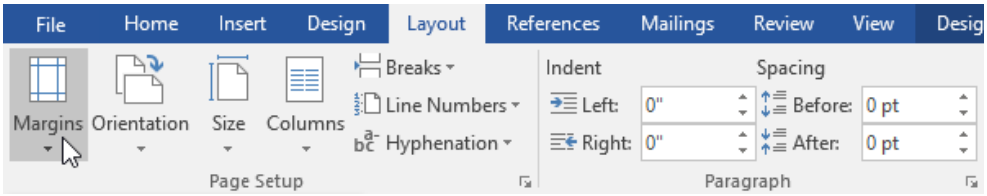
- Landscape means the page is oriented **horizontally**.



5.1.7.4 Page Margins

Page margin is the difference of the space between text and the edge of the document. The default value of Page margins is Normal style with one-inch space between a text and each of the edge.

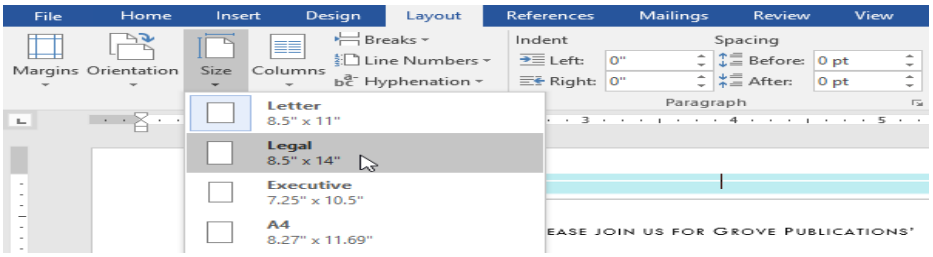
- Select the Page Layout tab, then choose the Margins command.
- Select the defined margin from a list of drop-down menu.
- The margins of the document will be modified.



5.1.7.5 Page Size

The default page size of an active document is 8.5 inches by 11 inches. If you need to change the page size, then you can modify using following steps:

- Click on the paper tab from the page set up dialog box
- You can change the paper size as per the need.



5.1.7.6 Page Break

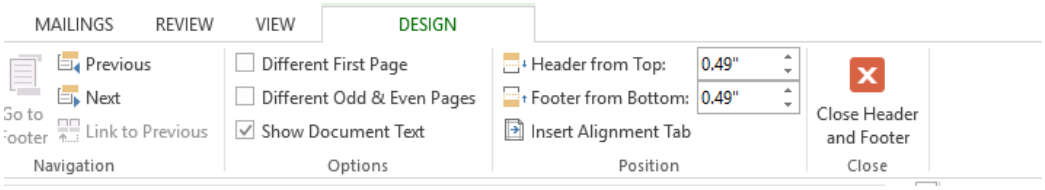
When you want to start a new page when current page is still not used fully, then Page Break allow to go to the next page.

- Put the pointer where you want to set the page break
- Click on the INSERT menu and select break
- Select the Page break from the or you can press Ctrl+ENTER

5.1.7.7 Creating Headers and Footers

It is used to create a heading o on the top of the page and some message at the bottom of the page. It is mainly used for setting heading of the chapter and on the footer, date or page number may be mentioned.

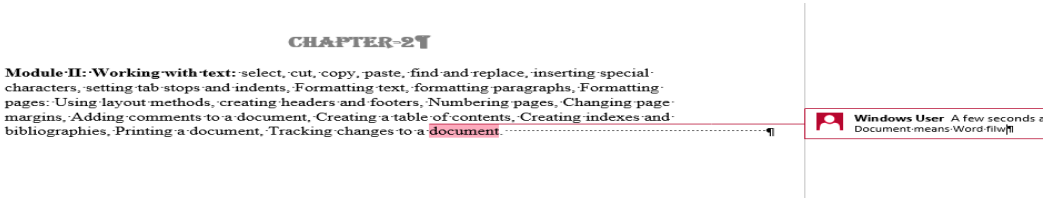
- Select the view option from drop down menu
- Select the header and footer option
- Enter the text in the header and in the footer set the date
- Heading will be appeared on the top and date will be onto the bottom.



5.1.7.8 Adding Comments to a Document

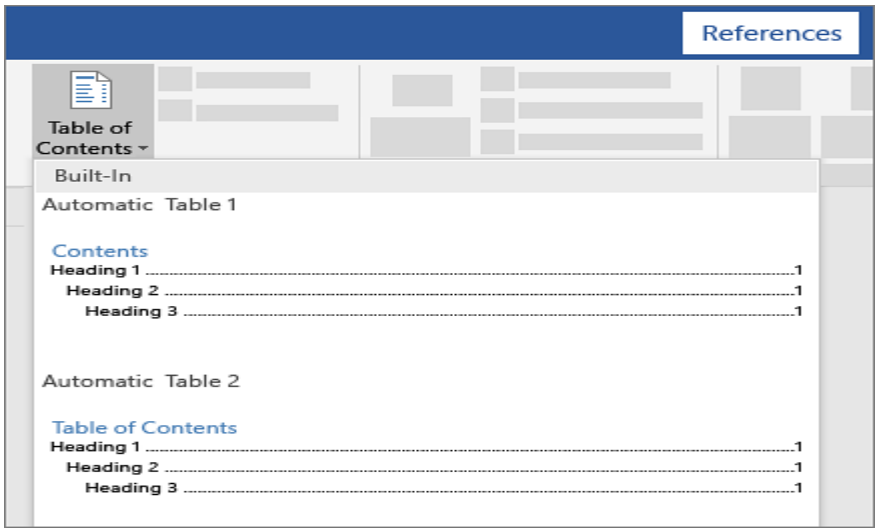
- Click the text where want to insert a comment.
- Choose on the **Review** tab and click on New Comment.

- Type your comment and word displays the comment in the document's margin.

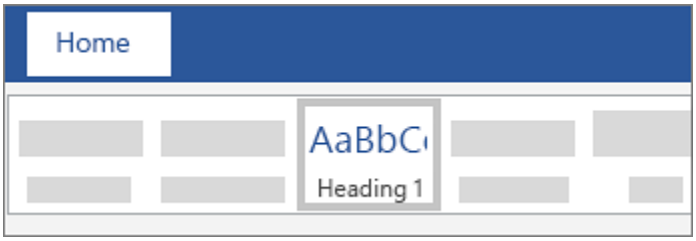


5.1.8 Create the Table of Contents

- Put the cursor where you need to add the table of contents.
- Click on the References and select Table of Contents option
- Choose an automatic style.



- If you want to modify the content that also effects the table of contents,
- Then, update the table of contents by right-clicking on the table of contents and select Update Field.
- For taking each heading the table of contents, select the heading text.
- Select Home then Styles option and then select Heading 1.



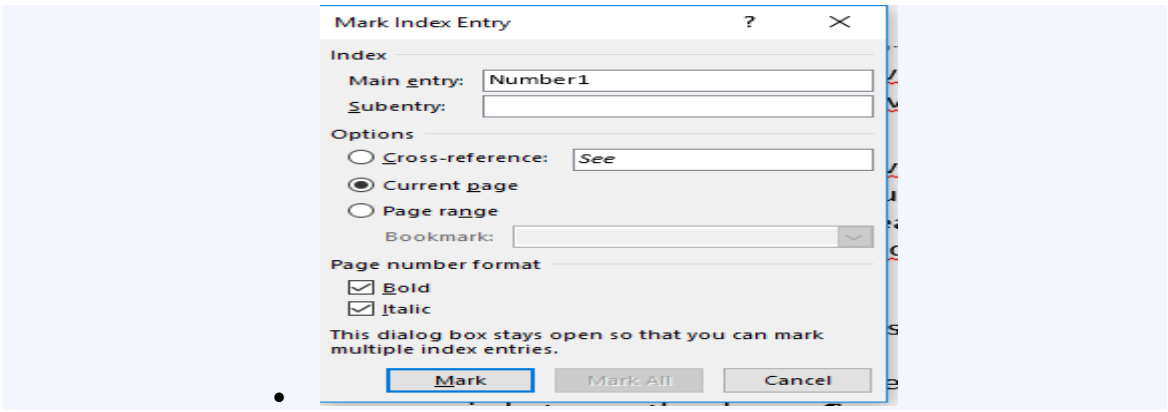
5.2 MODIFY YOUR TABLE OF CONTENTS

- Choose the text which you need to modify with table of contents.
- Click where you want to insert the entry
- .Select the References tab, in the Index group, click Mark Entry.

5.2.1 Create Indexes

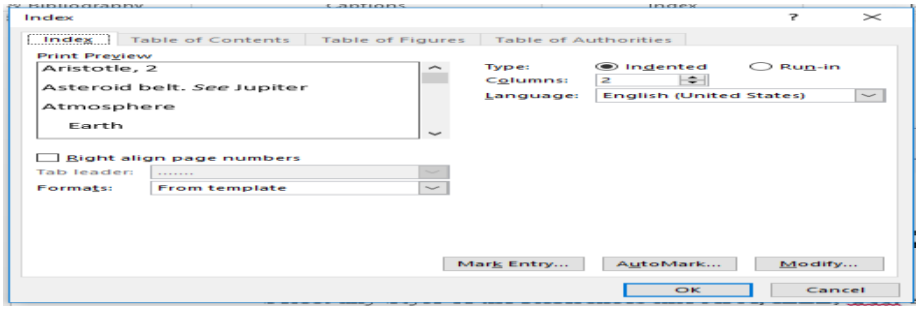
- Place the cursor where you want to create an index
- Go to References and select Mark Entry option
- Select any required formatting options from the menu

- Select Mark Entry Option
- Text can be edited in the Mark Entry option



  - Second level can be added the Subentry box.
  - Select Cross-reference tender Options, and then type the text for the other entry in the box.
  - Page can be formatted by using formatting features like Bold/Italic
- To mark this text in whole document select Mark Al option
- Click where you need to add the index.
- On the References tab select Insert Index.

- Formatting can be done for text entries, page numbers, tabs, and leader characters.



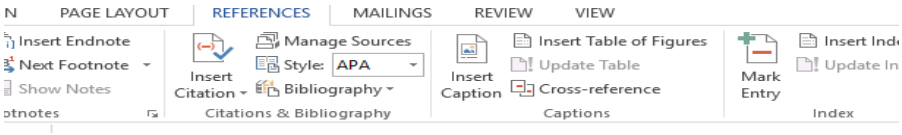
- Click **OK**.

5.2.2 Create Bibliography

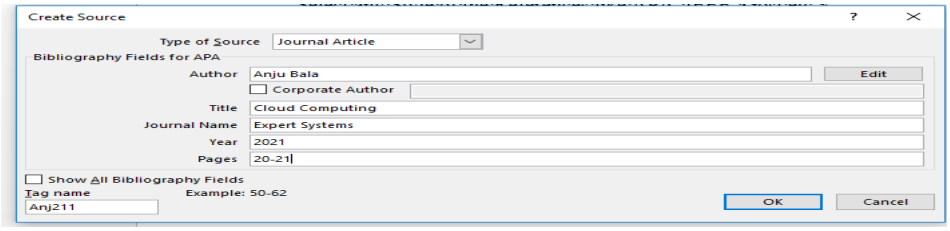
A bibliography means the list of references used in the document. The references can be taken in a bibliographic database or within the document itself [11].

Steps to Create and Update a Bibliography database

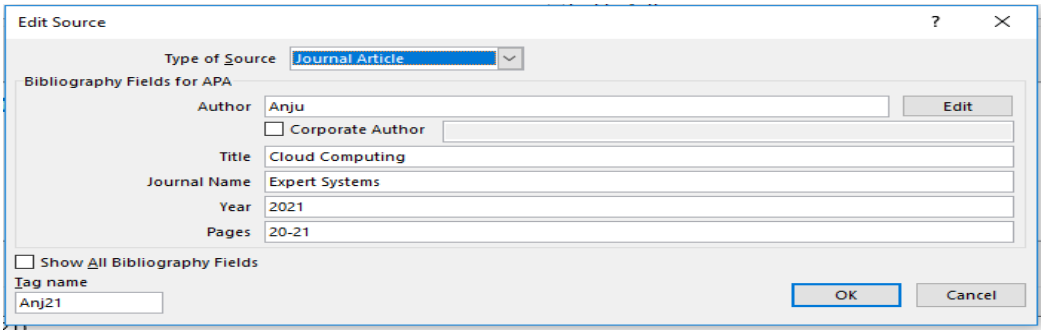
- Select the References option
- Select any Style of the References like APA, IEEE, Gost etc.



- Select the source for where you need to add references.



- Put the cursor at the end of the line to add citation
- Select Insert Citation option and select the source which you are citing. (Anju, 2021)
- Citation can be edited Go to the citation and add the references
- Click on OK.



5.2.3 Printing a document

When you want to take a hard copy of the document, then it is better to use print preview option, it gives the idea about formatting details before taking printout. You can modigy the document before taking print [10]. These are the steps to print a document.

- Make sure that the printer is on and ready to print.
- Save your document.
- Click the File tab.
- Firstly, select the print preview
- If you are satisfied with formatting, select the Print option or Ctrl+P command
- Specify the type of printer which is attached

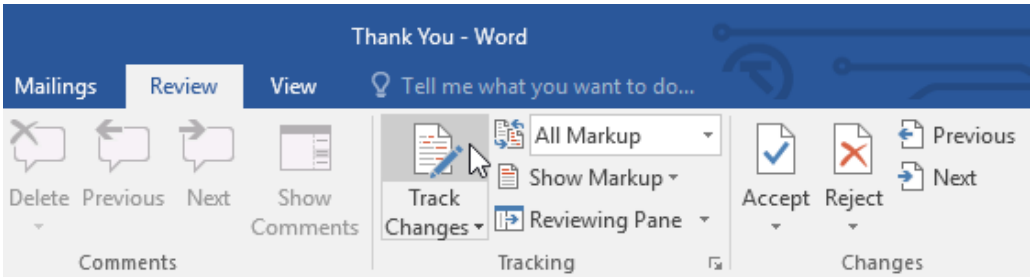
- Select the page range:
  - **All:** To print all the pages
  - **Current Page**: To print the current page

Pages: Number of pages you need to print

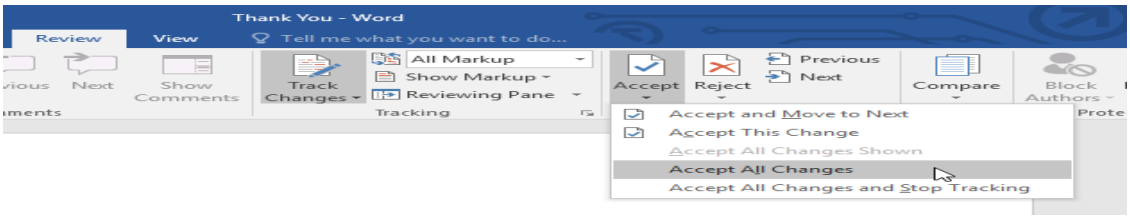
5.2.4 Track Changes in the Document

To turn on Track Changes:

- Select the Review tab
- click the Track Changes option



- Track Changes will be turned on.
- Any modifications you make to the document will be appeared as colored markups.
- The changes can be reviewed from where you can accept or reject the changes
- Select the change which you need to accept or reject
- Click the Accept drop-down arrow to select all the changes, select Accept
- If you do not want to continue, then select Stop Tracking.



5.3 SUMMARY

- Editing of the text or paragraph is possible by using CUT, COPY PASTE option or with shortcut keys

- Text Formatting can be done using various options like Auto Correct, Bullets and Numbering, Text Direction etc.
- Page Formatting can be done using Page Size, page break, Page Layout etc.
- For writing any book, chapter, table of contents can be created using Table of content option from the References Tab.
- Bibliography can be added using References Menu Bar, it would be useful to add citations in the paper or in the document.
- Print option is used to print any document. Ctrl+P shortcut key is also used to Print.
- The changes in the document is tracked by using Track changing option.

5.4 PRACTICE QUESTIONS

Q1. Differentiate between Cut Paste and Copy Paste

Q2. What is the importance of inserting Headers and Footers in the document?

Q3. Significance of Find and Replace

Q4. Create a Bibliography for any Research paper using IEEE style

Q5.Write various steps to create the Index?

Multiple Choice Questions

1. The space left between the start of a paragraph and Margin is called

a. Spacing
b. Indentation
c. Merging
d. None of these

2. To apply centre alignment to a paragraph which shortcut key can be used

a. Ctrl+E
b. Ctrl+A
c. Ctrl+B
d. Ctrl+N

3. Text Styling features in MSWORD is done by

a. Word Art
b. Word Color
c. Word Fill
d. Word Font

4. In which view Headers and Footers are visible

a. Print Layout
b. Page Layout
c. Normal View
d. None of these

5. For changing the line height to 1.5 we use shortcut key :

a. Ctrl+1B.
b. Ctrl + 2
c. Ctrl + 3D.
d. Ctrl + 5

6. We can insert a page number at

a. Header.
b. Footer
c. Both Header and Footer
d. None

7. _____can be used to change the thickness of a line.

a. Line Width
b. Line Height
c. Line Style
d. None of these

8. For selecting the Symbol dialog box, which menu is used?

a. Insert

b. Table

c. Format

d. Tools

9. Which is the default font size in MS-WORD

a. 12 pt

b. 8 pt

c. 6 pt

d. None of these

10. Which menu bar is used to add bibliography?

a. Insert
b. Home
c. References
d. None of these

B.Sc.(DATA SCIENCE)

SEMESTER-I

FUNDAMENTAL OF IT

UNIT VI: MAKING SMALL PRESENTATION

STRUCTURE

6.0 OBJECTIVES

- To know the basics of presentation software
- To Insert, Delete, Update the slides in a presentation
- To Add Clip Art and Pictures in the PowerPoint.
- To Set the timings for Slide Show

6.1 INTRODUCTION: BASICS OF POWER POINT

Power Point is an application program developed and distributed by Microsoft as a part of Microsoft office suit. It is very powerful, easy-to-use graphical presentation software that allows the user to create electronic slide show of presentations. It is widely used to show an important information and data in an organized manner. Power Point is used to display text, table, charts, graphics, audio and videos in the slides and it involves various tools like word processing, graphing and drawing etc. In this module, we will learn how to work with Microsoft Power Point and how to create exciting and interactive presentations [7][8].

First of all, you have to start the Microsoft Power Point from the start button of your windows as shown in Figure 6.1.

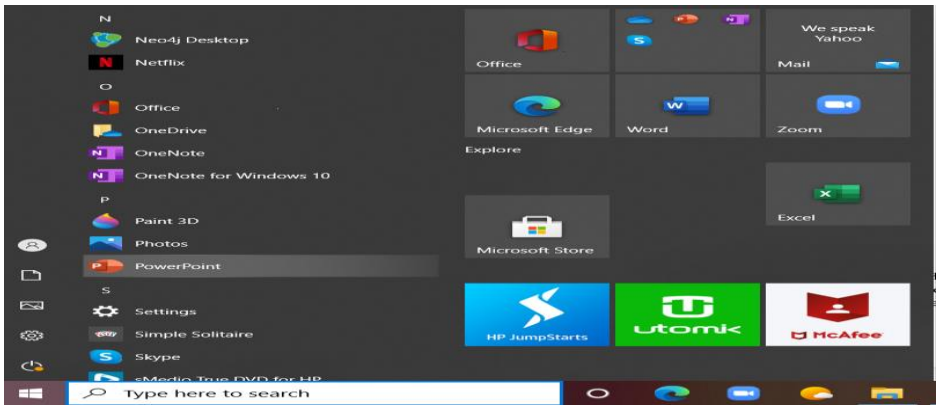


Figure: 6.1 Open Microsoft Power Point in Windows 10

6.1.1 Exploring the Parts of the Power Point Window

Power point window will appear as shown in Figure 3.2 at start-up and the various areas in a standard PowerPoint file are labelled. It provides the basic information of the graphical user interface of the window which is further helpful to the user to learn easily. The different labels of power point window are explained as follows: The different tabs for power point window is shown below:

- **File Tab**

This tab represents the backstage view that helps the user to create new file, open a file and print the presentation. The Save and Save as buttons are also user File Tab. Various design templates are shown in File Tab when user clicks on New button.

- **Ribbon**

The ribbon of power point window consists of the following components:

- Tabs: Tabs will be shown on the top of the ribbon along with the relevant command such as Home, Insert, Design, Layout and View.

- Groups: Groups are used to arrange commands that belong to same group on the basis of the function and the name of every group is displayed below the group on the Ribbon. For example, clipboard, font, paragraph, styles and editing are the names of groups displayed on the ribbon.
- Commands: Groups contain the related commands in the form of small icons

➢ **Title Bar**

It appears on the top part of the power point window. It displays a name of the file along with the name of the application program that is Microsoft PowerPoint. It also contains small button for save, undo, redo on the left corner and minimize, maximize and close buttons on the right corner of the title bar.

➢ **Quick Access Toolbar**

It appears just below to the ribbon in power point window. Quick Access Toolbar is used to place all the most frequently used commands inside it. It can be customized according to the requirement of the user.[4]

➢ **Slide**

It is the working area of power point presentation or the place where the information is represented of displayed. User can make the presentation by adding different layout or pictures, text boxes in this section of the window. It can be viewed as portrait or landscape as per the requirement.

Figure: 6.2 Power Point Presentation and parts of the window

➢ **Slide Pane**

Slide pane displays all the slides in sequence in the form of small icons for every slide. User can add or delete slides in this slide pane. The slides can also be rearranges here.

6.1.2 Creating Presentation

When Power Point window will open then by default a slide appears as shown in Figure 6.2. This slide has two placeholders or text boxes. Additional text boxes can be added from the Insert tab. To start creating presentation just click on “Click to add title” (title placeholder) or text box a blinking cursor will appear. Click once on “click to add subtitle” and add the subtitle of the slide or the other information that you want to present. You can also add table, image or graph in the subtitle box. But if you need to create more new presentation then follow the steps:

- Click the File tab to view new button that is available under the backstage view of File Tab, further click on new button then consequently window shown in Figure 6.3 will be displayed.
- The user can take any of the templates shown on the screen or can search for a specific template from search bar to find something more specific or click on the Blank Presentation.

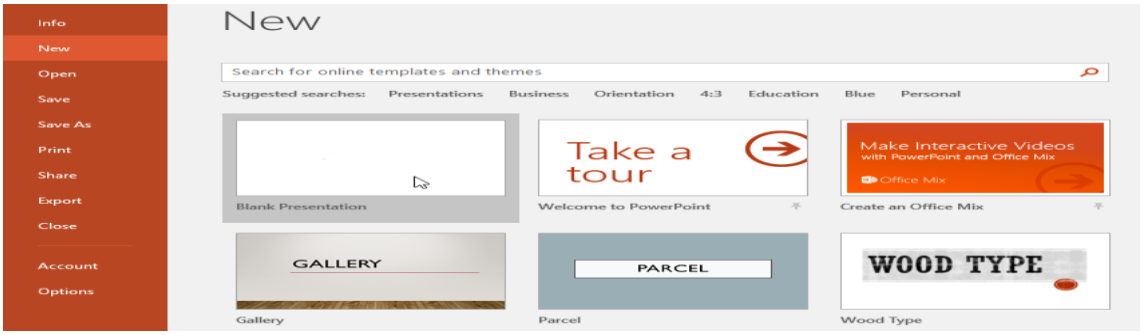


Figure: 6.3 Creating a New presentation in Power Point

6.1.3 Saving the Power Point Presentation

Power Point has two ways to save the presentation Save and Save as. These two options have similar operation but there is a significant difference also:

Save: When you create a presentation, then save command is used to save the changes which you have done. Save option is used to choose a file name and its location the first time. Then, click on the Save command to save it with the same name and same location.

Save As: This command is used to create a copy of the presentation at new location while keeping the original file as it is. By using Save As, You can select a different name and/or different location for the copied version.

Once you have finished the power point presentation and want to save it for future use then click on the File button in the menu bar. Then this Figure 6.4 will appear to you and click on the save or save as button and select the location from the given options such as computer, OneDrive or the other place by clicking on add a place.

Figure: 6.4 Saving a Power Point Presentation using Save As

Steps to Save a Presentation:

1. Save command is selected from the file menu or from the Quick Access Toolbar.
2. The dialog box will appear to you, where you can select the location to save the file along with the name of the file to be filled in the text box.
3. The Save As dialog box will displays as shown in Figure 6.5.
4. After clicking the save button the presentation will be saved.

5. The key combination of Ctrl+S will also perform the same function as save option.

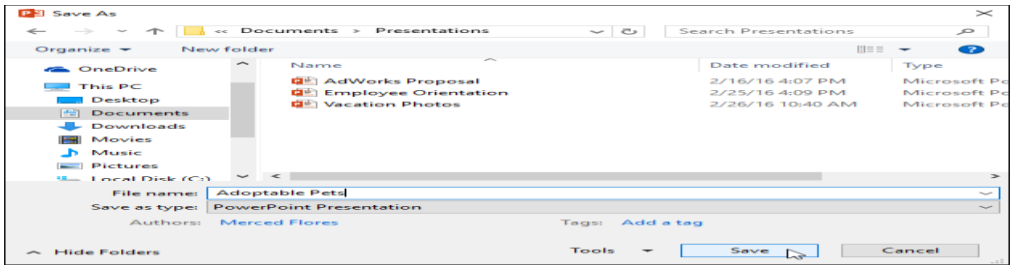


Figure: 6.5 Saving a presentation

6.1.4 Entering and Editing Text

Power Point allows users to enter text to the slide or to the text box also. The entered text can be arranged or displayed in desirable font, style, size and colour.

The new text can be added to the slide by clicking the tile box or subtitle box and then cursor will appear.

- The default text shown in the content box will automatically disappear.
- The added text initially follows the default formatting but latter the user can change the font or style.
- If you want to edit the text that has been entered previously then click in the text box or the placeholder box and change the text.

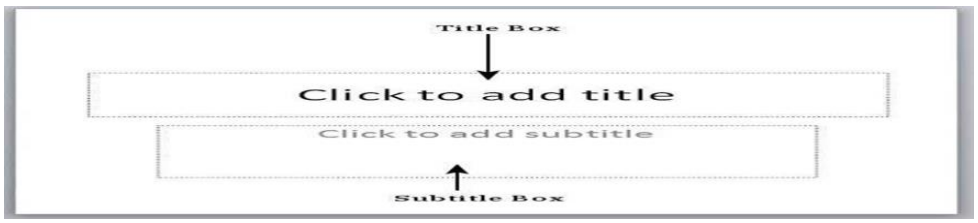


Figure: 6.6 Entering and Editing the Text

6.1.4.1 Font Formatting

It is the part of editing and entering text in a slide to make it more presentable. It can be modified using formatting toolbar. It includes different tasks:

- Font: It is used to change the style of the Font.
- Font Size: Font Size can be selected from the size box.
- Text appearance: It helps to change the appearance of text either in Bold, Italic or Underline etc.

Steps to edit the font setting using Format Menu.

- Select the text which you need to format
- Click the 'Format' menu from the Menu bar and select the font. The Font dialog box will be appeared
- Choose the appropriate option from the dialog box like font size, type of font, font style etc.
- Click on OK to obtain the result after formatting

6.1.4.2 Change Case

It is also used for editing the text in case of changing the case of letters either Capital to Small or Small to Capital.

Steps to change the Case

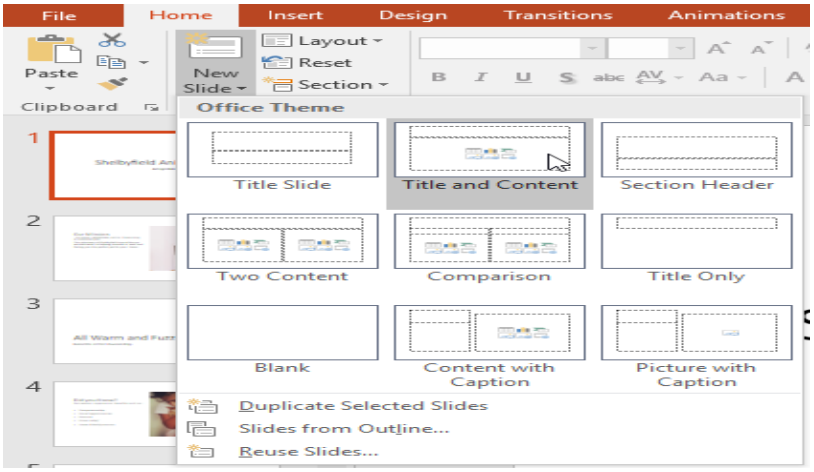
- Select the Format from Menu bar
- Select Change Case from the drop down Menu
- Select appropriate case from the options like Sentence Case, Lowercase, Uppercase, Title Case and Toggle Case.
- Click on OK.

6.1.4.3 Inserting and Deleting Slides in a Presentation

By default, the presentation contains only one slide at the beginning. The user can insert any number slides as per the requirement. The following steps are taken to add a new slide in power point:

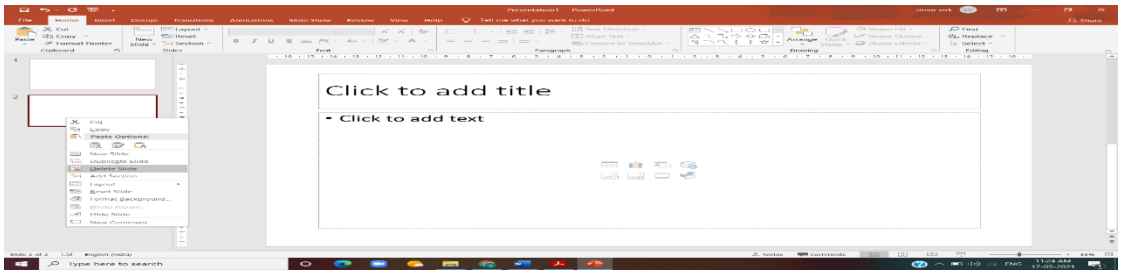
- Firstly, click on the Home tab, then further click on the small arrow on the New Slide command from the ribbon.

- The power point will ask you to choose the slide layout from the shown layouts and choose the slide as per the requirement.



- After selecting a slide layout then a new slide will be shown as below. Click any placeholder and enter the new text.

Delete slides: To delete a slide from your presentation if the slide is no more required. You have to choose the slide from the slide pane appearing on the left side the power point window, then press the Backspace key or Delete on your keyboard to delete the slide.

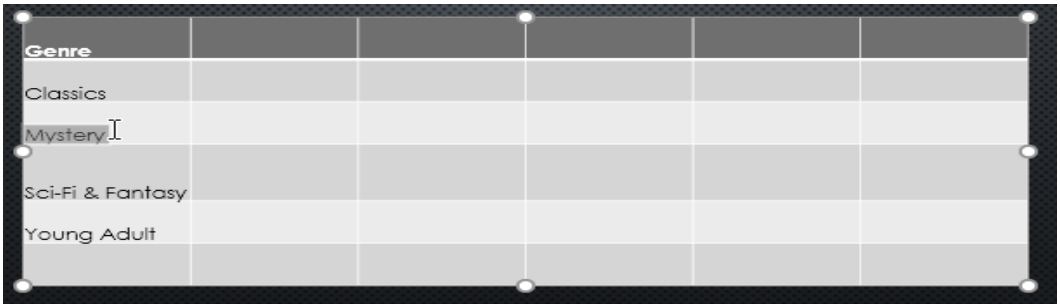


6.2 INSERTING WORD TABLE

A table is a collection of cells organized in the form of rows and columns. The tables are used for variety of tasks for presenting textual information and numerical data. To insert a table in power point presentation, follow the steps:

- Click on the Insert tab and then choose the Table command.
- Select the desirable number of rows and column that you want to take in a table. The example below is showing a table with six rows and six columns (6x6) is inserted:

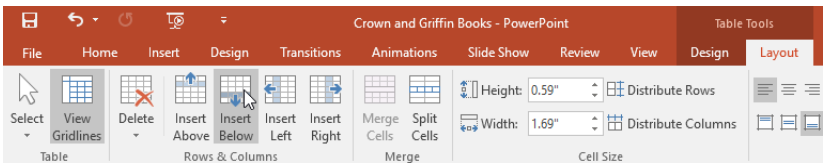
- The table is created now and will be displayed on the current slide.
- Click inside any of the cell in the table and add text to it.



6.2.1 Add a Row or Column to a Table

A new row or column can be added to table once it is created. The following are the steps to insert new row or columns:

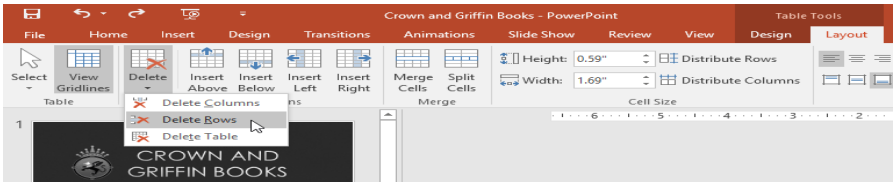
- Click a cell adjacent to which a new row or column is required.
- Click the Layout tab or right click in the cell.
- Search the Rows & Columns group from the ribbon. Then, select the options from given in the ribbon like to insert a new row, select any of the option either Insert Below or Insert Above. To insert a new column, select any of the option like Insert Left or Insert Right.



- After that a new row or column will be added to the table

6.2.2 To Delete a Row or Column from a Table

- Any row or column can be deleted. Choose the blank row at the bottom of the table as shown in the figure below.
- Click Layout tab under the Rows and Columns group on the ribbon, click the Delete command, which will ask you to select delete columns, delete rows or delete table options.

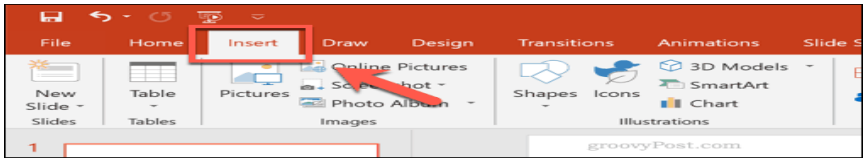


- The select row or column is deleted after selecting the delete option from the menu.

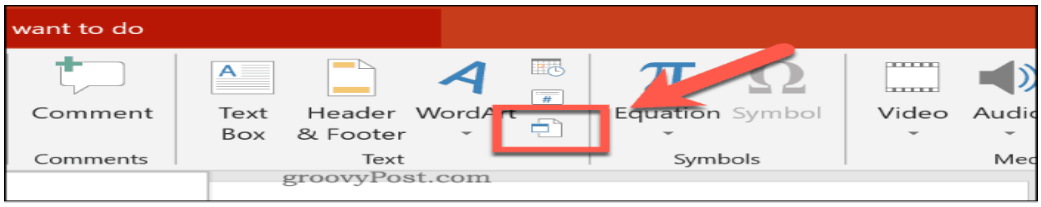
6.3 Inserting Spreadsheet Worksheet into Power Point

Using the **Insert Object** tool, insert data from your Excel spreadsheet as an object. This will add the contents of the most recently accessed worksheet into PowerPoint to view. To insert Excel spreadsheet, follow the steps:

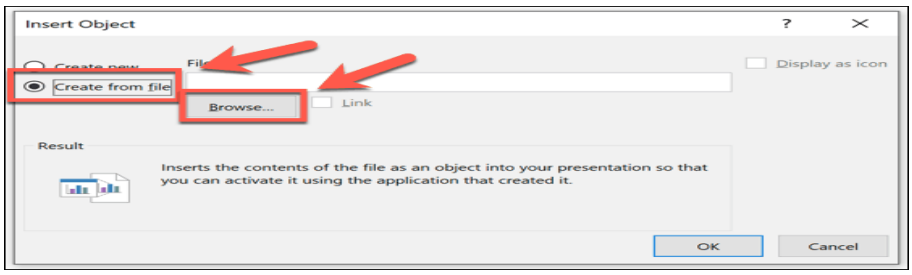
- To start, open your PowerPoint presentation and press the **Insert** tab on the ribbon bar.



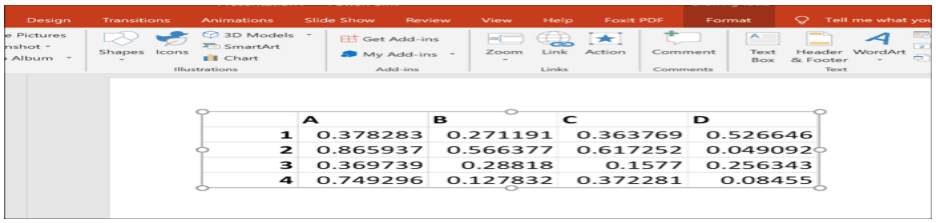
- To insert your Excel data, click the Object button. This may appear as a large or small icon, depending on your current screen resolution and the size of the PowerPoint window.



- This will open an insert object dialogue box. To add your Excel data, press the Create from file radio button, then press Browse to find and select your Excel spreadsheet.



- To add your data to PowerPoint, press the OK button. The data will be inserted as an object onto your PowerPoint slide, which you can then resize and manipulate.

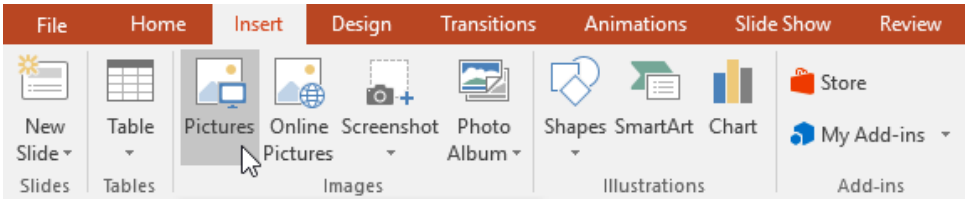


- By double clicking the table any required can be made to the spreadsheet data.

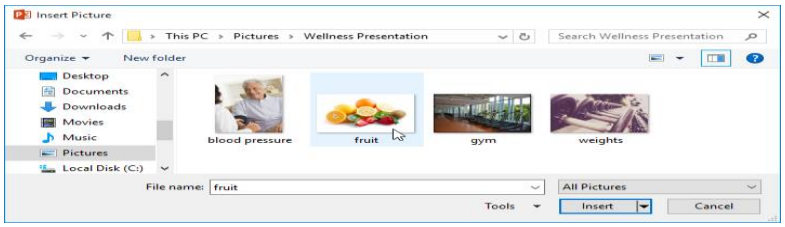
6.3.1 Adding Pictures and Other Objects

Objects are any element that can be added in PowerPoint. A text label is an object. An image is an object. Graphs and charts are objects. Any element within a slideshow is an object. Since any addition to a slide is considered an object, there are numerous options in the "Insert" tab.

- Click on the Insert tab, which will display all insert options such as Pictures, online pictures, screenshot and photo album.



- After clicking on the picture button, a dialog box will be display as shown below. Then choose the desired picture and press Insert button.



- The selected picture will appear on the current slide.

6.3.2 Inserting Video Clips

Video Clips add the liveliness to the presentation. It allows to insert a video into the slide and can be played it during presentation.

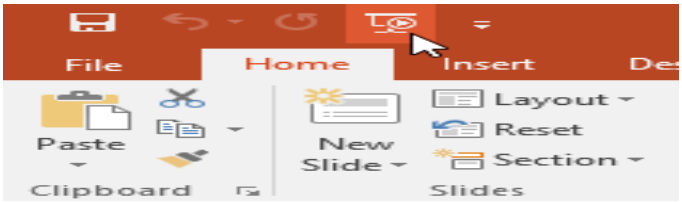
Steps to insert a Video Clip:

- Select the Insert Tab and Click on the Video drop down arrow from the Media Group and Select Video on My PC option.
- The **Insert Video** dialog box will be used to locate and select the desired video file, then click on the insert option.
- **Format and Playback** tabs under Video Tools can be used to **Insert** a Video by clicking on the Format Tab.
- Click on the Play button present at the extreme left of the ribbon.

6.3.3 Running a Slide Show

Once the presentation is finished then it is ready to run and slide show to its audiences. To run the slide show, follow the steps:

- Select the Start button icon from the Quick Access Toolbar and slide show of the presentation will appear or other way to run slide show is from your keyboard by pressing the F5 key from the function keys available on the top of the keyboard.



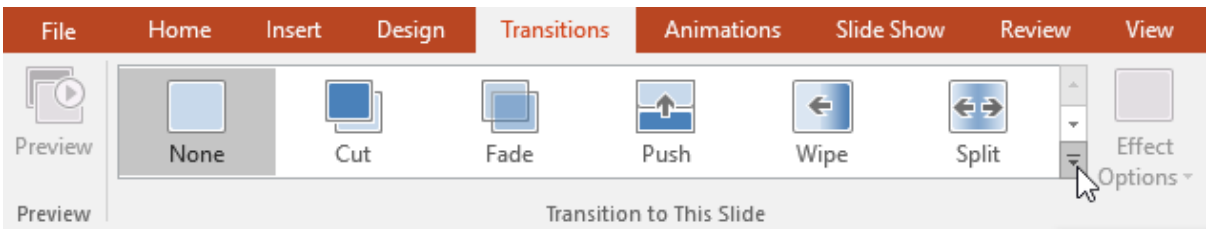
- The slide show will provide you full-screen mode of your presentation.
- By clicking the mouse or pressing the spacebar from the keyboard will help to move to next slide.
- The arrow keys of the keyboard help to move the slide forward or backward for the presentation.
- To exit presentation mode the Esc key, need to be pressed from the keyboard.

6.3.4 Transition and Slide Timings

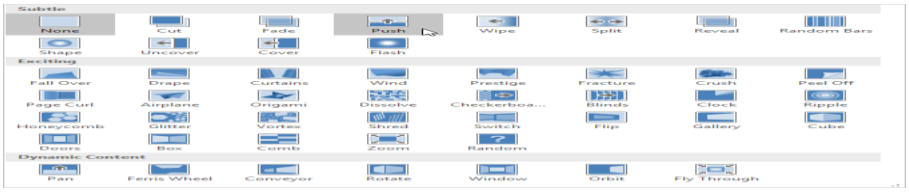
You can add special effect between each slide of your PowerPoint presentation, by using the feature slide transitions. A transition is a special visual effect that make the slide show attractive and eye-catching. By default, there is no transition effect on the slide. It can be added to the presentation in the following manner:

To apply a transition

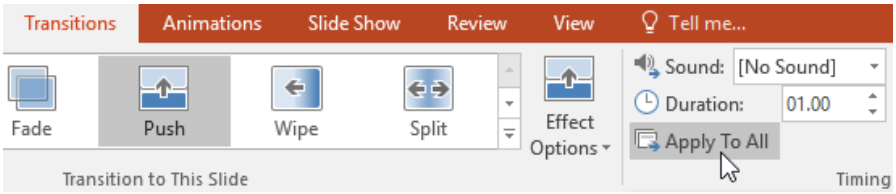
- Choose the particular slide from the Slide Navigation pane to apply transition. The slide shown below will appear after the transition.
- Click the Transitions tab from the menu tab of power point, then explore the transition effects from "Transition to This Slide group". By default, none is selected to each slide.
- All the transition effects can be explored by clicking the more drop-down arrow.



- Select the transition from the group to apply on the selected slide. It would be useful to automatically preview the transition.



- If you want to apply the transition effect to all the slides then go to ribbon in right corner, look into the Timing group click on the "Apply To All" option and then same transition effect will be applied to all slides of your presentation.

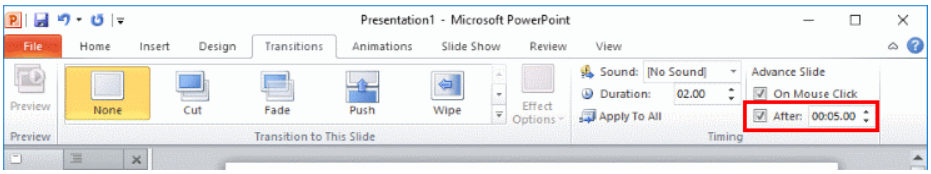


6.3.5 Automating the Slide Show

- Go to the Slide Show tab in the menu bar, then click on Set Up Slide Show button to automate the slide show.

- The Set Up Show dialog box will be display to you and you need to select the relevant options from Show type, Show options, Show slides, Advance slides and Multiple show monitors for the presentation and click Ok.

- Click on "Transitions" tab from the menu bar.
- Then go to "Advance Slide" area and select check box option "After", setup the elapse time for the presentation to advance to the next slide.



6.4 Summary

- The presentation can be created, saved and edited using various features.
- Power point has the ability to import data from other applications like Word, Excel and other applications.
- Other objects like Video clips, Pictures and Audio Clips can be added.
- Slide Show can be seen by pressing F5 or Slide Show option.
- Transition effects can be appeared during the slide show.
- Slide Transition Time is the Time during which the Slide will be active. It can be set by using Transitions Tab.

6.5 PRACTICAL EXERCISE

Q: 1 Make a presentation of 3 slides to describe yourself with different layouts (e.g., use title and content layout). Use automatic slide advancement effect by 5 seconds.

Q: 2 Create a presentation with 5 slides on the topic "Basics of Computer". Add picture, change background colour for each slide and change the design theme of your presentation.

Q: 3 Make slides with your introduction and academic qualification and insert your picture at right side. Insert the current date and time in the footer and slide number.

Q: 4 Prepare a presentation with animation and transition for any organization with minimum five slides using facet design theme.

Q: 5 Make table of 5 rows and 2 columns and insert the following data on the first slide and colour the table of your choice.

Items	Sales (Amount)
Apple	12000
Mango	10000
Grapes	20000
Orange	15000

Multiple Choice Questions:

Q 1: Which of the following tool enables you to add text to a slide without using the standard placeholders [19]?

A. Text tool
B. Line tool
C. Drawing tool
D. Auto shapes tool

Q 2: What happens if you edited an image inserted in PowerPoint?

A. The original file which was inserted is not modified
B. The original file that was inserted is changed
C. The original file is modified when you save presentation
D. None of above

Q3: What happens if you select first and second slide and then click on New Slide button on toolbar?

A. A new slide is being inserted as the first slide in presentation
B. A new slide is inserted as the second slide in presentation
C. A new slide is inserted as the third slide in presentation
D. None of above

Q4: In a presentation of PowerPoint, the special effects used to introduce slides are known as -

A. Custom Animation
B. Transitions
C. Annotations
D. None of the above

Q5: Which of the following shortcut key is used to start the slideshow?

A. Using F5 key
B. Using F3 key
C. Using F1 key
D. Using F6 key

Q6: Which of the following is the default standard layout in PowerPoint?

A. Blank slide
B. Title and content slide
C. Title slide
D. None of the above

Q7: Which of the following fill effects can be used to fill the background of the slide?

A. Picture
B. Gradient
C. Texture
D. All of the above

Q8: Which of the following option is correct to insert the chart as part of the PowerPoint presentation?

A. Insert -> Chart
B. Edit -> Chart
C. View -> Chart
D. All of the above

Q9: Which of the following are the uses of the PowerPoint presentation?

A. It can be used for project presentations
B. Communication of planning
C. Used to represent the data in an attractive way
D. All of the above

Q10: Is it possible to convert a PowerPoint presentation into a video?

A. Yes
B. No
C. May be
D. Can't say

B.Sc.(DATA SCIENCE)

SEMESTER-I

FUNDAMENTAL OF IT

UNIT VII: USING SPREADSHEET STATISTICAL FUNCTIONS

STRUCTURE

7.0 OBJECTIVES

- To know about the Mathematical Functions such as COUNT, SUM, AVERGAE, PRODUCT, POWER and SQRT functions
- To use the MAX and MIN functions to calculate the highest and lowest values from a set of cells.
- To learn about copy and paste formulas without formats applied to a cell location.
- To implement various Statistical Functions like MODE, MEDIAN, and MEAN etc.
- To design and implement Logical Functions like IF, COUNTIF, SUMIF etc.

7.1 INTRODUCTION

A spreadsheet is an electronic graph sheet that divided into rows and columns and can help arrange, calculate and sort data. The width of the rows and columns can be changed according to the user's choice. The rows are marked with positive integers like 1 and columns are marked with Alphabets like A. The rectangular boxes formed by the intersection of rows and columns is known as cell. MS-EXCEL has 256 columns and 65536 rows in one workbook. There are three worksheets per one workbook by default. Data can be represented in numeric values, text, functions, formulas and references [15].

To analyze the data in MS-EXCEL, statistical functions can be used. This chapter will help you to understand the meaning of the basic statistical functions.

7.2 STATISTICAL FUNCTIONS

There are many statistical functions like sum, count, median, mode, standard deviation, etc., are present in the MS Excel. It can also implement the logical operations like if, average if, sumif, etc.

The following points needs to be take care while writing the format of a user defined function

- Each Function must start with 'equal to' (=) sign
- Round braces are used to indicate the opening and closing of the function.
- Arguments are written within the parenthesis
- Commas can be used to separate the different arguments.

Example: The basic syntax of the function is shown below:

Syntax=Function Name (Argument)
=SUM ((B3:H3)), 50, 90)

This function will sum the values from cells B3 to H3 along with the constants 50 and 90.

Statistical functions are used to analyze the statistical data. The results can also be represented into Graphical or Pictorial form. To implement all these functions, this module has used MS Excel 2016. Some of the statistical functions are shown below using Employee Database as shown in Figure 7.1.

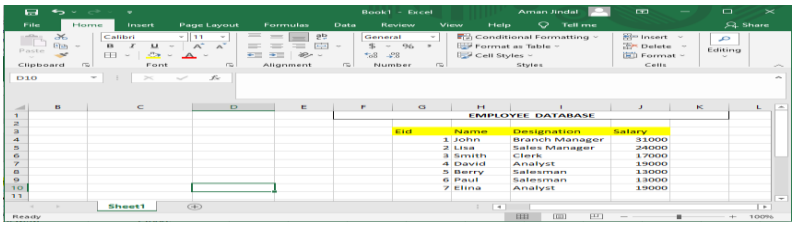


Figure 4.1: Employee Database

7.2.1 SUM()

Sum is a predefined function in the MS Excel. This function calculates the sum of the numerical values present in the range of cells mentioned in the argument. The formula for sum is **=SUM (number1, [number2],..).** For a range of cell, argument can be given as shown in Figure 7.2.

Example- Find the sum of salary of all employee.
Result- =SUM (J4:J10)
= 136000/-

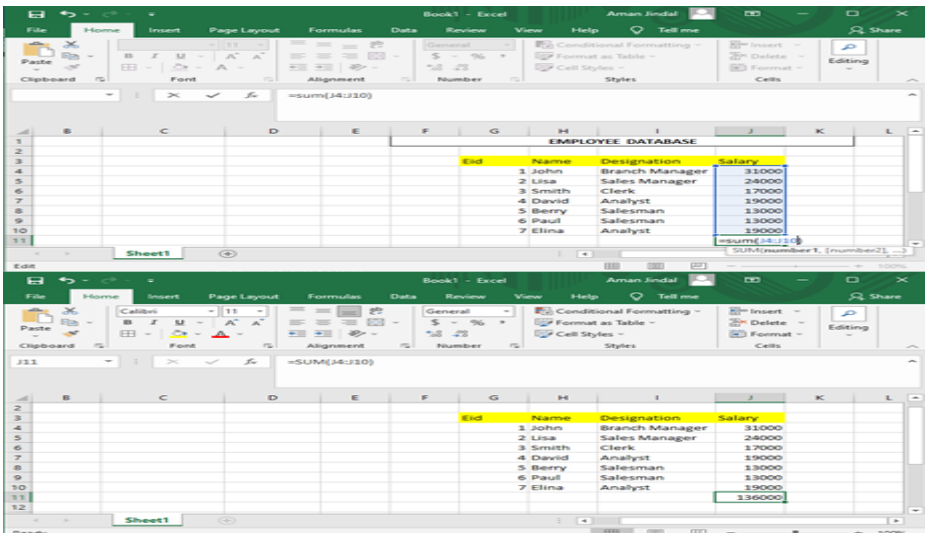


Figure 7.2: Sum of Salary

7.2.2 COUNT()

During entries in the worksheet, it becomes difficult to recall the number of entries which we have made in the worksheet. The Count function helps to count the number of cells within a range of cells. An implementation can be seen in Figure 7.3, for Count the number of employee in the company. The Syntax and example is written below:

Syntax: **COUNT (cell_1:cell_n).**
=COUNT (J4:J10)
Answer = 7

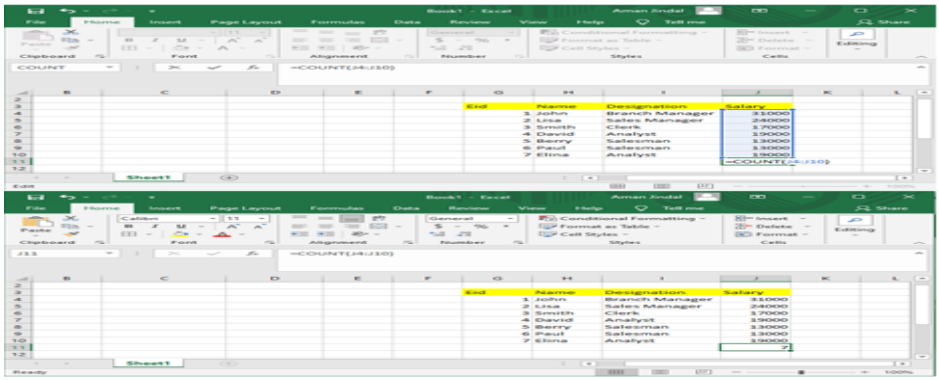


Figure 7.3: Counting the number of employees

If the user wants to count the number of entries without using cell references, then it could be written as:

=COUNT (23, 24, 67, 78, 89, 90) The result will be 6 =COUNT (“23”, 24, 67,78) The result will be 4, as the text value is converted into numeric by default

7.2.3 AVERAGE()

This function calculates the average of the numbers specified in the argument. The formula to calculate average is **=AVERAGE (number1, [number2],)**.

Example: Find the average salary among all the employees of the company.
The result can be seen in Figure 7.4.

Syntax=AVERAGE (number 1, number2----) Suppose you need to find out the average from J4 to J10 cell range, then it will be calculated =AVERAGE (J4:J10) =19428.57/-

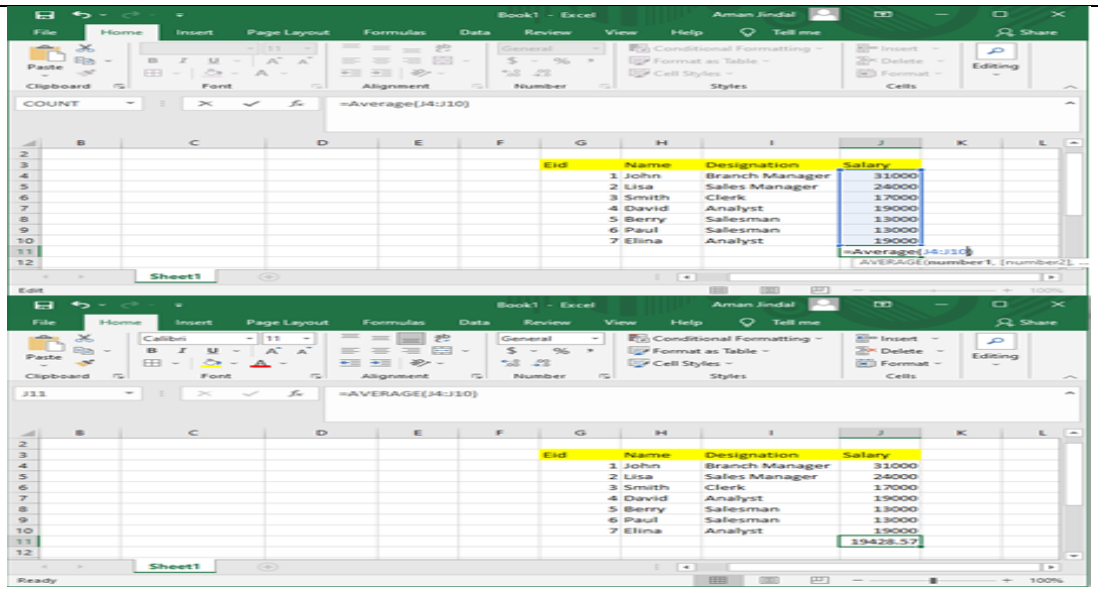


Figure 7.4: Average Function

7.2.4 PRODUCT()

The PRODUCT function is inbuilt function multiplies the numbers and returns the product as the output.

Syntax: PRODUCT(3,4,10), it returns the 120
Where Argument type : Number and return type is number
=PRODUCT("4",5, 3), the answer will be 60, it will take by default as number.

7.2.5 POWER()

The Power function will take two values of the specified cells for numeric constants, in the syntax first value defines the number and second value as a power. It returns the results as the power of a number.

Syntax=POWER(number, power)
Example1=POWER(2,3), Example2=POWER("2",3)
Answer will be 8 for both the examples, it will also take as numeric as a default argument.

7.2.6 SQRT()

This function will displays the square root of the positive number and returns the positive number. The square root of negative number cannot be evaluated.

Syntax=SQRT(number)
Example1=SQRT(64), result will be 8
Example2=SQRT("100"), results will be 10

7.2.7 MAXIMUM and MINIMUM

To find maximum and minimum value from a given set of values, MAX and MIN function can be used respectively. See Figure 7.5 for its implementation.

Syntax: Max(Number1,Number2,-------), Syntax of Min=Min(Number1, Number2----)	
Here return type: Number, Argument Type=Number	
Example: Find the maximum and minimum salary given to the employee.	
Result: =MAX(E4:E10)	=MIN(E4:E10)
=31000/-	=13000/-

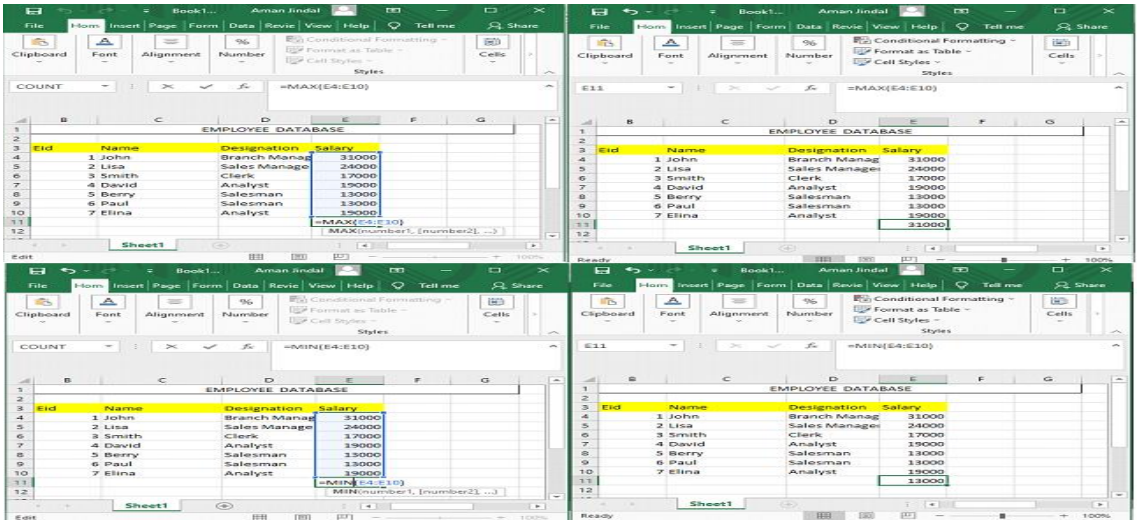


Figure 7.5: Maximum and Minimum Function

If the user write it in the constant form like =MAX("78",34,37,29), It will returns 78 because it will consider every argument as the constant term. MIN("78",34,37,29), it will return 29 similarly it will also take every argument as the constant.

7.2.8 MEDIAN()

The median is the central score for a set of data that has been arranged in order of magnitude. which is less affected by outliers and skewed data. In order to calculate the median, suppose we have the data below: The median function finds the median of the numbers passed as an argument.

The syntax for median function is =**MEDIAN(number1, [number2],…..)**
Example: Find the median salary from the employee data.(see Figure 7.6)
Result: =MEDIAN(E4:E10)
=19000/-

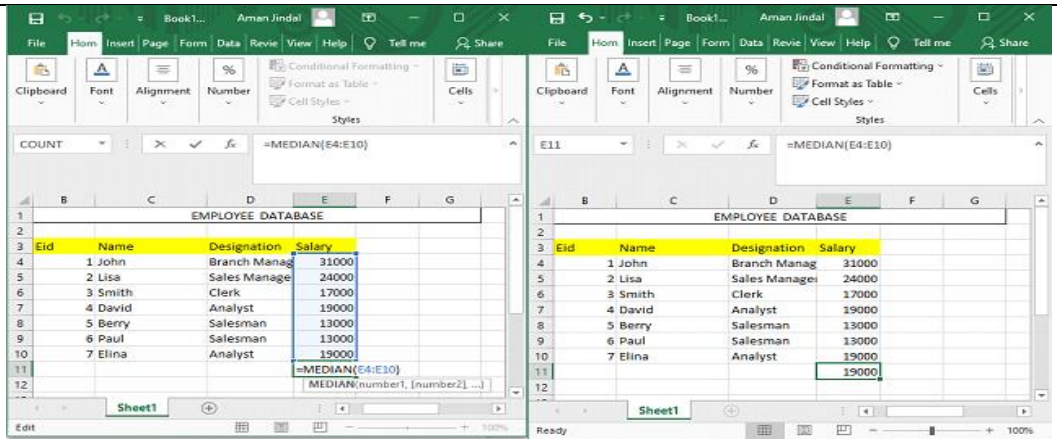


Figure 7.6: Median Function

7.2.9 MODE()

Mode function calculates the most frequently occurring value from the given set of arguments.

The syntax for mode function is =**MODE(number1, [number2],…..)**.
Example:- Find the mode value of the salary column of the employee data.
Result: =MODE(E4:E10)
=19000/-

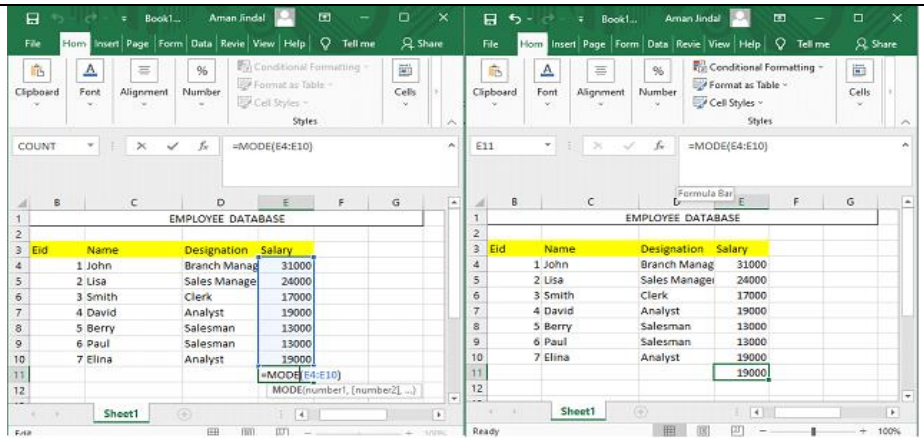


Figure 7.7: Mode Function

7.2.10 STDEV.S ()

It estimates the standard deviation of the numbers give as an argument. If the arguments consist logical values or text then STDEV.S ignores them. Its implementation can be seen in Figure 7.8.

The syntan of this function is =**STDEV.S(number1, [number2],…..)**.
Example:- Find the standard deviation of the employee data
Result: =STDEV(E4:E10)
=6373.307

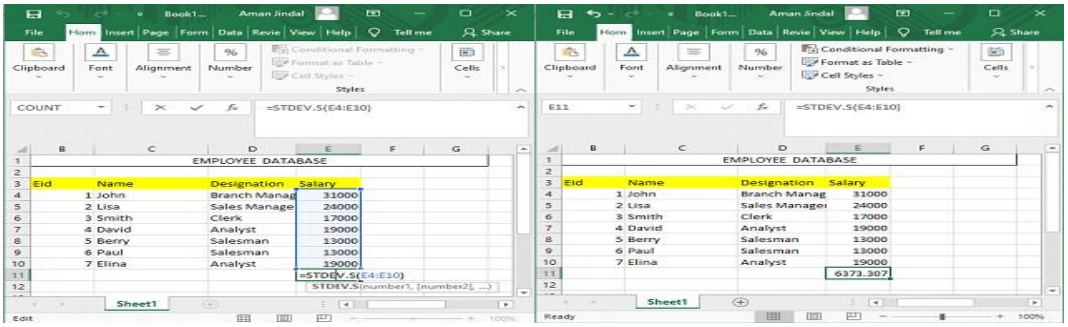


Figure 7.8: Standard Deviation Function

7.2.11 ABS()

ABS function finds the absolute value of a number. It returns a positive number if any number is passed as an argument. Its syntax is =**ABS(number).**

Example: Find the product, square root and absolute value of the data given in Figure 7.9

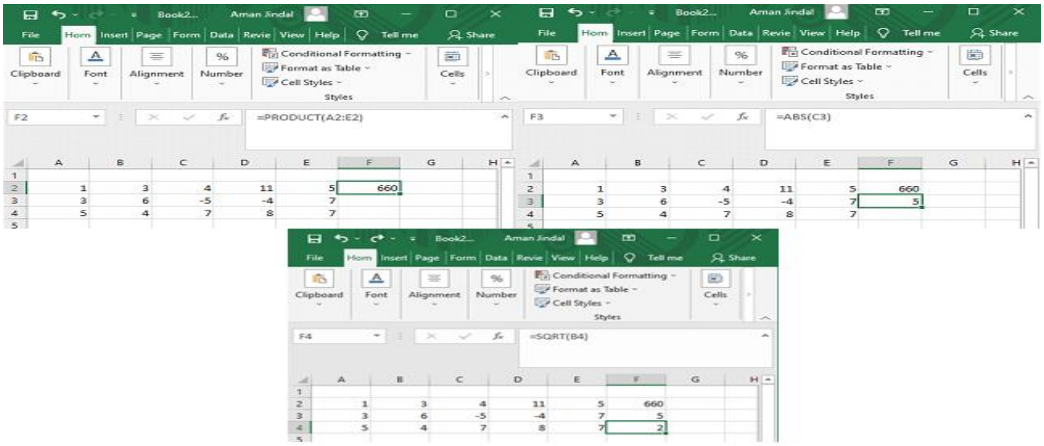


Figure 7.9: Product, Square root and Absolute function

7.2.12 QUARTILE()

Quartile function returns the quartile of a given set of values. It can return first quartile, second quartile, third quartile, maximum value and minimum value.

Syntax: =QUARTILE.INC(array, quart) If quart=0, then returns minimum value If quart=1, then returns 1st Quartile If quart=2, then returns 2nd Quartile If quart=3, then returns 3rd Quartile If quart=4, then returns maximum value

Example: Find the first, second, third quartile of student marks as shown in Figure 7.10

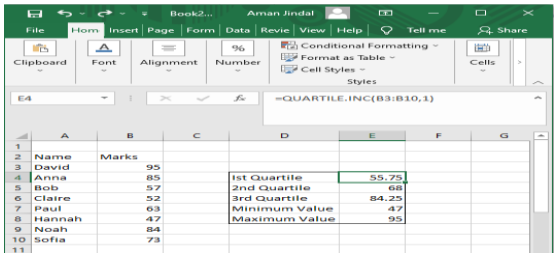


Figure 7.10: Quartile function

7.2.13 PERCENTILE()

This function calculates the kth percentile for the given set of arguments. The syntax for this function is =PERCENTILE.INC(array,k_value)

Example: Find the 90th percentile, 80th percentile, 70th percentile, 60th percentile, 50th percentile of the student marks shown in Figure 7.11.

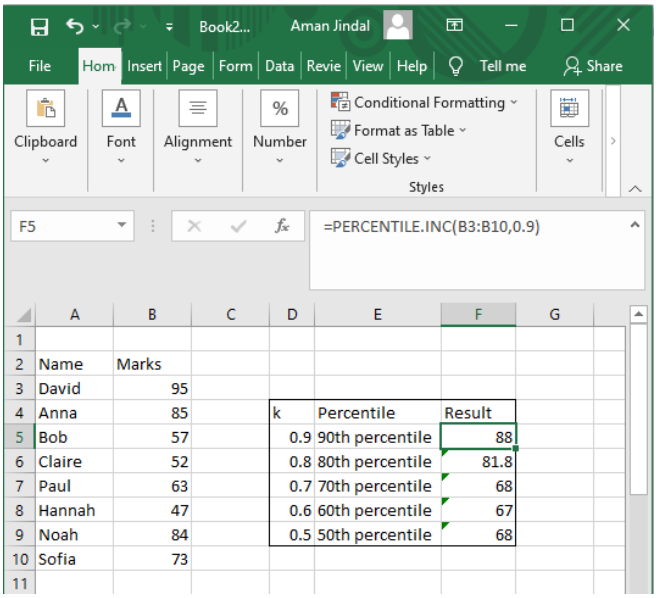


Figure 7.11: Percentile function

7.2.14 COUNTA and COUNTBLANK()

COUNTA function counts the non-empty cells from the given set of arguments. It means it will count numbers, text, logical values, etc. Whereas COUNTBLAK counts the empty or blank cell from the provided cell range in the argument.

Syntax: =COUNTA(value1,[value2],…)

=COUNTBLANK(value1,[value2],…)

Example: Count the non-empty cell from the data given in figure 7.1. Also count the empty cell from the given data

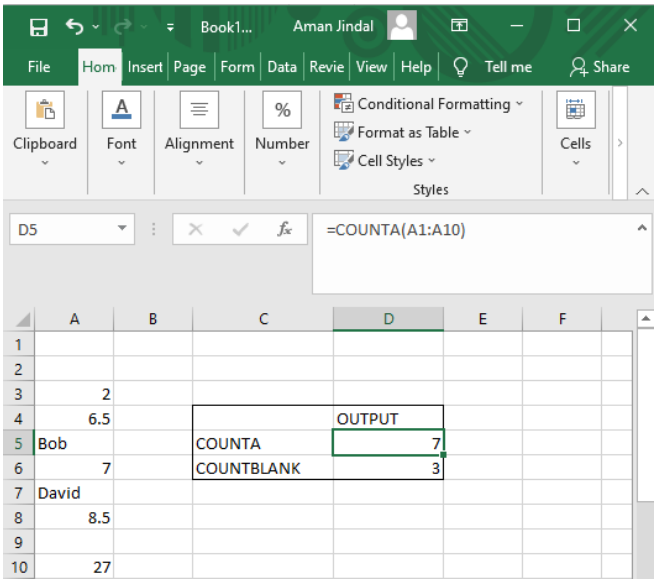


Figure 7.12: COUNTA and COUNTBLANK function

7.2.15 CORREL()

It calculates the correlation coefficient of the two given dataset or array.

Syntax: =CORREL(array1, array2)

Example: Find the correlation coefficient of the dataset given in figure 7.13

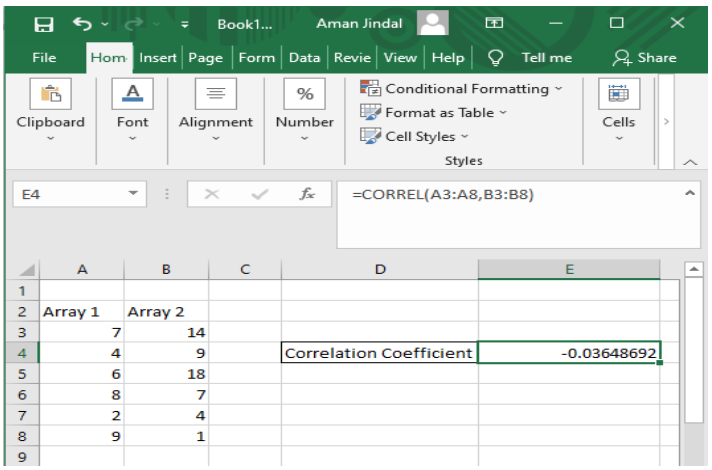


Figure 7.13: CORREL Function

7.2.16 LOGICAL OPERATIONS(IF, AVERAGEIF, SUMIF, COUNTIF)

IF is a logical operation that returns a value depending on the TRUE or FALSE result. The syntax of this operation is =IF (logical_test, [value_if_true], [value_if_false]).

Example: Calculate the result as PASS or FAIL of student data as shown in figure 7.14. Student will pass the examination only if marks is greater than 60.

Result:

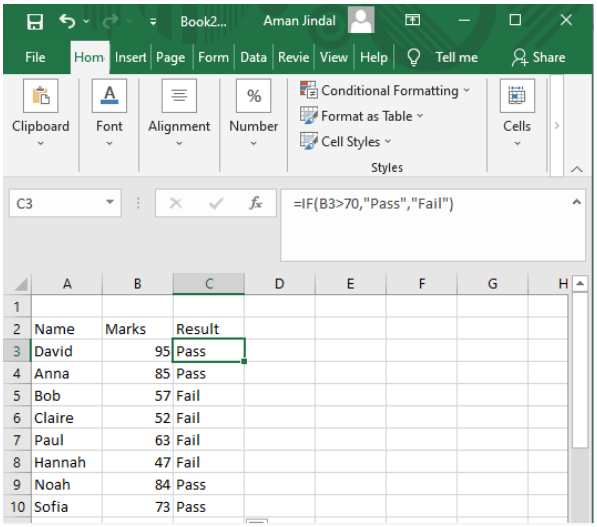


Figure 7.14: IF Function

AverageIf operation calculates the average of the given set of values depending on the given condition or criteria.

Syntax: =AVERAGEIF(array1,criteria,[array2],…)

Example: Find the average marks among students whose marks are greater than 50. See figure 7.15

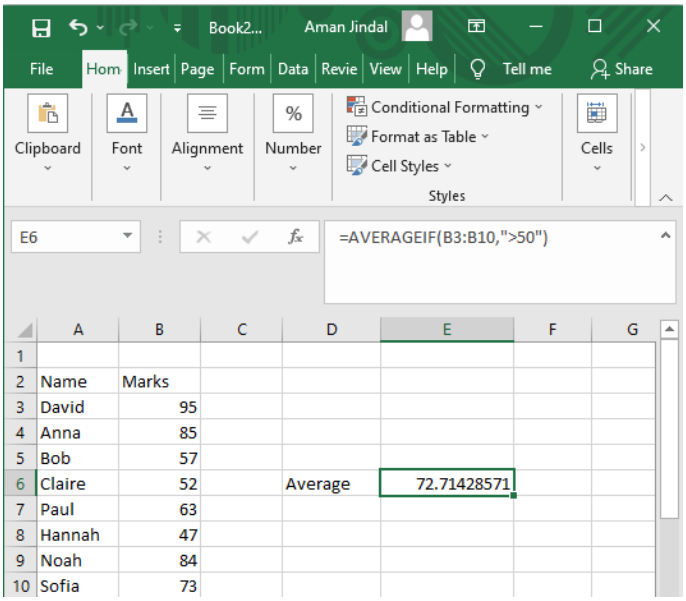


Figure 7.15: AVERAGEIF Function

7.2.17 SUMIF()

It is also a conditional function SumIf function is used to add up the range of cells satisfying the conditions given by the user, condition is to be represented in double quotes. The syntax below shows to calculate the sum of the given set of values depending on the given criteria.

Syntax: =SUMIF(array1,criteria,[array2],…) Example1=SUMIF(A3:A10,"=Bob",B2:B10),

It will answer the sum of marks whose name is Bob, answer for this example will be 57 only, because only one value is there for only Bob.

Example2: Find the sum of the marks among students whose marks are greater than 65. See Figure 7.16

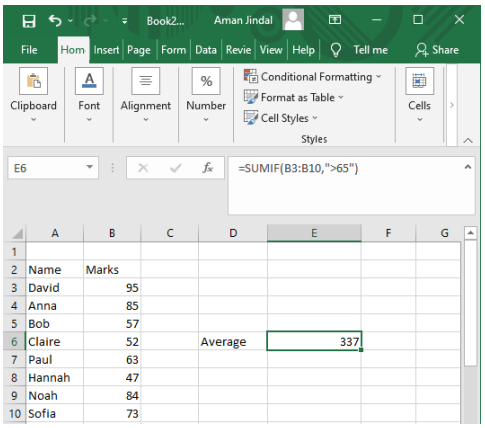


Figure 7.16: SUMIF Function

7.2.18 COUNTIF()

CountIf() operation counts the cell from the given set of values depending on the given condition.

Syntax: =COUNTIF(range of the values, “condition”)
Example=COUNTIF(B3:B10,”>75”)
Return Type: Number

Example: Find the number of students having marks greater than 75. See Figure 7.17

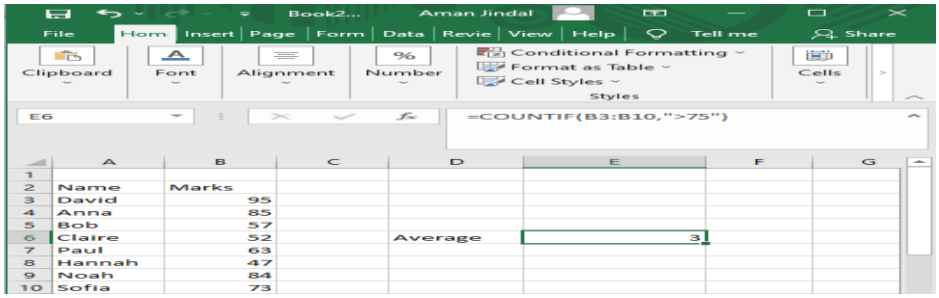


Figure 7.17: COUNTIF Function

7.3 SUMMARY

- Built in Functions are Pre-designed formulas in Excel to perform both simple and complex functions. Built in Functions include Mathematical/Statistical and Logical Functions etc.
- Statistical Functions SUM() is used to find the total of all the values
- AVERAGE(): To find the arithmetic mean of group of numbers
- PRODUCT(): To multiply given set of cell locations.
- POWER(): To calculate the raise to the power of any number
- SQRT(): The Positive Square root of a number
- MAX(), MIN(): To find the highest and minimum value from a set of cells
- MEDIAN(): To find the central number from a group of numbers
- MODE():To find the number which is frequently occurs from a set of numbers
- STDEV.S(): This function is for a set of numbers based on a sample.
- ABS(): Absolute value of any number.
- QUARTILE(),PERCENTILE()
- COUNTA(), COUNTBLANK(), COUNT() are used to quickly count the number of items in a list.
- LOGICAL FUNCTIONS:SUMIF(), AVERAGEIF(), COUNTIF() are used to apply the functions based on the condition.

7.4 PRACTICE EXERCISE

Q1. Differentiate between the following:

- Max() and Min() Functions
- SUMIF() and COUNTIF()

- MEDIAN and MODE
- SUM and COUNT
- COUNTA and COUNTBLANK

Q2. Calculate the average, mode, standard deviation of the data given below:

	Quarter 1	**Quarter 2**	**Quarter 3**
January	$400	$200	$350
April	$340	$140	$405
June	$107	$98	$55

Q3. The following data shows the inventory figures for 100-gallon tanks at something's Fishy

	Something's Fishy					
	100-Gallon Fish Tanks Inventory					
						Amount
2-Jan	Beginning Inventory	24	Units	@	$30.00	?
14-May	Purchase	20	Units	@	$34.50	?
10-Jul	Purchase	33	Units	@	$36.70	?
2-Aug	Purchase	33	Units	@	$49.75	?
Fish tanks available for sale	?	Cost of tanks available for sale	?			

- Enter the data into Excel in the same format and Find the amount for each date
- Calculate the cost of tanks available for each date
- How many total tanks are available for the sale?

Q4. Enter the data of 20 students for five subjects like Maths, Chemistry, Biology, English and Hindi. Enetr the marks of each student out of 100. Find the student who scores first division using logical functions by using conditions If total marks>=60, print a message "First Division" Else IF Marks>50 Print:" Second Divison, Else IF marks>40, PRINT: Third Divison, Else PRINT "FAIL"

Q5. Use Logical operations to calculate the following

	Sales ($ millions)
Quart. 1	500
Quart. 2	350
Quart. 3	495
Quart. 4	620

Which quarter is the better?	
Find the maximum and minimum sales	
Compare through logical if	

MCQ Based Questions

1. _____ Function in Excel tells how many numeric entries are there.

a) COUNT
b) SUM
c) NUM
d) CHKNUM

2. Which is not a Function in MS Excel?

a) SUM
b) AVG
c) MAX
d) MIN

3. Functions in MS Excel must begin with ___

a) An () sign
b) An Equal Sign
c) A Plus Sign
d) A > Sign

4. Which function in Excel checks whether a condition is true or not ?

a) SUM
b) AVERAGE
c) COUNT
d) IF

5. Which of the following formulas is not entered correctly?

a) =10+50
b) =B7*B1
c) =B7+14
d) 10+50

6. Which of the following formulas will Excel Not be able to calculate?

a) SUM(Sales)-A3
b) SUM(A1:A5)*.5
c) SUM(A1:A5)/(10-10)
d) SUM(A1:A5)-10

7. Which function will be performed first in this formula?
=IF(SUM(C2:C4)>500,"Yes","No")

a) IF
b) SUM

Q8.Statistical calculations and preparation of tables and graphs can be done using

a) Adobe Photoshop b) Excel c) Notepad d) PowerPoint

Q9.Which function is used to count the cells using condition

a) SUMIF()

b) COUNT ()

c) COUNTIF ()

d. None of these

Q10.____Function is used to find the frequently used data

a) Mode ()

b)Median()

c) STDEV.S()

d) MEAN()

B.Sc.(DATA SCIENCE)

SEMESTER-I

FUNDAMENTAL OF IT

UNIT VIII: FORMAT TEXT BY USING FUNCTIONS

STRUCTURE

8.0 OBJECTIVES

- To know about various functions like Right, Left and Mid etc.
- To implement the text formatting using various functions such as Upper, Lower Proper, and Concatenate etc.
- To create various charts like Pie chart, Area Chart, Bar Chart, Line chart etc.
- To generate the pivot tables
- To remove the duplicate values from the file

8.1 FORMATTING TEXT

To create a proper spreadsheet in Excel, there is a need to do formatting of the text of the cells. There are some functions which can be used to format the text of the cell. These functions have been implemented using MS EXCEL 2016. UPPER, LOWER, PROPER, LEFT, RIGHT, MIDDLE, CONCATENATE, etc. are such functions to format the text.

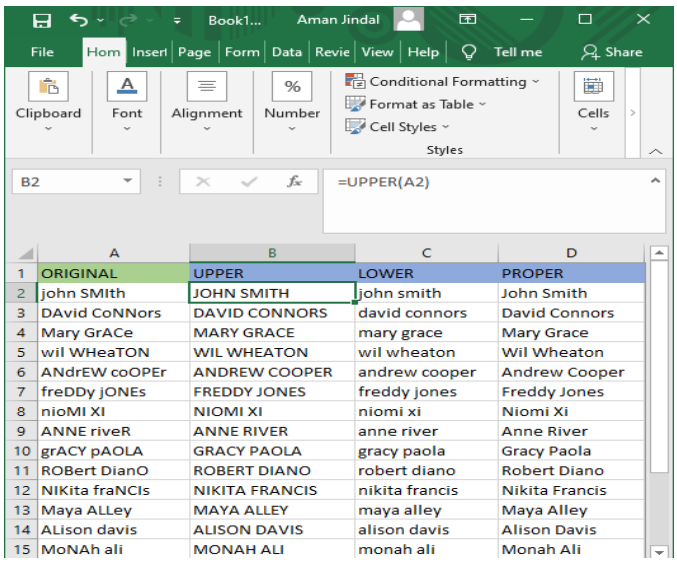
8.1.1 Using UPPER, LOWER and PROPER

UPPER function is used to change the characters of text to capitals. LOWER function will change the text to lower case. PROPER function changes only the first character of each word to capital and rest of the characters to lower case. If a text contains numbers, punctuation or special characters then these are not affected by UPPER, LOWER or PROPER function[18].

Syntax of each function is given below: -

- =UPPER(text)
- =LOWER(text)
- =PROPER(text)

Here, argument text will be a cell number that contains a text. See the example shown in figure 8.1



	A	B	C	D
1	ORIGINAL	UPPER	LOWER	PROPER
2	john SMIth	JOHN SMITH	john smith	John Smith
3	DAvid CoNNors	DAVID CONNORS	david connors	David Connors
4	Mary GrACe	MARY GRACE	mary grace	Mary Grace
5	wil WHeaTON	WIL WHEATON	wil wheaton	Wil Wheaton
6	ANdrEW coOPEr	ANDREW COOPER	andrew cooper	Andrew Cooper
7	freDDy jONEs	FREDDY JONES	freddy jones	Freddy Jones
8	nioMI XI	NIOMI XI	niomi xi	Niomi Xi
9	ANNE riveR	ANNE RIVER	anne river	Anne River
10	grACY pAOLA	GRACY PAOLA	gracy paola	Gracy Paola
11	ROBert DianO	ROBERT DIANO	robert diano	Robert Diano
12	NIKita fraNCIs	NIKITA FRANCIS	nikita francis	Nikita Francis
13	Maya ALLey	MAYA ALLEY	maya alley	Maya Alley
14	ALison davis	ALISON DAVIS	alison davis	Alison Davis
15	MoNAh ali	MONAH ALI	monah ali	Monah Ali

Figure 8.1: UPPER, LOWER and PROPER Function

Functions applied on row 2 are as follows:

- Cell B2: =UPPER(A2)
- Cell C2: =LOWER(A2)
- Cell D2: =PROPER(A2)

8.1.2 Using LEFT, RIGHT and MID

Some functions are used to retrieve the characters or substring from a given text. A text can contain characters, numbers, special characters, punctuations and spaces. Such functions are given below:

- LEFT function retrieves a specific number of characters from left side of a given text. Syntax: =LEFT(text, [num_chars])
- RIGHT function retrieves a specific number of characters from right side of a given text. Syntax: =RIGHT(text, [num_chars])
- MID function retrieves a specific number of characters from the middle of a given text. Syntax: =MID(text, start_num, [num_chars])

Here,

- text will be a cell number.
- num_chars is the number of characters to be retrieved.
- start_num is the starting position of the characters to be retrieved.

An Example is given in the figure 8.2. In this example text is given in cell A1. Value of num_chars is 8 and start char is 3.

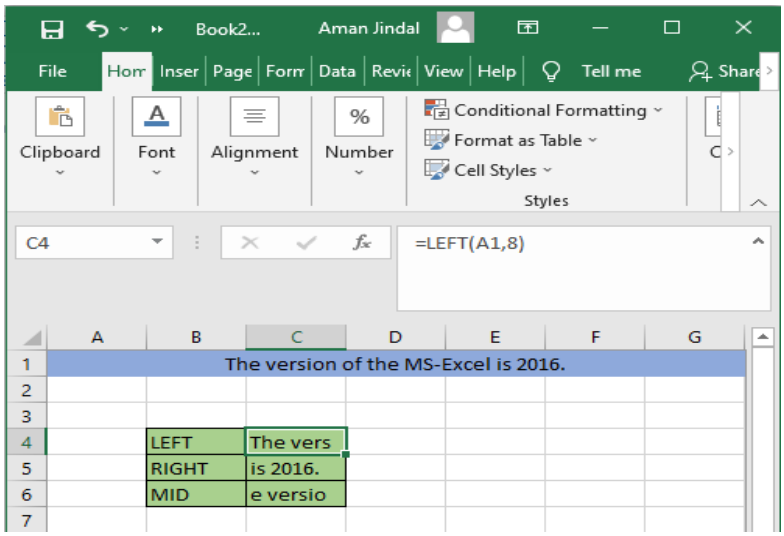


Figure 8.2: LEFT, RIGHT, MID Function

8.1.3 CONCATENATE

CONCATENATE function joins the several text strings into one text string. The syntax of the function is =CONCATENATE (text1, [text2], [text3],.).

Here, text can be a string or a cell number. A string will be represented within double quotes (“”) as shown in the example given in the figure 8.3.

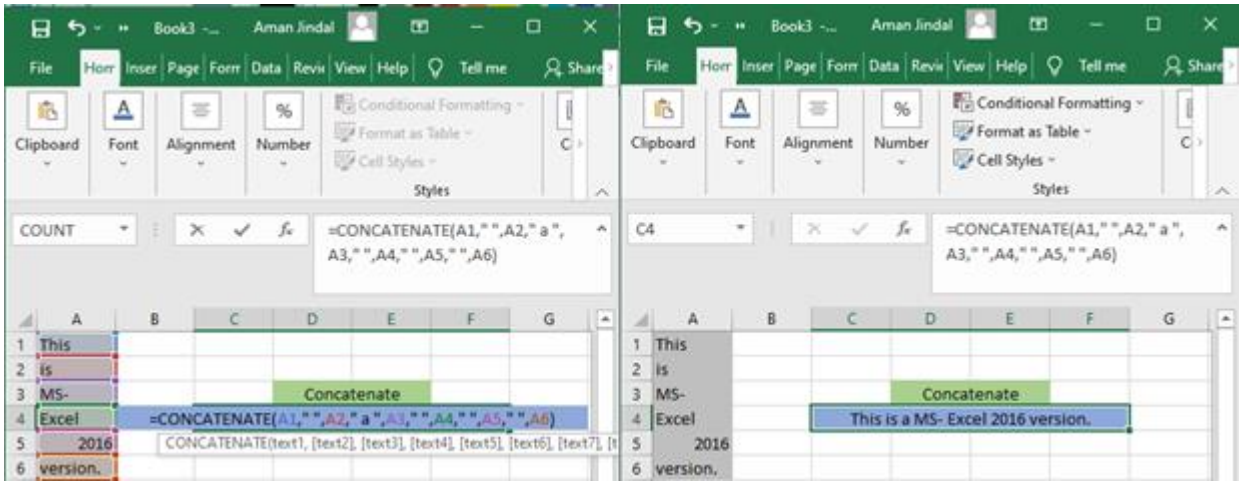
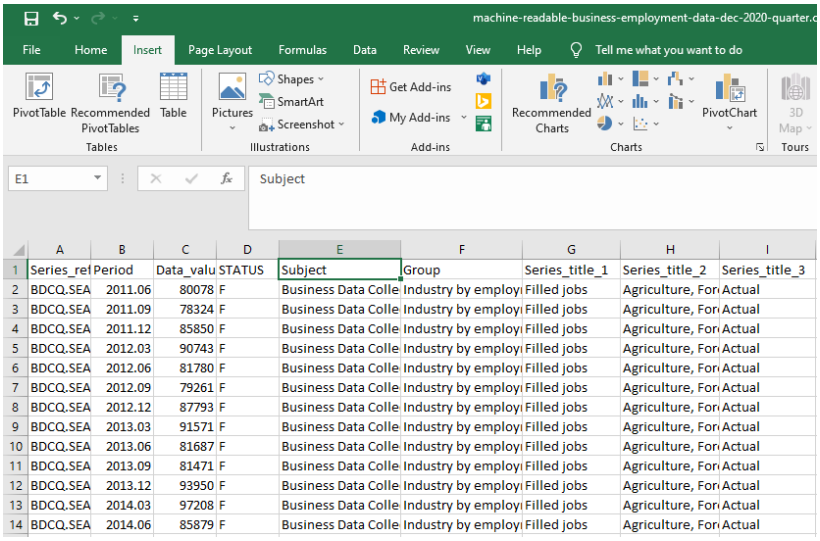


Figure 8.3: CONCATENATE Function

8.1.4 Pivot Table

Pivot table is the most useful tool of the MS-Excel. It allows us to extract the information from a large, complex and detailed dataset. It arranges and summarize the complex dataset. To implement this, an employee dataset is used as shown in figure 8.4 having 12317 rows.



	A	B	C	D	E	F	G	H	I
1	Series_ref	Period	Data_valu	STATUS	Subject	Group	Series_title_1	Series_title_2	Series_title_3
2	BDCQ.SEA	2011.06	80078	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
3	BDCQ.SEA	2011.09	78324	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
4	BDCQ.SEA	2011.12	85850	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
5	BDCQ.SEA	2012.03	90743	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
6	BDCQ.SEA	2012.06	81780	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
7	BDCQ.SEA	2012.09	79261	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
8	BDCQ.SEA	2012.12	87793	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
9	BDCQ.SEA	2013.03	91571	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
10	BDCQ.SEA	2013.06	81687	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
11	BDCQ.SEA	2013.09	81471	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
12	BDCQ.SEA	2013.12	93950	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
13	BDCQ.SEA	2014.03	97208	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual
14	BDCQ.SEA	2014.06	85879	F	Business Data Colle	Industry by employ	Filled jobs	Agriculture, For	Actual

Figure 8.4: Employee Dataset

Steps to create a pivot table is shown below: -

1. In Insert tab, click on the Pivot Table option as shown in Figure 8.5. A new window will appear. In this, first option is to select the table or range. Second option is to create pivot table on new worksheet or existing one. Select the appropriate option and click on OK.

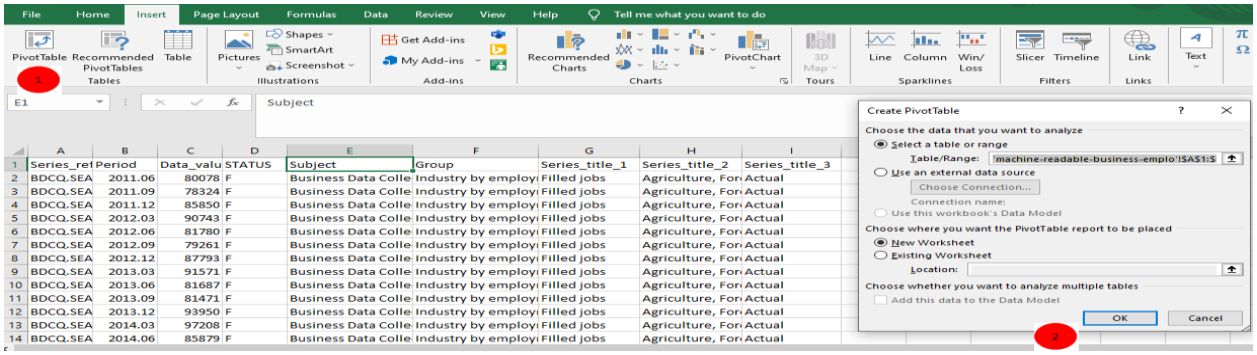


Figure 8.5: Step-1 to Create Pivot Table

2. After this, new page will appear as shown in figure 8.6. Here, firstly select the fields to include in the pivot table, then apply filters on it. According to the fields selected and filters applied pivot table will be created.

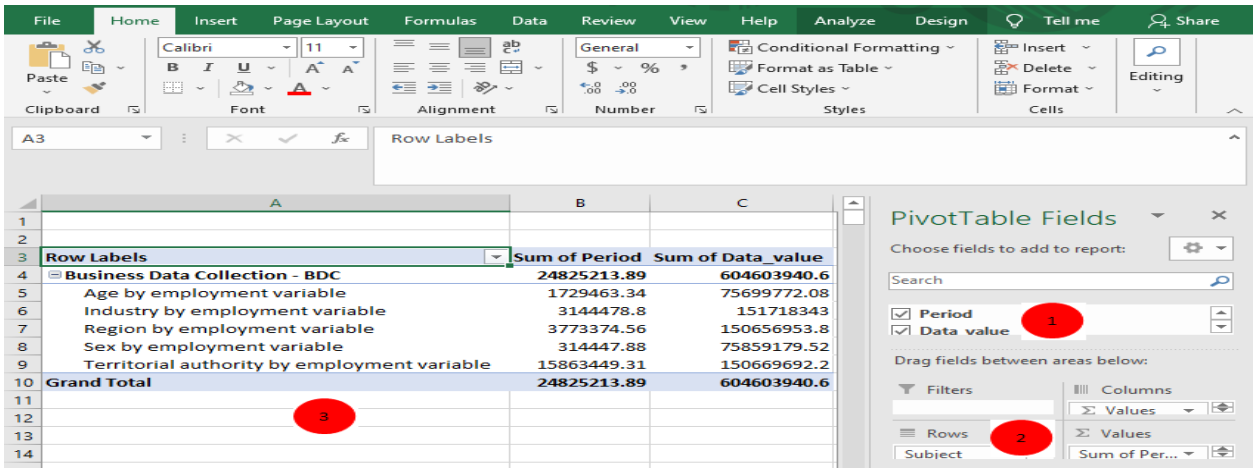


Figure 8.5: Step-2 to Create Pivot Table

8.1.5 Charts

A chart is a visual representation of data present in both rows and columns. It analyze the pattern and trends in the data sets. The dataset shown in figure 8.6 is used to prepare the chart [9].

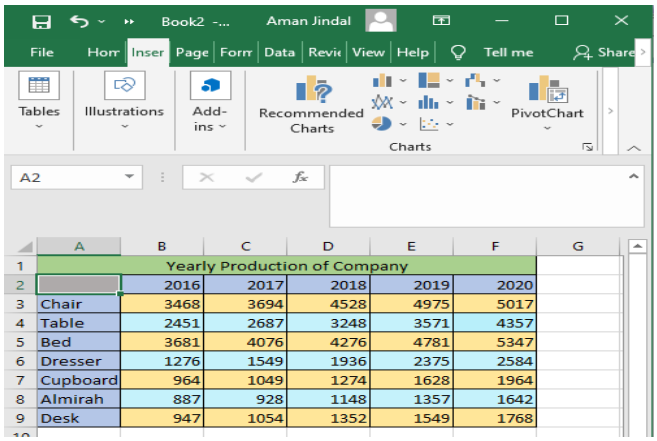


Figure 8.6: Dataset to create the charts

In Figure 8.7, some steps are mentioned that must be followed to create a chart:

1. Select the data to represent in the graph.
2. Click on the insert tab.
3. Select the appropriate chart type.

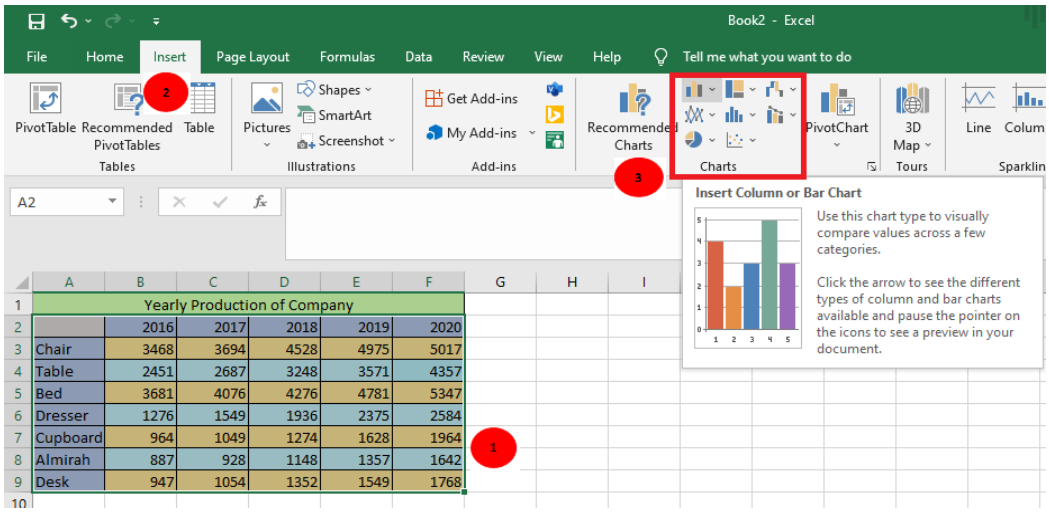


Figure 8.7: Creating a chart

There are various types chart available in MS-Excel 2016 version and that are:

- Bar or Column Chart
- Line Chart
- Area Chart
- Hierarchy Chart
- Pie Chart
- Doughnut Chart
- Statistical Chart
- Scatter or Bubble Chart
- Combo Chart

8.1.5.1 Bar or Column Chart

Bar chart is used to compare the values according to the categories. It is used when the order of the categories doesn't matter. It is also known as column chart. An Example of Bar Chart is shown in figure 8.8.

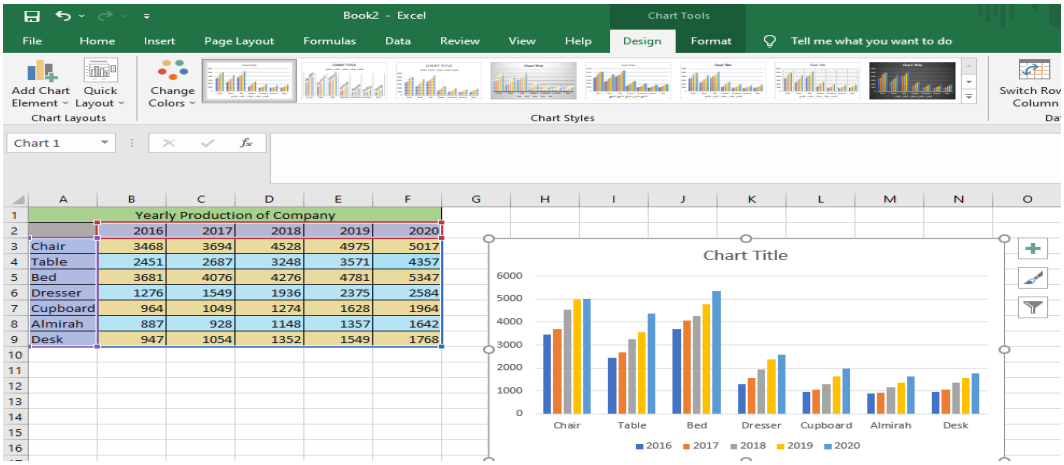


Figure 8.8: Bar or Column Chart

8.1.5.2 Line Chart

It shows the trends over categories or time (years, months, days). It represents the chart in the form of line. Each line has many data points as shown in Figure 8.9.

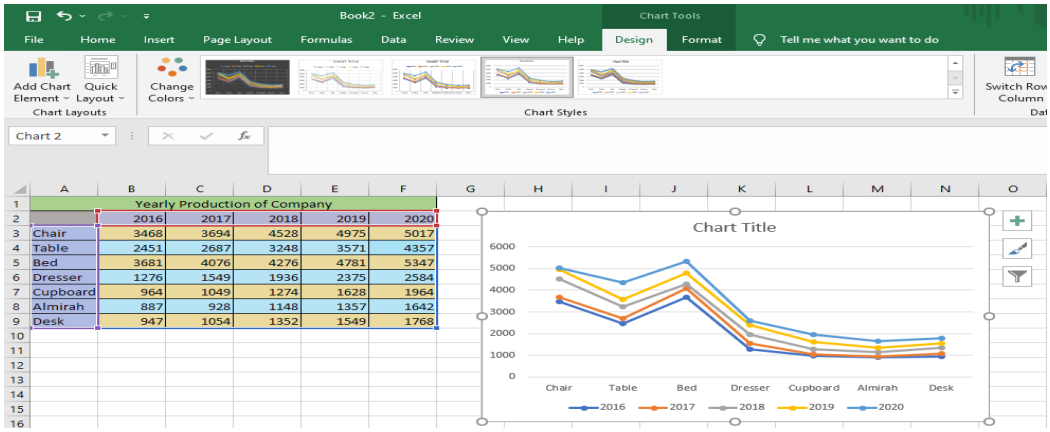


Figure 8.9: Line Chart

8.1.5.3 Area Chart

It represents the chart in the form of area in 2-D or 3-D. It shows the trends over categories or time (years, months, days). An example of the Area chart is shown in Figure 8.10.

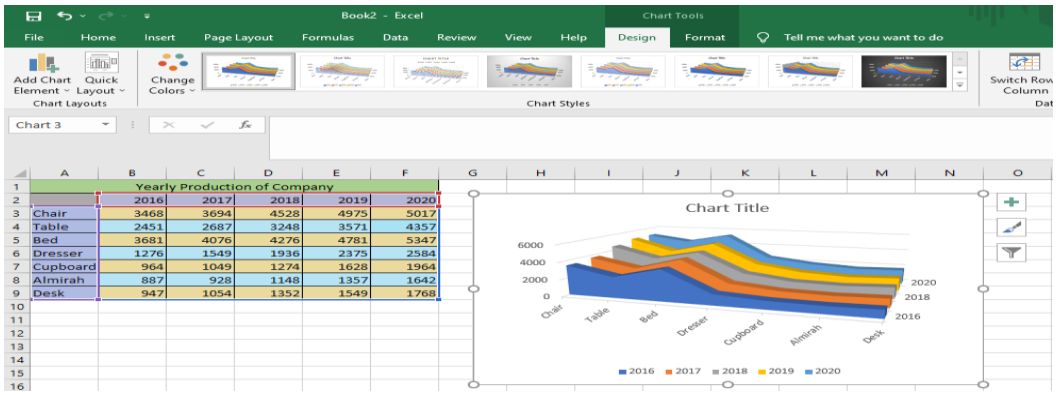


Figure 8.10: Area Chart

8.1.5.4 Hierarchy Chart

It represents the values of the dataset in hierarchical level. There are two ways to represent the chart in hierarchy, Treemap and Sunburst. Treemap will show the proportion within the hierarchical level as rectangles whereas Sunburst shows the proportion as rings. Its implementation can be seen in Figure 8.11

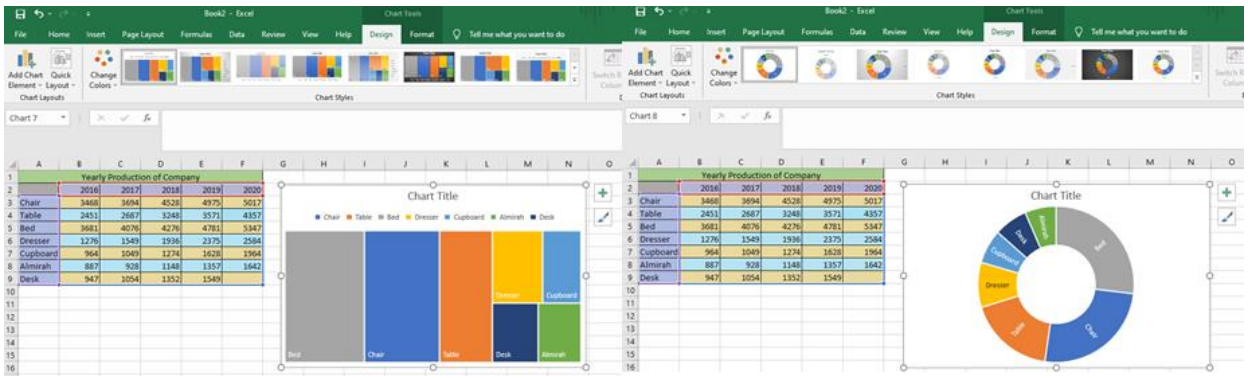


Figure 8.11: Hierarchy Chart (Treemap and Sunburst)

8.1.5.5 Pie Chart

This type of chart represents the whole dataset in the form of proportion. Each proportion will represent the category of the dataset. It can be represented in 2-D and 3-D. A 2-D representation of the pie chart is shown in Figure 8.12.

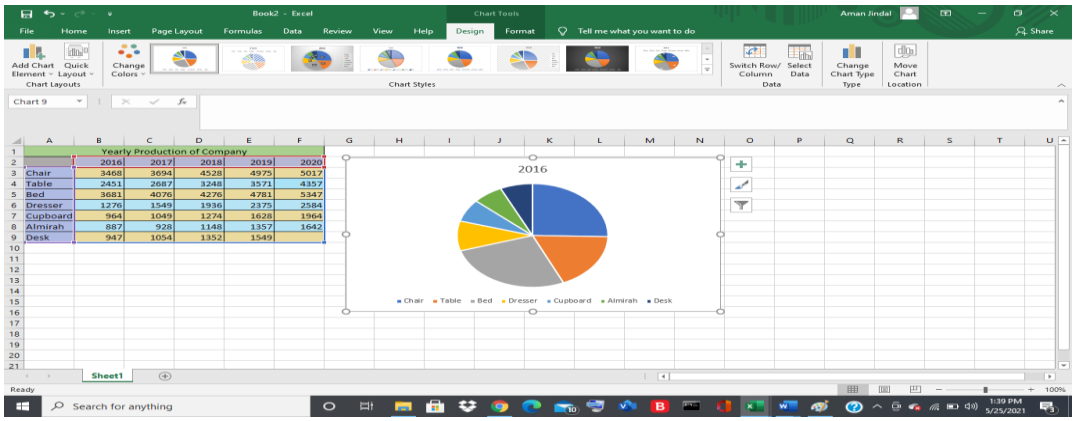


Figure 8.12: Pie Chart (2-D)

8.1.5.6 Doughnut Chart

This chart type is similar to Pie chart. But here, chart is represented in the form of doughnut. It is used when multiple series are present in the dataset. An example is shown in Figure 8.13.

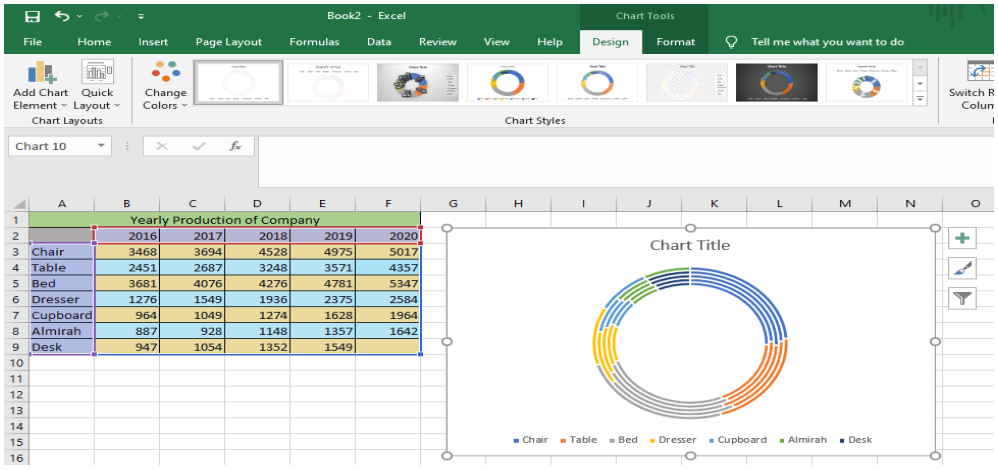


Figure 8.13: Doughnut Chart

8.1.5.7 Statistic Chart

This chart type shows the statistical analysis of the data values. An example of company production in the year 2016 is shown in Figure 8.14.

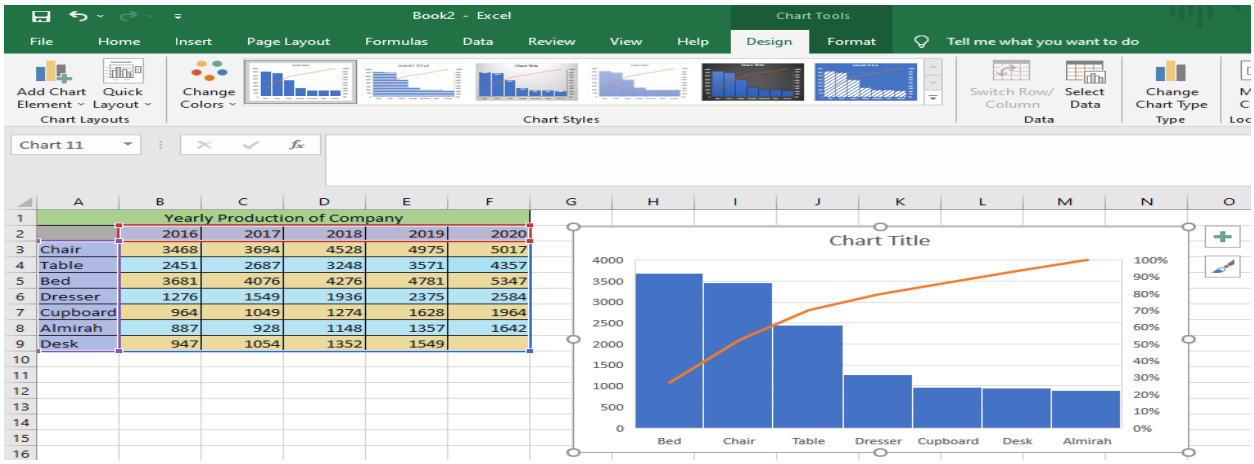


Figure 8.14: Statistic Chart

8.1.5.8 Scatter or Bubble Chart

Scatter chart compares the set of value and shows their relationship. It is also known as bubble chart. Year wise relationship between the values is shown in Figure 8.15.

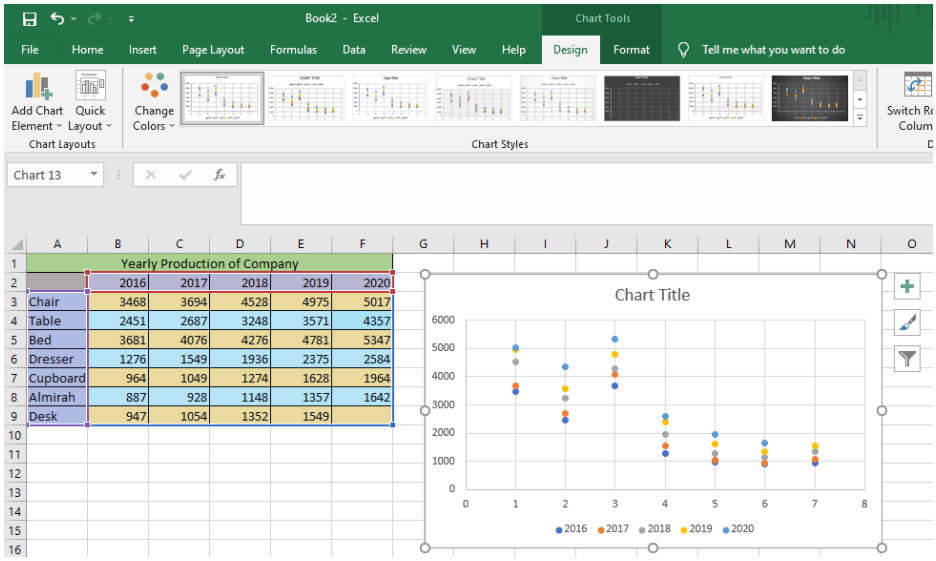


Figure 8.15: Scatter Chart

8.1.5.9 Combo Chart

Combo chart will combine various types if chart to highlight different information. It can be customized. Any type of chart can be chose to make a combo chart. It is used when the range of values varies widely or mixed type of data is present in the dataset. A combination of line and column chart is shown in the Figure 8.16.

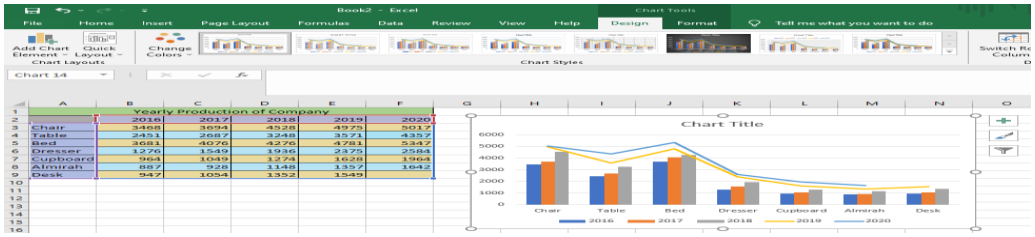


Figure 8.16: Combo Chart

8.1.6 Data Cleaning

In Excel analysis of various data values is performed. But there are many things that can go wrong while creating a spreadsheet like improper cases, misspelled words, duplicate data, unwanted spaces, etc. In this section, numerous ways to remove these errors will be discussed.

8.1.6.1 Removing Duplicate Values

This method is used to deal with the duplicate data present in the dataset. There are two actions, which can be taken on duplicate data: -

- Highlight the duplicate data
- Remove the duplicate data

The steps that must be followed to highlight the duplicate data is shown in Figure 8.17.

1. Select the data to check for duplicity.
2. In Home tab, click on the Conditional Formatting.

3. In Conditional formatting, click on the Highlight Cells Rules.
4. After this, click on the Duplicate Values.

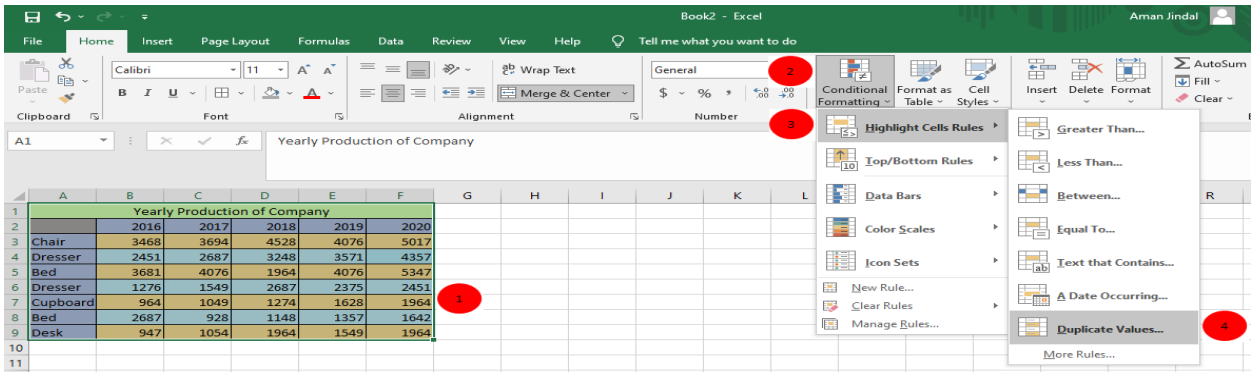


Figure 8.17: Steps to Highlight Duplicate Data

After performing these steps, the duplicate data in the spreadsheet gets highlighted in red as shown in figure 8.18.

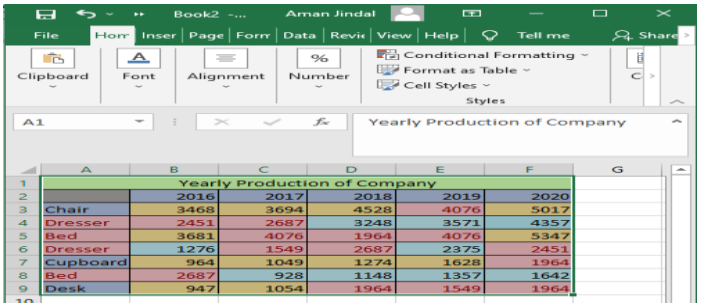


Figure 8.18: Highlighting Duplicate Data

Duplicate data can be removed following the step given below: -

1. Select the data to check for duplicity.
2. In Data tab, click on the Remove Duplicates.
3. Then, a window will appear where one or more than column can be selected to delete the duplicate data.
4. After selecting columns, click on OK button. (See the figure 8.19)

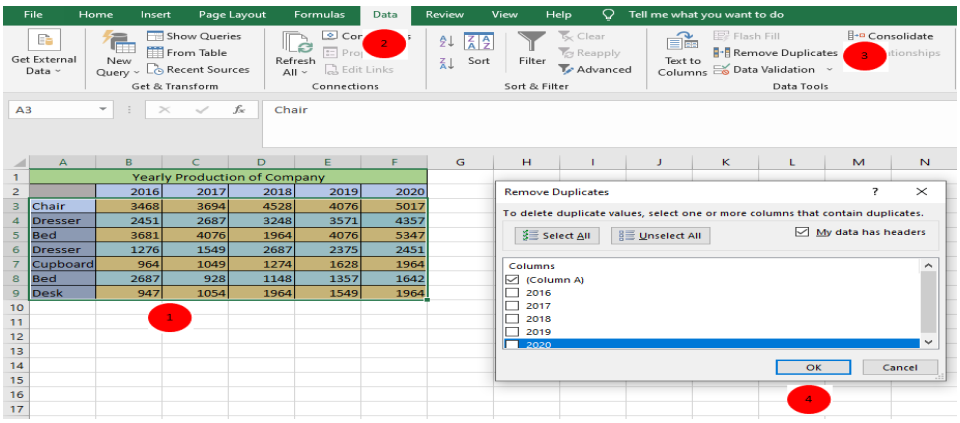


Figure 8.19: Steps to Remove Duplicate Data

The result of the above steps performed is given in the Figure 8.20.

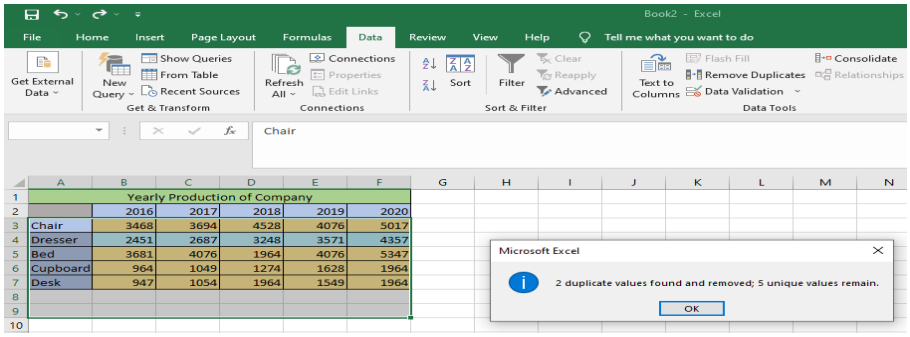


Figure 8.20: Removing Duplicate Data

8.1.6.2 Parse Data using Text to Column

This method converts the selected text into columns on the basis of delimiter or fixed length. It will parse the text and separate the data on the basis of spaces, delimiter like tab, semicolon (;), comma (,), etc. Following steps must be followed to convert the text into columns:

1. Select the text to convert it into columns. Then go to Data tab and click on Text to Columns. A window box will appear as shown in figure 8.21. Here the type of separator is selected. Select Delimited for, tap, commas, etc. and Fixed width for space. After choosing one of the options, click on Next.

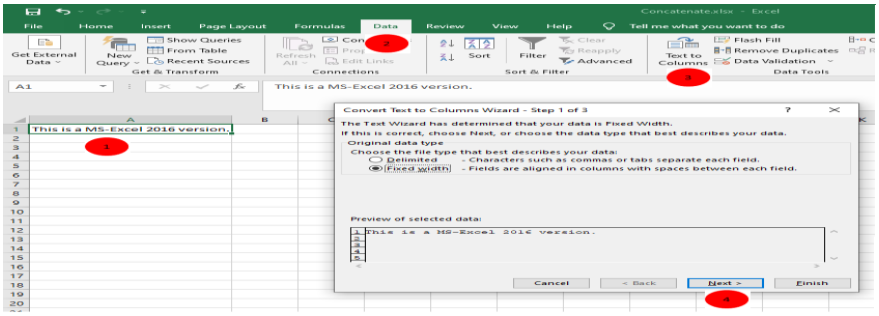


Figure 8.21: Step 1 of conversion

2. After performing step 1, a window will appear showing the preview of the conversion. It can be seen in Figure 8.22. Click on Next to go to the 3rd step of the conversion.

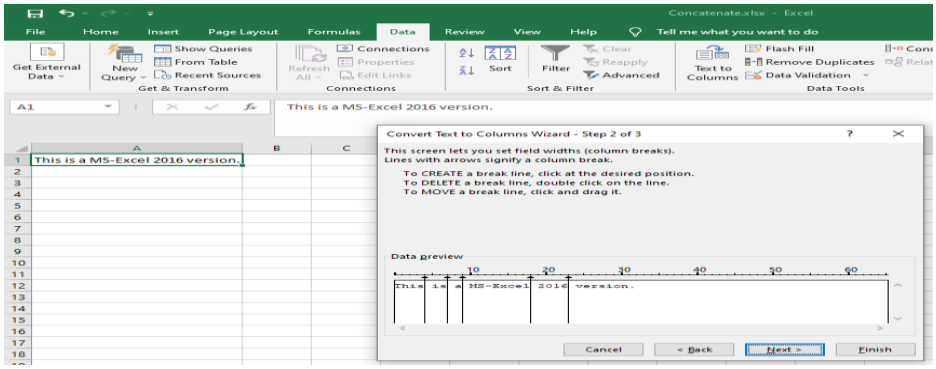


Figure 8.22: Step 2 of conversion

3. This step allows us to format each column or cell going to be formed. See the Figure 8.23. After performing the desired formatting, click on Finish.

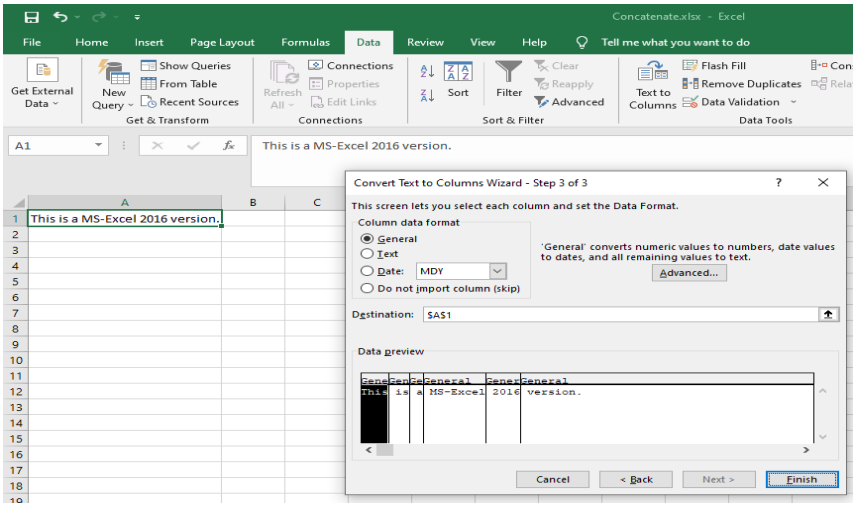


Figure 8.23: Step 3 of conversion

4. Figure 8.24. Shows the conversion of the selected text into columns.

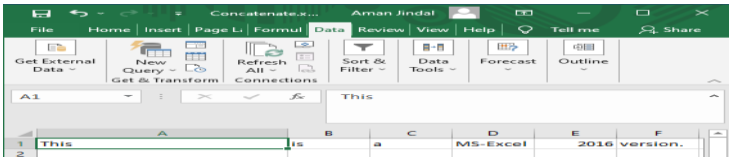


Figure 8.24: Conversion of text to columns

8.2 SUMMARY

- To convert lower case to Upper Case and Upper Case to Lower Case various inbuilt functions has discussed like UPPER, LOWER and PROPER etc.
- To retrieve a specific number of characters from the LEFT, RIGHT or MIDDLE, various functions like MID, LEFT, RIGHT are used.
- Pivot table is the most useful tool allows us to extract the information from a large, complex and detailed dataset.
- Concatenate function is used to merge two or more than two strings.
- Insert Chart option is used to add any type of Chart like Line, Bar, Area chart etc.
- Data can be cleaned by removing duplicate values from the sheet.

8.3 PRACTICE QUESTIONS

1. A Travel Agent table is shown below:

Wise Owl Travel Agents

Country	Resort Name	No of Days	Travel Method	Price	Holiday ID
Australia	Great Barrier Reef	32	Plane	£750	I990AUS
Australia	Perth	28	Plane	£985	AUS112J
Chile	Santiago	21	Plane	£1,259	CH266H
England	London	3	Train	£69	I456UK
England	Bognor	1	Coach	£12	BG726H
France	Lyon	14	Plane	£399	A7995FR
France	Paris - Euro Disney	5	Train	£269	TH789FR
France	Paris - Euro Disney	3	Train	£125	TH788FR

Design a pivot table using above data, then by using filters, to view the average prices of holidays that have either Travel **Method** of **Plane** or a **Resort Name** that starts with the letter G.

2. A list of UK rides is shown in table below:

Open the spreadsheet in the folder above:

Roller Coaster	Amusement Park	Type	Design	Status	Opened	Speed (mph)
Air	Alton Towers	Steel	Flying	Operating	2002	46.6
Boomerang	Pleasure Island Family Theme Park	Steel	Sit Down	Operating	1993	47
Cobra	Paultons Park	Steel	Sit Down	Operating	2006	31.1
Colossus	Thorpe Park	Steel	Sit Down	Operating	2002	45
Corkscrew	Alton Towers	Steel	Sit Down	Operating	1980	40
Corkscrew	Flamingo Land Theme Park & Zoo	Steel	Sit Down	Operating	1983	40
Crazy Mouse	South Pier	Steel	Sit Down	Operating	1998	29.1
Crazy Mouse	Brighton Pier	Steel	Sit Down	Operating	2000	29.1
Enigma	Pleasurewood Hills	Steel	Sit Down	Operating	1995	34
Express	M&Ds Scotland's Theme Park	Steel	Sit Down	Operating	2006	28
Fantasy Mouse	Fantasy Island	Steel	Sit Down	Operating	2000	29.1

Change this data into a pivot table and calculate the overall average speed for taking all rides that satisfy the following conditions:

- The **Type should be Steel**
- The **Design** is to be **Sit Down**
- The A**musement Park** has the word towers somewhere in the title

3. A property portfolio is given below in the table:

PostCode	Type	Location	No Bedrooms	No Bathrooms	Reception Rooms	Garden Size	Date on Market	Date Sold	Asking Price	Sale Price
SK13 7AZ	Detatched	Town	4	2	3	Medium	11/26/2017		£345,000	
SK22 9GT	Semi-detatche	Village	3	1	2	Small	7/18/2017	2/1/2018	£245,000	£238,500
SK13 6DD	Terraced	Countryside	2	1	2	Small	10/24/2017	12/19/2017	£199,000	£199,000
SK14 8DS	Detatched	Town	4	2	2	Large	10/18/2018	1/23/2018	£398,000	£387,500
SK13 7CW	Semi-detatche	Town	3	1	2	Medium	11/29/2017	12/19/2018	£329,000	£319,500
SK22 3YT	Detatched	Remote	4	2	3	Large	10/13/2017		£478,500	
SK13 4DF	Terraced	Town	2	1	2	Small	9/5/2017	1/16/2017	£213,000	£199,500
SK14 7AD	Semi-detatche	Town	3	2	2	Medium	10/29/2017	2/18/2018	£278,500	£277,000
SK13 2AA	Semi-detatche	Village	3	1	2	Large	8/11/2017		£278,500	
SK13 5YY	Terraced	Town	3	2	1	Small	10/30/2017	1/29/2018	£176,500	£174,300
SK14 9FT	Bungalow	Countryside	2	2	2	Medium	11/16/2017	1/13/2018	£223,750	£219,750
SK23 4RF	Flat	Town	1	1	1	None	11/15/2017		£135,000	
SK13 1GG	Terraced	Town	3	1	2	Small	1/5/2018	1/19/2018	£165,900	£168,000
SK13 6YH	Bungalow	Countryside	3	2	2	Large	9/15/2017	12/28/2017	£415,500	£419,500
SK13 6YH	Bungalow	Countryside	3	2	2	Large	9/15/2017	12/28/2017	£415,500	£419,500
SK13 6YH	Bungalow	Countryside	3	2	2	Large	9/15/2017	12/28/2017	£415,500	£419,500
SK13 6YH	Bungalow	Town	2	2	2	Medium	9/11/2017		£199,500	
SK22 8BN	Flat	Town	2	1	1	None	10/3/2017	1/19/2018	£175,500	£169,500
SK14 7JJ	Semi-detatche	Countryside	3	2	2	Medium	12/21/2017	2/15/2018	£319,750	£315,750
SK22 3LP	Bungalow	Remote	3	2	2	Large	10/15/2017		£289,500	
SK13 4DT	Detatched	Countryside	5	2	3	Large	8/9/2017		£525,750	
SK13 9SS	Detatched	Town	4	3	2	Medium	11/14/2017	2/25/2018	£495,000	£495,000
SK14 6HN	Semi-detatche	Town	3	1	2	Medium	8/6/2017	1/15/2018	£369,500	£362,500

Design a pivot table for the property portfolio to display:

- The asking price as the value in the field;
- The type of property to be in the rows;
- The location is to be in the columns;
- The remaining fields is to be in the filter area.

Convert the filters and aggregate functions to display a **count** of properties that have:

- 4 bedrooms;
- A medium garden; and
- 3 bathrooms.

4. Prepare a line chart and pie chart to compare the favorite films data for 26-40 years old only.

	15 - 25 yrs	26 - 40 yrs	Over 40's
Barbarella	17%	31%	18%
Die Hard	20%	15%	1%
Gone with the Wind	4%	19%	41%
Jurassic Park	34%	12%	3%
Speed	17%	8%	11%
Titanic	8%	15%	26%

5. Design a column 2-D chart of the data given below: -

Athens 2004 Medals Table

Country	Gold	Silver	Bronze
USA	35	39	29
China	32	17	14
Russia	27	27	38
Australia	17	16	16
Japan	16	9	12
Germany	14	16	18
France	11	9	13
Italy	10	11	11
South Korea	9	12	9
Great Britain	9	9	12
Cuba	9	7	11
Ukraine	9	5	9
Hungary	8	6	3
Romania	8	5	6
Greece	6	6	4
Norway	5	0	1
Netherlands	4	9	9
Brazil	4	3	3
Sweden	4	1	2
Spain	3	11	5
Canada	3	6	3

Multiple Choice Questions

Q1. For the formula, which symbol used to specify the fixed rows or columns?

a. $

b.;

c. %

d. None of these

Q2. The function which is used within another function is called:

a. SUM Function
b. Nested Function
c. Text Function
d. All of these

Q3.. _____Formatting is used to delete duplicate values

a. Conditional Formatting

b. text Formatting

c. Page Formatting

d. All of these

Q4._____ function is used to extract the characters from the left

a. LEFT

b.. RIGHT

c. M IDDLE

d. None of these

Q5. Which is the visual representation of data in Excel file

a. Graphs

b. Pie Charts

c. Lines

d. None of these

REFERENCES

1. https://www.slideshare.net/ravimishra155/word-excel-pp
2. http://www.tissa.co.za/businesssolutions/index.html
3. https://www.montclair.edu/information-technology/
4. http://docplayer.net/20240723-Introduction-to-word-2007.html
5.https://support.microsoft.com/en-us/office/insert-a-table-of-contents-882e8564-0edb-435e-84b5-1d8552ccf0c0
6.https://cmusr.files.wordpress.com/2016/01/computer-application-1-preparationsheet.pdf
7. http://lib.bbu.edu.az/read.php?file=142&file_type=pdf&item_type=lecture
8. https://link.springer.com/book/10.1007%2F978-1-4302-2950-6
9. https://onlinelibrary.wiley.com/doi/book/10.1002/9781118093955
11. https://ability.com/support/ability2002manual.pdf
12. https://ugv.edu.bd/cbet/pdf/1581073061.pdf
13. https://link.springer.com/book/10.1007%2F978-1-4302-2953-7
14. https://cyber.olympiadsuccess.com/class-6-microsoft-word
15. https://www.scribd.com/document/221490432/How-to-Use-Microsoft-Excel
16.https://communities.geoplatform.gov/disasters/wpcontent/uploads/2018/11/Preliminary-Developer-Guide-and-User-Manual.pdf
17. https://gov.texas.gov/files/disabilities/accessdocs/06-TemplatesStyles.pdf
18. https://excelchamps.com/excel-functions/

ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਪੰਜਾਬ ਸਟੇਟ ਓਪਨ ਯੂਨੀਵਰਸਿਟੀ
ਪਟਿਆਲਾ

The Motto of Our University
(SEWA)
SKILL ENHANCEMENT
EMPLOYABILITY
WISDOM
ACCESSIBILITY

SELF-INSTRUCTIONAL STUDY MATERIAL FOR JGND PSOU

JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY, PATIALA

(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

B.Sc.(Data Science)

Semester I

BSDB31101T

Problem Solving using Computer

Head Quarter: C/28, The Lower Mall, Patiala-147001
Website: www.psou.ac.in

The Study Material has been prepared exclusively under the guidance of Jagat Guru Nanak Dev Punjab State Open University, Patiala, as per the syllabi prepared by Committee of experts and approved by the Academic Council.

COURSE COORDINATOR AND EDITOR:

Dr. Amitoj Singh
Associate Professor
School of Sciences and Emerging Technologies
Jagat Guru Nanak Dev Punjab State Open University

LIST OF CONSULTANTS/ CONTRIBUTORS

Sr. No.	Name
1	Dr. Vinay Kukreja
2	Dr. Virender Kadyan
3	Dr. Amitoj Singh

JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA
(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

PREFACE

Jagat Guru Nanak Dev Punjab State Open University, Patiala was established in December 2019 by Act 19 of the Legislature of State of Punjab. It is the first and only Open University of the State, entrusted with the responsibility of making higher education accessible to all, especially to those sections of society who do not have the means, time or opportunity to pursue regular education.

In keeping with the nature of an Open University, this University provides a flexible education system to suit every need. The time given to complete a programme is double the duration of a regular mode programme. Well-designed study material has been prepared in consultation with experts in their respective fields.

The University offers programmes which have been designed to provide relevant, skill-based and employability-enhancing education. The study material provided in this booklet is self-instructional, with self-assessment exercises, and recommendations for further readings. The syllabus has been divided in sections, and provided as units for simplification.

The University has a network of 10 Learner Support Centres/Study Centres, to enable students to make use of reading facilities, and for curriculum-based counselling and practicals. We, at the University, welcome you to be a part of this instituition of knowledge.

Prof. Anita Gill
Dean Academic Affairs

B.Sc. (Data Science)
Core Course (CC)
Semester I
BSDB31101T: Problem Solving using Computers

Total Marks: 100
External Marks: 70
Internal Marks: 30
Credits: 4
Pass Percentage: 35%

Objective

Objective of this paper is to explain the basic of Python concepts objects, data structures and concepts related to Methods and Functions in python. Paper explicate Object Oriented Programming with Python and comprehend the concepts related to Python Generators and file handling.

INSTRUCTIONS FOR THE PAPER SETTER/EXAMINER

1. The syllabus prescribed should be strictly adhered to.
2. The question paper will consist of three sections: A, B, and C. Sections A and B will have four questions from the respective sections of the syllabus and will carry 10 marks each. The candidates will attempt two questions from each section.
3. Section C will have fifteen short answer questions covering the entire syllabus. Each question will carry 3 marks. Candidates will attempt any ten questions from this section.
4. The examiner shall give a clear instruction to the candidates to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.
5. The duration of each paper will be three hours.

INSTRUCTIONS FOR THE CANDIDATES

Candidates are required to attempt any two questions each from the sections A and B of the question paper and any ten short q uestions from Section C. They have to attempt questions only at one place and only once. Second or subsequent attempts, unless the earlier ones have been crossed out, shall not be evaluated.

Section A

Unit I: **Introduction to Python:** Python installation and setup, Command line Basics; Python Objects and Data Structures Basics: Introduction to Python data types, Variable assignments, Numbers, String, String methods, Lists

Unit II: **Python Comparison Operators:** Chaining comparison operators with logical operators, Pass Break and continue.

Unit III: **Program Flow Control:** If Elif and Else statements in python, for loops, While loops

Unit IV: **Methods and Functions in python:** Introduction to functions, Def keyword, User defined functions, arguments and parameters, Parameter naming in python

Section B

Unit V: Object Oriented Programming: Introduction, Classes and objects, attributes and methods, Inheritance and polymorphism, Special methods; Modules and Packages: Pip install and PyPi.

Unit VI: Errors and Exception Handling: Introduction to errors, Built-in errors, raising errors in python, Pylint overview

Unit VII: Python Generators: Yielding and Generator function, Making an iterable from a generator, Generator expressions and performance.

Unit VIII: File handling in Python: Files in python, importing own files, Read and writing text files, working with CSV, XML and JSON files.

Suggested Readings

1. Timothy Budd, Exploring Python, TMH, 1st Ed, 2011
2. Allen Downey, Jeffrey Elkner, Chris Meyers , How to think like a computer scientist : learning with Python , Green Tea Pr, 2002
3. Paul Barry, Head First Python: A Brain-Friendly Guide, O′Reilly, 2nd ed. 2016
4. Udemy, https://www.udemy.com/course/complete-python-bootcamp/
5. Udemy, https://www.udemy.com/course/python-the-complete-python-developer-course/

JAGAT GURU NANAK DEV PUNJAB STATE OPEN UNIVERSITY, PATIALA
(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)

BSDB31101T PROBLEM SOLVING USING COMPUTER
COURSE COORDINATOR AND EDITOR: DR. AMITOJ SINGH

UNIT NO.	UNIT NAME
UNIT 1	INTRODUCTION TO PYTHON
UNIT 2	PYTHON COMPARISON OPERATORS
UNIT 3	PROGRAM FLOW CONTROL
UNIT 4	METHODS AND FUNCTIONS
UNIT 5	OBJECT ORIENTED PROGRAMMING
UNIT 6	ERRORS AND EXCEPTION HANDLING
UNIT 7	PYTHON GENERATORS
UNIT 8	FILE HANDLING

`

B.Sc.(DATA SCIENCE)

SEMESTER-I

PROBLEM SOLVING USING COMPUTERS

UNIT I: INTRODUCTION TO PYTHON

STRUCTURE

1.0 OBJECTIVES

- Learn the fundamentals of installing Python in different Operating Systems
- Work with different data types of Python
- Get deep insights about how to work with different data structures like lists, tuples, sets and dictionary.
- Students will be able to make programs related to the above-mentioned concepts.

1.1 INTRODUCTION

This module targets to inculcate python basics, data structures, and its operations in user minds so that they can build basic programs by using these concepts extensively. This module is developed by keeping the graduate learners in mind. Python is a very user-friendly language and is widely used by professional developers in different applications like artificial intelligence, web applications, and data analytics, etc. Python is recognized as one of the fastest-growing languages among different types of users. In the module, how to install python on different platforms is elaborated along with basic data types with appropriate programs so that learners will understand the concept. The main target of this module to cover data structures like Lists, tuples, dictionary, and sets with suitable programs.

1.2 PYTHON

Guide van Rossum (Dutch Programmer) is the creator of the Python programming language and it was invented in the year 1989. Its first public release was in 1991. Python is managed and distributed by Python Software Foundation. Python is very user-friendly, free to use and easy to learn if you are fresher in programming too [1]. This is popularly known as one of the powerful languages and open-source. Python is an interpreted and high-level language. Programmers have to focus on what to do in the tasks instead of how to do the tasks.

1.3 PYTHON USAGE

Python is used for the development of various applications in different fields. Some of these are mentioned below:

a) Web programming
b) Mobile Applications
c) Game development
d) Data Science and Data Visualization
e) Data Analytics
f) Frontend (GUI) development
g) Network programming
h) System Administrator
i) Machine learning and Artificial Intelligence
j) Web scrapping applications
k) Embedded applications

1.4 DEVELOPMENT ENVIRONMENT

There are three popular and common platforms. These are:

a) Terminal based or Shell-based
b) IDLE (Spyder IDE, PyCharm IDE)
c) Python notebook (Jupyter)

1.5 PYTHON INSTALLATION AND SETUP

→ If python is installed you can check the version of Python installed in your system by the following command.

C:\> python –version

→ To know the current location where your python is installed, you can use the below-mentioned command

C:\> where.exe python

If Python is not installed in your system then you have to install python according to your operating system. Below are the steps for installation according to your operating system.

1.5.1 Python installation on Windows

There are different ways of installing Python on windows and these are listed [2] as:

A) Package from Microsoft Store
B) Full Installer Package
C) Windows Subsystem for Linux

A) Package from Microsoft Store

One of the easiest, upfront and interactive ways of installing python on windows is with the help of app from the Microsoft store. If any of the beginners want to install Python, this is the only suggested way. Select the python version which you want to install on your machine and then click *GET* button on that suitable version, as it helps in downloading the python file. After that click on *INSTALL ON MY DEVICES* and select the device where you want to install the software. Now, select and click the button *INSTALLNOW*, after proper installation, you will see a message of installation congratulations.

B) Full Installer Package

If you are an advance or intermediate developer then this way of downloading from Python.org is a recommended step as it helps to control the things during installation set up. Go to the site and select the latest Python 3 release under Python Release for Windows and also select Windows x86-32 or Windows x86-64 executable installer and download the appropriate version. Now, double-click the downloaded file and look for the different options on the dialog box that appears after double-clicking the installer file. The first one is the default path which is set for

current Windows users. The second feature is to select for customization installation where users can select the features which they want to install like pip and IDLE also. The third one is to install a launcher for all users. This has a checkbox that is ticked default. It means all users can access py.exe whereas if you untick this checkbox, then only current can access py.exe. The fourth feature is to Add python to the path. This has a checkbox that is unticked by default and it is up to the user whether the user wants to add python in the environment path or not. After working on these, you can go to click on the button *INSTALL NOW* and after proper installation, you will see a message of installation congratulations.

C) Windows Subsystem for Linux (WSL)

You can run a virtual Linux in Windows system directly with the help of WSL. Python can be used with the help of the Linux platform here.

1.5.2 Python Installation on Mac

There are different ways of installing Python on macOS and these are listed [2] as:

A) Package from the official installer
B) Package Manager (Homebrew)

A) Package from the official installer

The most recommendable way of downloading is Python.org as it helps to control the things during installation set up. Go to the site and select the latest Python 3 release under Python Release for Mac OS X and also select macOS 64 bit executable installer and download the appropriate version. Now, double click the downloaded file and click *continue* button a few times until you reach software agreement and then click *Agree* button and you will see a dialog box where destination location with total space is shown and the user can change the destination location and then click on *INSTALL NOW* button and after proper installation, you will see a message of installation congratulations.

B) Package Manager (Homebrew)

First, install homebrew package manager unless ignore if it is already in the system. Go to browser and open it and type http://brew.sh/ and copy the command for install homebrew from there and open a Window terminal and paste that command and it will start homebrew installation process and suitably type your macOS password when it is asked in the installation process. This will take few minutes and after successful installation of this package manager, install python by using the command brew install python3.

1.5.3 Python Installation on Linux

There are different ways of installing Python on Linux and these are listed [2] as:

A) Package manager of machine OS

B) Building from source code

A) Package manager of machine OS

One of the easiest and popular methods for installing python on the Linux platform. This is done by running a command on the command line.

B) Building from source code

This is a typical harder method of installation when compared with the package installer method. This step involves a set of commands for installing python along with that take care of dependencies that are required for properly running the python code.

1.6 COMMAND LINE BASICS

The command-line interface is a text-based interpreter that helps in the interaction of the user with a running program. Executable commands are written on this and the appropriate function is done by the operating system according to the command. Python executable command is run by writing python program name in front of keyword python [3].

Example1: Message printing example and save the program as niceprog1.py (.py is the extension of python)

```
print("CSE-CA")

print("Mechatronics")
```

Now, how to run this program with the help of the command line and what will be the output of that command line.

Run the following command and press enter at end of the command:

```
C:\User\Python>python niceprog1.py
```

 # The output will be after running the above command is

```
CSE-CA

Mechatronics
```

If you want some data elements should be passed to the Python program using the command line then pass data elements values with python file name by putting up space as delimiter. These data elements that are used with space are known as command-line arguments.

Example 2: Command Line Arguments Program (Save this file as niceprog2.py)

```
import sys

print("The 1st argument in Command Line is",sys.argv[1])

print ("The 2nd argument in Command Line is",sys.argv[2])
```

Run the following command and press enter at end of the command:

C:\User\Python>python niceprog2.py “CSE-CA” “Mechatronics”

 # The output will be after running the above command is

The 1st argument in Command-Line is CSE-CA

The 2nd argument in Command-Line is Mechatronics

1.7 VARIABLES

Variables store the data and these are reserved memory spaces. There is no command available to declare variables unlike C/C++/Java has commands to declare variables. Here in python, only the ‘=‘operator is used for assigning value to a variable.

Example 3:

```
str_BDay_Boy_Name="Vinay Stylish"  #String variable

float_BDay_Boy_Age=21.6 #Float Variable

int_Bday_Boy_License_Num=12234562#Integer Variable

print(str_BDay_Boy_Name)

print(float_BDay_Boy_Age)

print(int_Bday_Boy_License_Num)
```

#Output

```
Vinay Stylish
21.6
12234562
```

NO7E: # is used to comment the line. # is used for single-line comment

1.7.1 Variables Reassignment

A variable can be assigned a value many times but the value that is assigned latest is considered for consideration.

Example 4: Variables Reassignment

```
float_BDay_Boy_Age=21.6 #Float Variable

float_BDay_Boy_Age=43.98

print(float_BDay_Boy_Age)
```

#Output

43.98

1.7.2 Multi-variable Assignment

When different variables are assigned with a same value that is the concept of multi-variable assignment.

Example 5: Multi-variable Assignment

```
int_var1=int_var2=int_var3=156 # Multivariable assignment

print(int_var1)

print(int_var2)

print(int_var3)

print(id(int_var1))  # id is used to get the memory address of the variable

print(id(int_var2))

print(id(int_var3))

int_var3=20

print(int_var3)

print(id(int_var3))
```

#Output

```
156
156
156
8790917690608
8790917690608
8790917690608
20
8790917686256
```

NOTE: Same value is pointing to the same reserved memory and variables int_var1, int_var2 and int_var3 having the same reserved memory as they point towards the same value. When the value is changed to 20 of int_var3, its memory address is also changed.

1.8 BASIC DATA TYPES

Data types are associated with every value. In object-oriented python, everything is treated as object. Objects are variables and classes are data types of these variables [4]. The basic data types are mentioned as:

1.8.1 Numeric
   i) Integer
   ii) Float
   iii) Complex
1.8.2 Strings
1.8.3 Boolean

1.8.4 Numeric

The numeric value is associated with numeric data type in python programming. The numeric data type can be integer, float or complex numbers. They are presented as int, float or complex respectively.

Integer numbers – These are whole numbers and can be positive or negative. No limit is restricted to how long can be an integer number. The memory is the only constraint for setting up the integer value.

Float numbers – These are real numbers with a fraction or decimal points. For scientific notation, e or E is used with integer numbers whether they are positive or negative.

Complex numbers – It consists of real and imaginary part.

Example 6: Numeric data type example

```
float_BDay_Boy_Age=21.6 #Float Variable

int_Bday_Boy_License_Num=12234562#Integer Variable

complex_Bday_Gift=2+5j # Complex number

print(type(float_BDay_Boy_Age))

print(type(int_Bday_Boy_License_Num))

print(type(complex_Bday_Gift))
```

#Output

```
<class 'float'>
<class 'int'>
<class 'complex'>
```

Example 7: Binary, Octal and Decimal Numbers Representation and their data type.

```
octal_var1=0o12

hexa_var2=0x16

binary_var3=0b1011
```

```
print(octal_var1)

print(type(octal_var1))

print(hexa_var2)

print(type(hexa_var2))

print(binary_var3)

print(type(binary_var3))
```

#Output

```
10
<class 'int'>
22
<class 'int'>
11
<class 'int'>
```

Example 8: Deep insights of floating points

```
float_var1=.56e7

float_var2=56.2e-3

print(float_var1)

print(float_var2)
```

#Output

```
5600000.0
0.0562
```

1.8.2 Strings

They are a group or sequence of characters and strings data types are represented as str. The use of single quotes and double quotes are used for strings [5].

Example 9: Usage of single quote and double quotes for printing strings

```
print('Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID') # Usage of Single quotes

print("Do follow social distancing") # Usage of double quotes

print('Properly use mask')

print("Avoid unnecessary shopping and walking-out")
```

#Output

Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID
Do follow social distancing
Properly use mask
Avoid unnecessary shopping and walking-out

Example 10: Another usage of single quotes and double quotes

```
print('Take both vaccination(")s with a normal 4 to 6 weeks gap and say bye to COVID') # Usage of Single quotes

print("Do follow(')s social distancing") # Usage of double quotes

print('Take both vaccination\"s with a normal 4 to 6 weeks gap and say bye to COVID') # Usage of Single quotes

print("Do follow\'s social distancing") # Usage of double quotes
```

#Output

Take both vaccination(")s with a normal 4 to 6 weeks gap and say bye to COVID
Do follow(')s social distancing
Take both vaccination"s with a normal 4 to 6 weeks gap and say bye to COVID
Do follow's social distancing

Example 11: Usage of triple single quotes for printing single and double quotes in one print statement.

```
print('''Take both vaccination(')s with a normal 4 to 6 weeks gap and say bye(") to COVID. Do follow social distancing''')
```

#Output

Take both vaccination(')s with a normal 4 to 6 weeks gap and say bye(") to COVID. Do follow s ocial distancing

Example 12: Usage of triple double quotes for printing single and double quotes in one print statement.

```
print("""Take both vaccination(')s with a normal 4 to 6 weeks gap and say bye(") to COVID. Do follow social distancing""")
```

#Output

Take both vaccination(')s with a normal 4 to 6 weeks gap and say bye(") to COVID. Do follow s ocial distancing

1.8.2.1 String Methods

i) String slicing

Accessing characters from the string is called string slicing. For slicing, the colon ':' is used. For better understanding see example 12A.

ii) Updating strings

The whole string can be updated but the characters of strings cannot be updated. For better understanding see example 12A.

iii) Deleting strings

del keyword is used to delete the whole string as shown in example 12A.

Example 12A: Slicing, Updating and deleting operations in string

```
str_var='Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID'

print(str_var)

print(str_var[5:14]) # Print elements from 5 to 13 and exclude 14

print(str_var[-12:-2])

# Print elements from -12 to -1 and exclude -1.

#At last, index is -1 value is D and -2 is I and -12 is b

str_var='Hey COVID, you will be completely over by 2022. I think!!!'

print(str_var)

str_var2='Hey COVID, you will be completely over by 2022. I think!!!'

print(str_var2)

del str_var2 # Deleted str_var2

str_var[0]='Z' # This cannot be done in string
```

#Output

```
Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID
both vacc
bye to COV
Hey COVID, you will be completely over by 2022. I think!!!
Hey COVID, you will be completely over by 2022. I think!!!
TypeError: 'str' object does not support item assignment
```

iv) Formatting Method

format() method is used to do formatting in strings and {} braces are used for formatting and they act as placeholders for arguments. These can be positional as well as keyword arguments. See example 12B for the usage of the format() method.

Example 12B: Program to show deeper understanding of format method

```
str_var1="The first dose of {0} and second dose of {1} saves lives from attack of COVID {2}".format("injection1","injection2",2019)

print(str_var1)

print(type(str_var1))

str_var2="The first dose of {} and second dose of {} saves lives from attack of COVID {}".format("injection1","injection2",2019)

print(str_var2)

print(type(str_var2))

str_var3="The first dose of {val1} and second dose of {val2} saves lives from attack of COVID {val3}".format(val1="injection1",val2="injection2",val3=2019)

print(str_var3)

print(type(str_var3))

str_var4="{0:.3f}".format(2/7)  # Here.3f tell how many decimal points you need

print(str_var4)

print(type(str_var4))

str_var5="{0:o}".format(54) # Convert into octal format

print(str_var5)

print(type(str_var5))

str_var6="{0:b}".format(54) # Convert into binary format

print(str_var6)

print(type(str_var6))
```

#Output

```
The first dose of injection1 and second dose of injection2 saves lives from attack of COVID 201
9
<class 'str'>
```

The first dose of injection1 and second dose of injection2 saves lives from attack of COVID 201
9
<class 'str'>
The first dose of injection1 and second dose of injection2 saves lives from attack of COVID 201
9
<class 'str'>
0.286
<class 'str'>
66
<class 'str'>
110110
<class 'str'>

1.8.3) **Boolean**

This is one of the inbuilt data types in python programming and this data type can take only two values i.e. True or False. When checking of type of a variable is performed and it shows <'class bool'> then it is Boolean data type [6].

Example 13: Boolean Basic Understanding Program

```
var_bool1=True

var_bool2=False

print(type(var_bool1))

print(type(var_bool2))

print(type(var_bool1==var_bool2))
```

#Output

```
<class 'bool'>
<class 'bool'>
<class 'bool'>
```

NOTE: bool() function can be used to get the values in form of True or False

Example 14: bool() function usage

```
int_var1 = 19

print(bool(int_var1))

int_var2 = 0

print(bool(int_var2))
```

```
float_var3 = 17.89

print(bool(float_var3))

float_var4 = -17.89

print(bool(float_var4))
```

#Output

```
True
False
True
True
```

Explanation: Any value other than 0 is treated as True in the above example of bool() unless it is treated as False.

1.9 BASIC DATA STRUCTURES IN PYTHON

Data Structures help in managing, handling, dealing and storing the data. They help in traversing, modifying and updating the data. Different operations can be performed on the data with the help of data structures. This helps in controlling the functionalities involved in getting the proper data output.

The basic data structures of python are:

1.9.1 Lists
1.9.2 Tuples
1.9.3 Dictionary
1.9.4 Sets

1.9.1 Lists

This data structure helps in storing different data types. This is created using a square bracket []. The data types can be strings, other lists or integers. The list elements can be changed even after its creation. This is known as mutable property. Addresses are assigned to every element of the list, this is known as an index of the list. Two types of indexes are there one is positive and the other one is negative. While traversing in a list when the user starts from 0 to the last element of the list, it is called positive index. When the user traverse from the last (i.e. index=-1) to the start is called negative-index [7]. The different basic operations that can be performed on the list are mentioned as:

- The list can be created empty as well as initialized with different types of elements.
- Elements can be inserted in list using append(), extend() and insert() functions. These functions usage is explained in example 15.

- Elements can be deleted in the list using remove(), pop() and remove() functions. The del keyword is also used to delete elements in the list. The explanation is given in example 16.
- Accessing elements of the list is explained in example 17.

Example 15: Program for List creation (empty, initialization with elements) and Insertion in list (Usage of insert, extend and append functions) [7]

```
mixed_list1=[]   # Empty list creation

print(type(mixed_list1)) # Check the data type

print(mixed_list1) # Printing the empty list

mixed_list2 = [2,"injection1",23.56,"injection1",23.56]

# Adding heterogeneous elements to the mixed_list2

print(type(mixed_list2)) #Check the data type

print(mixed_list2) # Printing the mixed_list2 elements

mixed_list1.extend([2,"injection1"])

#Extend function is used to add elements one by one in the empty list (mixed_list1)

print(mixed_list1)

mixed_list1.append([23.56,"Wear Mask Properly","Rules for Social Distancing"])

#Append function is used to add all elements as single element in mixed_list1 and added at end of list

print(mixed_list1)

# INDEXING IN LIST STARTS FROM 0

# Adding element at INDEX 2 i.e. after value of injection1

mixed_list1.insert(2,"injection2") # injection2 is inserted at index 2

print(mixed_list1)
```

#Output

```
<class 'list'>
[]
<class 'list'>
[2, 'injection1', 23.56, 'injection1', 23.56]
[2, 'injection1']
```

[2, 'injection1', [23.56, 'Wear Mask Properly', 'Rules for Social Distancing']]
[2, 'injection1', 'injection2', [23.56, 'Wear Mask Properly', 'Rules for Social Distancing']]

Example 16: Program for deleting elements in the list using different ways [7]

```
mixed_list1=[2, 'injection1', 'injection2', 23.56, 'Wear Mask Properly', 'Rules for Social Distancing']

print(mixed_list1)

# Remove function is used to delete the elements using value and it deletes the first occurrence of value

mixed_list1.remove('Wear Mask Properly') # 'Wear Mask Properly' element is removed from the list

print(mixed_list1)

# Pop is used to remove the element from the list by passing index as the argument in pop function.

# If  index is not given then last element of the list is deleted

pop_element=mixed_list1.pop(3) # Element is removed at index 3 using pop function

print("popped element is",pop_element)

print(mixed_list1)


#del keyword is used to delete the element using the index value

del mixed_list1[2] # Element is deleted at index 1

print(mixed_list1)

# Clear function is used to remove the whole list

mixed_list1.clear()

print(mixed_list1)
```

#Output

[2, 'injection1', 'injection2', 23.56, 'Wear Mask Properly', 'Rules for Social Distancing']
[2, 'injection1', 'injection2', 23.56, 'Rules for Social Distancing']
popped element is 23.56
[2, 'injection1', 'injection2', 'Rules for Social Distancing']

[2, 'injection1', 'Rules for Social Distancing']
[]

Example 17: Program to show ways of accessing list elements

```
mixed_list1=[2, 'injection1', 'injection2', 23.56, 'Wear Mask Properly', 'Rules for Social Distancing']

print(mixed_list1) # Accessing all list elements

print(mixed_list1[5])

print(mixed_list1[3:5])

# Slicing Method is used here, [starting element index : end element range-1].

# [3:5] here denotes go from index 3 to 5-1 i.e. 4

# Access elements from index 3 to 4 and exclude 5 (Last element of range is excluded)

print(mixed_list1[-4:-2])

# When access from last, index values from last are -1,-2,-3 and so on

# When index is -4,by calculating from last it is "injection2"

# Index is -2,by calculating from last it is "Wear Mask Properly"

# Range is -4 to -2 i.e Access elements from -4 to -3 and exclude -2 (Last element of range is excluded)

# See below access elements of list using loop

for mixed_var1 in mixed_list1: # one by one elements of list are accessing

    print(mixed_var1)
```

#Output

```
[2, 'injection1', 'injection2', 23.56, 'Wear Mask Properly', 'Rules for Social Distancing']
Rules for Social Distancing
[23.56, 'Wear Mask Properly']
['injection2', 23.56]
2
injection1
injection2
23.56
Wear Mask Properly
Rules for Social Distancing
```

1.9.2 Tuples

Tuples exactly work like lists of python but they are immutable i.e. it elements cannot be changed. They are created using () or tuple() function. They also store heterogeneous elements. While creating () are optional as can be seen in example 18. The different basic operations that can be performed on the tuple are mentioned as [7]:

- Tuple can be created empty as well as initialized with different types of elements. It can be created with () brackets, without () brackets and tuple() function. Tuple() function has one optional argument that consists of an iterator like lists, tuples, dictionary, etc. See example 18 for a better understanding.
- Tuple concatenation using '+' operator. An example is shown in 18.
- Tuple elements cannot be removed or deleted due to their immutable property. Immutable property is explained in example 19. But, the whole tuple can be deleted using 'del' keyword as shown in example 20.
- Slicing method used in tuples as mentioned in example 21.

Example 18: Tuple elements creation and accessing program

```
mixed_tuple1= ()  # Empty tuple creation

print(type(mixed_tuple1)) # Check the data type

print(mixed_tuple1) # Printing the empty tuple

# Adding heterogeneous elements to the mixed_tuple2

mixed_tuple2 = (2,"injection1",23.56,"injection1",23.56)

print(type(mixed_tuple2)) #Check the data type

print(mixed_tuple2) # Printing the mixed_tuple2 elements

# Without bracket creating tuple elements

mixed_tuple3 = 2,"injection1",23.56,"injection1",23.56

print(type(mixed_tuple3)) #Check the data type

print(mixed_tuple3) # Printing the mixed_tuple3 elements

# For concatenating  elements in tuple '+' is used

mixed_tuple3=mixed_tuple3 + ("bye_covid",576,"huge effort required")

print(mixed_tuple3) # Printing the mixed_list2 elements

# Creation of tuple elements using tuple function

mixed_tuple4= tuple()  # Empty tuple creation with no argument in tuple() function
```

```
print(type(mixed_tuple4)) # Check the data type

print(mixed_tuple4) # Printing the empty tuple

# tuple() is used when list is passed as argument

mixed_list =[2,3,17,67]

mixed_tuple4= tuple(mixed_list)  # In tuple()function, list is passed as argument

print(type(mixed_tuple4)) # Check the data type

print(mixed_tuple4) # Printing the empty tuple
```

#Output

```
<class 'tuple'>
()
<class 'tuple'>
(2, 'injection1', 23.56, 'injection1', 23.56)
<class 'tuple'>
(2, 'injection1', 23.56, 'injection1', 23.56)
(2, 'injection1', 23.56, 'injection1', 23.56, 'bye_covid', 576, 'huge effort required')
<class 'tuple'>
()
<class 'tuple'>
(2, 3, 17, 67)
```

Example 19: Program to show immutable property of tuples

```
mixed_tuple2 = (2,"injection1",23.56,"injection1",23.56)

mixed_tuple2[1]="vaccination1"  # This cannot be done as it tries to change element value of tuple.

print(mixed_tuple2)
```

#Output

TypeError: 'tuple' object does not support item assignment

Example 20: Program to delete the whole tuple using ‘del’ keyword

```
mixed_tuple2 = (2,"injection1",23.56,"injection1",23.56)

print(mixed_tuple2)

del mixed_tuple2

print(mixed_tuple2)
```

#Output

(2, 'injection1', 23.56, 'injection1', 23.56)

NameError: name 'mixed_tuple2' is not defined

Example 21: Program for slicing and accessing elements of tuples

```
mixed_tuple2 = (2,"injection1",23.56,"injection1",23.56)

print(mixed_tuple2[1:3])

print(mixed_tuple2[-4:-1])

print(mixed_tuple2[:])

print(mixed_tuple2[:3])

print(mixed_tuple2[::-1])

for mixed_var1 in mixed_tuple2:

    print(mixed_var1)
```

#Output

```
('injection1', 23.56)
('injection1', 23.56, 'injection1')
(2, 'injection1', 23.56, 'injection1', 23.56)
(2, 'injection1', 23.56)
(23.56, 'injection1', 23.56, 'injection1', 2)
2
injection1
23.56
injection1
23.56
```

1.9.3 Dictionary

In python programming, the dictionary can be created using {} brackets and it consists of keys and values where two keys cannot be the same in a dictionary but values can be repeated and of different datatype in a dictionary, Keys are immutable. Many keys and values can be there in the dictionary but they are separated using a comma operator as shown in syntax. Nesting of dictionaries is also possible like lists.

Syntax:

Dictionary_name = { key1:value, key2:value ……. Key n:value}

The different basic operations that can be performed on the dictionary are mentioned as [7]:

- Dictionary creation and accessing elements as mentioned in example 22. Empty dictionary and elements in the dictionary are created using {} and dict() function is also used to create a dictionary. Accessing elements of dictionary can be used keys and get() function.
- Updation and insertion elements in the dictionary as shown in example 23.
- Deletion of elements in a dictionary using del keyword, clear() function, pop() function as shown in example 24.
- Nesting of dictionaries as shown in example 25.

Example 22: Creation of dictionary

```
base_dnary={} # Empty dictionary creation
print(base_dnary)
mixed_dnary = {'BdayBoyName':'XYZee', 2:'injections', 2019:'Year'}
print(mixed_dnary) # Whole dictionary is printed
print(mixed_dnary[2])# key is 2nd and its value is printed
print(mixed_dnary['BdayBoyName'])#Key value of 'BdayBoyName' is printed
print(mixed_dnary.get(2)) # get method is used to access value
mixed_dnary2=dict({'BdayBoyName':'XYZee', 2:'injections', 2019:'Year'})
print(mixed_dnary2)
print("Using keys() method")
for key_elements in mixed_dnary2.keys():  # Accessing elements using keys() method
  print (key_elements, mixed_dnary2[key_elements])
print("Using items() method")
for key_elements, val_elements in mixed_dnary2.items(): # items() method
  print (key_elements, val_elements)
print(mixed_dnary[0])
#Generate error as key 0 is not present and it does not take index 0 into consideration
```

#Output

```
{}
{'BdayBoyName': 'XYZee', 2: 'injections', 2019: 'Year'}
```

injections
XYZee
injections
{'BdayBoyName': 'XYZee', 2: 'injections', 2019: 'Year'}
Using keys() method
BdayBoyName XYZee
2 injections
2019 Year
Using items() method
BdayBoyName XYZee
2 injections
2019 Year
KeyError: 0

Example 23: Program for inserting and updating elements in dictionary [8]

```
mixed_dnary = {'BdayBoyName':'XYZee', 2:'injections', 2019:'Year'}

print(mixed_dnary)

mixed_dnary[2]='doses of injections' # Updating elements using key

print(mixed_dnary)

mixed_dnary['Mask']='Wear properly' # Insert new key and value in dictionary

print(mixed_dnary)
```

#Output

{'BdayBoyName': 'XYZee', 2: 'injections', 2019: 'Year'}
{'BdayBoyName': 'XYZee', 2: 'doses of injections', 2019: 'Year'}
{'BdayBoyName': 'XYZee', 2: 'doses of injections', 2019: 'Year', 'Mask': 'Wear properly'}

Example 24: Program to show deletion of elements in a dictionary

```
mixed_dnary = {'BdayBoyName': 'XYZee', 2: 'doses of injections', 2019: 'Year', 'Mask': 'Wear properly'}

mixed_dnary2= {'BdayBoyName': 'XYZee', 2: 'doses of injections', 2019: 'Year', 'Mask': 'Wear properly'}

print(mixed_dnary)

print(mixed_dnary2)

del mixed_dnary['Mask'] # deleting a particular key and value using 'key'
```

```
print(mixed_dnary)

mixed_dnary2.pop(2019) # pop(key) put key value as argument to delete key and value from dictionary

print(mixed_dnary2)

mixed_dnary2.popitem() # popitem() randomly delete any key and value from dictionary

print(mixed_dnary2)

mixed_dnary2.clear()# Removing all elements of 2nd dictionary using clear function

print(mixed_dnary2)

del mixed_dnary # Whole dictionary is deleted

print(mixed_dnary)
```

#Output

```
{'BdayBoyName': 'XYZee', 2: 'doses of injections', 2019: 'Year', 'Mask': 'Wear properly'}
{'BdayBoyName': 'XYZee', 2: 'doses of injections', 2019: 'Year', 'Mask': 'Wear properly'}
{'BdayBoyName': 'XYZee', 2: 'doses of injections', 2019: 'Year'}
{'BdayBoyName': 'XYZee', 2: 'doses of injections', 'Mask': 'Wear properly'}
{'BdayBoyName': 'XYZee', 2: 'doses of injections'}
{}
NameError: name 'mixed_dnary' is not defined
```

Example 25: Program to show the concept of nesting of dictionary

```
mixed_new_dnary1={'Trade':'CSE-CA',2:123,'E2':234,4:{'1st':'basic','2nd':888,3:'style'}}

print(mixed_new_dnary1)

print(mixed_new_dnary1[2]) # Accesing value when key is 2

print(mixed_new_dnary1[4]['1st']) # Accesing value when key is '4' and again its key is 1st

print(mixed_new_dnary1[4]) # Accessing value even if it is dictionary while using key '4'
```

#Output

```
{'Trade': 'CSE-CA', 2: 123, 'E2': 234, 4: {'1st': 'basic', '2nd': 888, 3: 'style'}}
123
basic
{'1st': 'basic', '2nd': 888, 3: 'style'}
```

1.9.4 Sets

Sets are a collection data structure and these are unordered (i.e. elements cannot be accessed using the index), mutable. Sets do not contain duplicate elements i.e. they are unique.

Syntax:

mixed_set = set(iterable_object)

where iterable_object can be strings, list, tuple and any other iterable object.

Example 26: Program to understand basic concept of sets

```
mixed_set=set(["injection1","Wear Mask",2, "Social Distancing"])
print(mixed_set) # Look at its output, it is unordered
print(type(mixed_set))
mixed_set.add("injection2") # Add new element in set
print(mixed_set) # Look at its output, it is unordered
print(len(mixed_set)) # len function is used to know the length of set
```

#Output

```
{'Wear Mask', 2, 'Social Distancing', 'injection1'}
<class 'set'>
{2, 'Social Distancing', 'injection1', 'injection2', 'Wear Mask'}
5
```

1.10 SELF-CHECK QUESTIONS

A) Which of the options are true for the following statements:

Statement 1: List is mutable

Statement 2: Tuple is mutable

Statement 3: List is immutable

Statement 4: Tuple is immutable

a) Statement 1 and Statement 2 are correct
b) Statement 1 and Statement 4 are correct
c) Statement 2 and Statement 3 are correct
d) All are correct

B) What will be the output of the following code?

```
print(("CSE-CA ")+("Mechatronics"))
```

a) CSE-CA Mechatronics
b) CSE-CAMechatronics
c) Mechatronics
d) CSE-CA

C) Choose the correct option where indexing is not valid for doing opertaions.

a) Lists
b) Dictionary
c) Strings
d) Tuples

D) What will be the output?

```
str_var4="{0:.3f}".format(8/11)
print(str_var4)
```

a) 0.727
b) 0.728
c) 0.726
d) 0.730

E) Print the appropriate output of the following code

```
mixed_list1=[2, 'injection1', 'injection2', 23.56, 'Wear Mask Properly', 'Rules for Social Distancin
g']
print(mixed_list1[-3])
```

a) 23.56
b) injection2
c) Rules for Social Distancing
d) Wear Mask Properly

1.11 SUMMARY

This module helps the students in understanding how different types of variables are used and what type of operations can be performed on these different variables along with appropriate examples. Deep discussions about lists, tuples, dictionary, and sets have been done. Different operations related to these data structures have been explained along with the output of different examples. This module helps the students in building up the basic blocks of python that are required for doing hard problems or competitive problems at later stages.

1.12 UNIT END QUESTIONS

1) Construct a program to add two lists of the same size having integer numbers.

2) Which is the correct option for the following code (Note: Read carefully)

```
mixed_tuple=[55,34,89,165]
```

mixed_tuple.pop(2)

print(mixed_tuple)

a) Error

b) [55, 34, 165]
c) [55, 34, 89]
d) [55, 89, 165]

3) Correct way of selecting 67 as output from the code is:

mixed_tuple = ("COVID_19", [13, 25, 36, 67], (15, 675, 543))

a) print(mixed_tuple[1][3])
b) print(mixed_tuple[2][3])
c) print(mixed_tuple[2][2])
d) print(mixed_tuple[1][2])

4) List down the operation that can be performed on dictionary with appropriate examples.

5) A tuple in python can be created without () brackets (True/False)

6) What will be the output of the below-mentioned code

mixed_tuple = 13, 17, 23, 78, 141

mixed_tuple[3] = 90

print(mixed_tuple)

a) (13, 17, 23, 78, 141)
b) (13, 17, 23, 90, 141)
c) (13, 17, 90, 78, 141)
d) Error

7) What will be the output of the below-mentioned code

mixed_list1 = 13, 17, 23, 78, 141

mixed_list1[3] = 90

print(mixed_list1)

a) [13, 17, 23, 78, 141]
b) [13, 17, 23, 90, 141]
c) [13, 17, 90, 78, 141]
d) Error

REFERENCES

[1] https://beginnersbook.com/2018/01/introduction-to-python-programming/

[2] https://realpython.com/installing-python/

[3] https://www.tutorialspoint.com/How-do-we-access-command-line-arguments-in-Python

[4] https://www.programiz.com/python-programming/variables-datatypes

[5] https://realpython.com/python-data-types/

[6] http://localhost:8888/notebooks/Untitled30.ipynb?kernel_name=python3

[7] https://www.edureka.co/blog/data-structures-in-python/

[8] https://www.tutorialspoint.com/python/python_dictionary.htm

B.Sc.(DATA SCIENCE)

SEMESTER-I

PROBLEM SOLVING USING COMPUTERS

UNIT II: PYTHON COMPARISON OPERATORS

STRUCTURE

2.0 OBJECTIVES

- Having knowledge about logical operators
- Students will be able to understand chaining comparison operators.
- Students will be inculcated concepts like break, continue and pass statements deeply and clearly.
- Students can build programs using relational, logical, identity and membership operators.

2.1 INTRODUCTION

This module helps the students in better understanding of logical operators. The different types of logical operators (and, or, not) are elaborated with many programming examples. Truth tables along with flowcharts are also mentioned for clearly understand the logical operators. The next topic is followed with precedence of logical operators and appropriate programs are mentioned to explain this topic briefly. This unit also targets to explain chaining comparison operators. Before deeply understanding this concept, firstly comparison operators, identity operators and membership operators are explained and then these are explained with chaining programming examples also. The difference between equal to operator and is operator is clearly explained followed with not equal to operator and is not operator. For explaining both differences clearly, python programming examples are given. Statements like break, continue and pass are discussed with programming examples. Distinction between pass and comments ae also assessed and mentioned. These whole things are necessary for doing complex problems and projects. This unit is followed with self-checked questions and summary of the whole topic. At last, practice questions are mentioned.

2.2 LOGICAL OPERATORS

It is used when decision making has to be performed on multiple conditions. Condition is the operand and it is assessed with a True of False. Example: have_injection1 and have_injection2

2.2.1 Types of Logical Operators

In Python, three logical operators are used for conditional statements. The operators with its symbols are mentioned in Table 2.1.

Table 2.1: Logical operators with symbols

Operator Symbol	Name	Narrative (Comparison between two conditions)
and	Logical AND	1. Returns True if both operands are True 2. Returns False if one of the operands is False
Or	Logical OR	1. Returns True if one of the operands is True 2. Returns False if both operands are False
Not	Logical NOT	1. Returns True if operand is False 2. Returns False if operand is True

2.2.1.1 Logical AND

Logical AND operator returns True if both operands are True unless it returns False. The truth table is mentioned in table 2.2 and its flow diagram is shown in figure 2.1.

Table 2.2: Truth Table for Two Conditions using Logical AND

Condition1	Condition2	Output (Condition1 or Condition2)
True	True	True
True	False	False
False	True	False
False	False	False

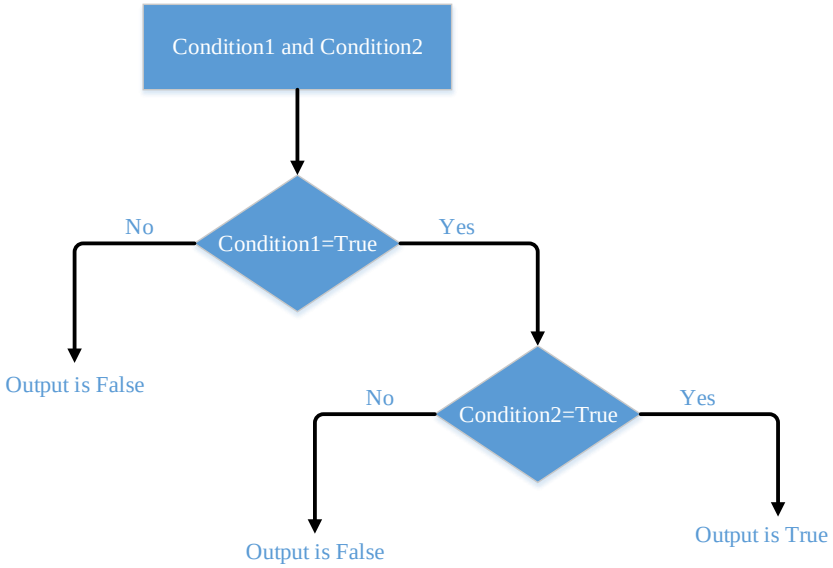


Figure 2.1: Flowchart: Logical AND

Example 1: Use of Logical AND

#variables with initial values

have_injection1=True

have_injection2=True

have_fever=False

have_headache=False

 # Condition with use of LOGICAL AND

have_injection1 and have_injection2

 # Output

True

 # Condition with use of LOGICAL AND

have_injection1 and have_fever

#Output

False

 # Condition with use of LOGICAL AND

have_fever and have_injection2

#Output

False

 # Condition with use of LOGICAL AND

have_fever and have_headache

#Output

False

Example 2: Make a python program in which get the input from user as CGPA of student, if CGPA is between 8 to 10 (including both 8 and 10) then print is message ‘Outstanding’ and if it is between 6 to 8 (including 6 only) then print a message ‘Average’ and if it lies between 4 to 6 (including 4 only) then print a message ‘Hard Work Needed’ and if it lies between 0 to 4 then print a message ‘Fail’.

Solution:

```
CGPA = float(input("Enter CGPA "))  #Input for CGPA from the user
if CGPA >=8 and CGPA <=10:    # Check CGPA between 8 to 10, (including 8 and 10 both)
   print("Outstanding")
if CGPA >=6 and CGPA <8:       # Check CGPA between 6 to 8, (including 6 only)
   print("Good Job")
if CGPA >=4 and CGPA <6:       # Check CGPA between 4 to 6, (including 4 only)
   print("Hard Work Needed")
if CGPA >=0 and CGPA <4:       # Check CGPA between 0 to 4, (including 0 only)
   print("Fail")
if CGPA < 0:                   # Check CGPA should not be less than 0 then print Wrong Input
   print("Wrong Input")
if CGPA > 10:                  # Check CGPA should not be more than 10 then print Wrong
Input
   print("Wrong Input")
```

Test Case 1:
Enter CGPA 8.1
#Output
Outstanding
Test Case 2:
Enter CGPA -1
#Output
Wrong Input
Test case 3:
Enter CGPA 11.3
#Output
Wrong Input

2.2.1.2 Logical OR

Logical OR operator returns True if any one of the operand is True unless it returns False. The truth table is mentioned in table 2.3 and its flow diagram is shown in figure 2.2.

Table 2.3: Truth Table for Two Conditions Using Logical OR

Condition1	Condition2	Output (Condition1 or Condition2)
True	True	True
True	False	True
False	True	True

False	False	False

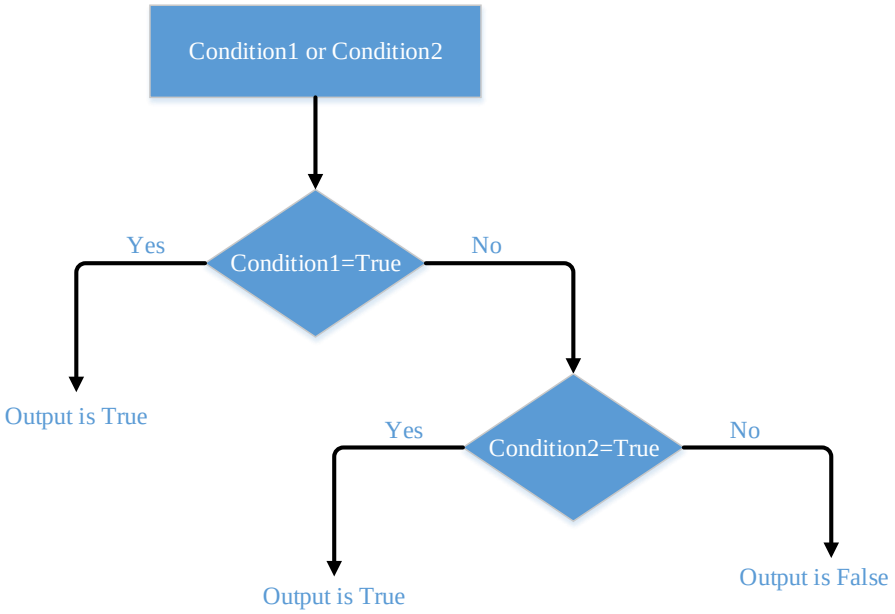


Figure 2.2: Flowchart: Logical OR

Example 3: Use of Logical OR

#variables with initial values

have_injection1=True

have_injection2=True

have_fever=False

have_headache=False

 # Condition with use of LOGICAL OR

have_injection1 or have_injection2

 # Output

True

 # Condition with use of LOGICAL OR

have_injection1 or have_fever

#Output

True

 # Condition with use of LOGICAL OR

have_fever or have_injection2

#Output

True

 # Condition with use of LOGICAL OR

have_fever or have_headache

#Output

False

Example 4:

Make a python program and input three integer numbers from the user if either one of the number is positive then print 'Atleast one of them is positive' else print 'All three numbers are negative'

Solution:

```
number1 = int(input())          #Input first number from the user
number2 = int(input())          #Input second number from the user
number3 = int(input())          #Input third number from the user
if number1>=0 or number2 >= 0 or number3>=0:     # Check if any number is positive
   print("Atleast one of them is positive")
else:
   print("All three numbers are negative")
```

Test Case 1:

100
-300
-900

#Output

Atleast one of them is positive

Test Case 2:

-700
-12300
-18000

#Output

All three numbers are negative

Test Case 3:

0
-12
-23

#Output

Atleast one of them is positive

2.2.1.3 Logical NOT

Logical NOT operator returns True if the operand is False and returns False if the operand is True. The truth table is mentioned in table 2.4 and its flow diagram is shown in figure 2.3.

Table 2.4: Truth Table for Logical NOT

Condition	Output (not(Condition))
True	False
False	True

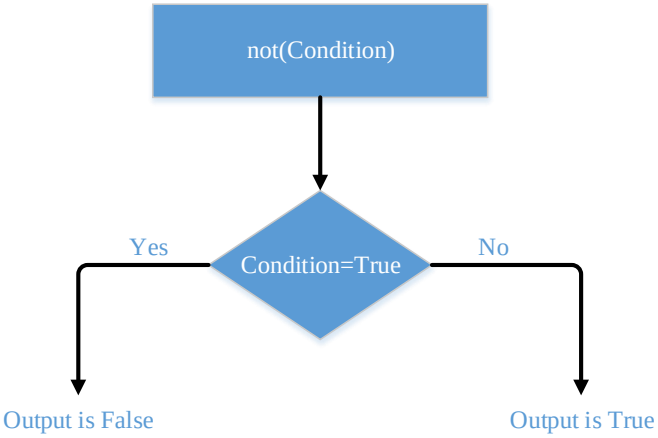


Figure 2.3: Flowchart: Logical NOT

Example 5: Use of Logical NOT

```
#variables with initial values
have_injection1 = True
have_fever = False
# Condition with use of LOGICAL NOT
not(have_injection1)
# Output
False

# Condition with use of LOGICAL NOT
not(have_fever)
# Output
True
```

2.3 PRECEDENCE OF LOGICAL OPERATORS

Table 2.5 shows the precedence of logical operators.

Table 2.5: Precedence of Logical Operators

Operator Name	Symbol	Precedence Level
Logical NOT	Not	1
Logical AND	And	2
Logical OR	Or	3

Example 6:

```
#variables with initial values
have_injection1=True
have_injection2=True
have_fever=False
have_headache=False
#Condition with Multiple Logical Operators
if have_fever == True or have_injection1==True and not(have_headache):
    print("CovidChances")
else:
```

```
    print("NoCovidChances")
```
#Output

CovidChances

Explanation of Example 6:

In above example 6, first ‘not’ operator is executed for not(have_headache) and the value becomes True, then ‘and’ operator is executed for (have_injection1==True and True) and the value becomes True and at last ‘or’ operator is executed for have_fever or True and the value becomes True, the final output is CovidChances.

Example 7:

```
#variables with initial values
have_injection1=True
have_injection2=True
have_fever=False
have_headache=False
#Condition with Multiple Logical Operators
if have_fever == True and have_injection1==True or have_headache==True:
    print("CovidChances")
else:
    print("NoCovidChances")
```
#Output

NoCovidChances

Explanation of Example 7:

In above example 7, first ‘and’ operator is executed for (have_fever == true and have_injection1==True) and the value becomes False, then ‘or’ operator is executed for (False or have_headache==True) and the value becomes False and the final output is NoCovidChances.

2.4 ORDER OF EVALUATION OF LOGICAL OPERATORS

Python evaluates the expression from left to right if there are multiple operators in the expression.

Example 8:

```
#variables with initial values
have_injection1=True
have_injection2=True
have_fever=False
have_headache=False
#Condition with Multiple Logical Operators
if have_headache == True or have_fever==True and have_injection1==True and
have_injection2==True:
    print("CovidChances")
else:
    print("NoCovidChances")
#Output
```

NoCovidChances

Explanation: In example 8, firstly ‘and’ operator is implemented as it has higher precedence then all other operators. There are two ‘and’ operators in above expression, so order of evaluation of logical operators is from left to right, so firstly (have_fever==True and have_injection1) is evaluated and its output is False and then this output is evaluated as (False and have_injection2==True) and the output is again False and then last operation is performed i.e. (have_headache == True or False) and for the final output else part will work and it is NoCovidChances.

Example 9:

```
#variables with initial values
have_injection1=True
have_injection2=True
have_fever=False
have_headache=False
#Condition with Multiple Logical Operators
if have_headache == True or not(have_fever==True) and have_injection1==True and
have_injection2==True:
   print("CovidChances")
else:
   print("NoCovidChances")
```

#Output

CovidChances

Explanation: In example 9, firstly ‘not’ operator is executed as it has higher precedence then all other operators in the expression and the output of not(have_fever==True) becomes True and then ‘and’ operator is implemented as (True and have_injection1==True) and the output is true and then this output is used as (True and have_injection2==True) and the output is True and then it is used as (have_headache==True or True) and the final output is True so else part is executed and it results in CovidChances.

Example 10: Make a python program to find out a year is leap year or not using Logical operators.

Basic Understanding: First of understand the basics of leap year, a year always divided by 4 is not a leap year. Surprised, yes but it is true. A year when divided by 4 it should not be divided by 100 then it is a leap year. If it is divided by 100 then it should be divided by 400 for being a leap year unless it is not a leap year

Solution:

```
# Python Program to check whether a year is leap year or not using Logical Operators
normal_year = int(input())
if normal_year%400 == 0 or normal_year%4 == 0 and normal_year%100 != 0:
   print("Leap Year")
else:
   print("Not a Leap Year")
```

Hints: firstly ‘and’ operators is executed and then ‘or’ operator is executed
Test case 1:
1008
#Output
Leap Year
Test case 2:
2000
#Output
Leap Year
Test case 3:
1900
#Output
Not a Leap Year

2.5 COMPARISON OPERATORS

These operators compare the operand values and on the basis of these results, it returns True or False. The list of comparison operators used in python are:

a) < (less than operator)

It checks condition whether left hand side value is less than right hand side value. After comparing these values, it returns True or False. If condition is True then it returns True else False.

Example:

i) 17 < 23
Output: True

ii) 28 < 23
Output: False

b) > (greater than operator)

It checks condition whether left hand side value is greater than right hand side value. After comparing these values, it returns True or False. If condition is True then it returns True else False.

Example:

i) 17 > 23
Output: False

ii) 28 > 23
Output: True

c) <= (less than and equal to operator)

It checks condition whether left hand side value is less than and equal to right hand side value. After comparing these values, it returns True or False. If condition is True then it returns True else False.

Example:

i) 17 <= 23
Output: True

ii) 28 <= 23
Output: False
iii) 17 <= 17
Output: True

d) >= (greater than and equal to operator)

It checks condition whether left hand side value is greater than and equal to right hand side value. After comparing these values, it returns True or False. If condition is True then it returns True else False.

Example:

i) 17 >= 23
Output: True
ii) 28 >= 23
Output: False
iii) 17 >= 17
Output: True

e) == (equal to operator)

It checks condition whether left hand side value is equal to right hand side value. After comparing these values, it returns True or False. If condition is True then it returns True else False.

Example:

i) 17 == 23
Output: False
ii) 180== 180
Output: True

f) != (Not equal to operator)

It checks condition whether left hand side value is not equal to right hand side value. After comparing these values, it returns True or False. If condition is True then it returns True else False.

Example:

i) 17 != 23
Output: True
ii) 180!= 180
Output: False

g) is (identity operator)

is operator is used to check whether both operands point to the same object or not. This is different from ==. In equal to operator, it is checked whether both operand have same value but here is operator checks whether both operands point to the same object or not [1].

Example :

```
#List1 is initialised with four values
mixed_list1 = [34,45,89,123]
```

```
# List2 is initialised with four values
mixed_list2 = [34,45,89,123]
# List 3 is assigned values of List1 (Note: Both will have same address)
mixed_list3 = mixed_list1
#Printed the elements of List1
print(mixed_list1)
#Printed the elements of List2
print(mixed_list2)
#Printed the elements of List3
print(mixed_list3)
# Here address of List1 is printed
print(id(mixed_list1))
# Here address of List1 is printed
print(id(mixed_list2))
# Here address of List1 is printed
print(id(mixed_list3))
# Here, values are checked of both operands
if mixed_list1 == mixed_list2:
   print("Both mixed-list1 and mixed_list2 have equal value")
else:
   print("Both mixed-list1 and mixed_list2 have not equal value")

if mixed_list1 == mixed_list3:
   print("Both mixed-list1 and mixed_list3 have equal value")
else:
   print("Both mixed-list1 and mixed_list3 have not equal value")

if mixed_list2 == mixed_list3:
   print("Both mixed-list2 and mixed_list3 have equal value")
else:
   print("Both mixed-list2 and mixed_list3 have not equal value")

# Here checking of operands point to the same object or not
if mixed_list1 is mixed_list2:
   print("Both mixed-list1 and mixed_list2 point to the same object")
else:
   print("Both mixed-list1 and mixed_list2 do not point to the same object ")

if mixed_list1 is mixed_list3:
   print("Both mixed-list1 and mixed_list3 point to the same object")
else:
   print("Both mixed-list1 and mixed_list3 do not point to the same object ")

if mixed_list2 is mixed_list3:
```

```
        print("Both mixed-list2 and mixed_list3 point to the same object")
    else:
        print("Both mixed-list2 and mixed_list3 do not point to the same object ")
```

#Output

```
[34, 45, 89, 123]
[34, 45, 89, 123]
[34, 45, 89, 123]
89447112
89448072
89447112
Both mixed-list1 and mixed_list2 have equal value
Both mixed-list1 and mixed_list3 have equal value
Both mixed-list2 and mixed_list3 have equal value
Both mixed-list1 and mixed_list2 do not point to the same object
Both mixed-list1 and mixed_list3 point to the same object
Both mixed-list2 and mixed_list3 do not point to the same object
```

h) is not (identity operator)

is not operator is used to check whether both operands do not point to the same object . This is different from !=. In not equal to operator, it is checked whether both operand have not same value but here is operator checks whether both operands do not point to the same object.

Example:

```
#List1 is initialised with four values
mixed_list1 = [34,45,89,123]

# List2 is initialised with three values
mixed_list2 = [34,89,123]

# List 3 is assigned values of List1 (Note: Both will have same address)
mixed_list3 = mixed_list1
#Printed the elements of List1
print(mixed_list1)
#Printed the elements of List2
print(mixed_list2)
#Printed the elements of List3
print(mixed_list3)
# Here address of List1 is printed
print(id(mixed_list1))
# Here address of List1 is printed
print(id(mixed_list2))
# Here address of List1 is printed
print(id(mixed_list3))
# Here, values are checked of both operands
```

```
if mixed_list1 != mixed_list2:
   print("Both mixed-list1 and mixed_list2 have not equal value")
else:
   print("Both mixed-list1 and mixed_list2 have equal value")

if mixed_list1 != mixed_list3:
   print("Both mixed-list1 and mixed_list3 have not equal value")
else:
   print("Both mixed-list1 and mixed_list3 have equal value")

if mixed_list2 != mixed_list3:
   print("Both mixed-list2 and mixed_list3 have not equal value")
else:
   print("Both mixed-list2 and mixed_list3 have equal value")

# Here checking of operands point to the same object or not
if mixed_list1 is not mixed_list2:
   print("Both mixed-list1 and mixed_list2 do not point to the same object")
else:
   print("Both mixed-list1 and mixed_list2 point to the same object ")

if mixed_list1 is not mixed_list3:
   print("Both mixed-list1 and mixed_list3 do not point to the same object")
else:
   print("Both mixed-list1 and mixed_list3 point to the same object ")

if mixed_list2 is not mixed_list3:
   print("Both mixed-list2 and mixed_list3 do not point to the same object")
else:
print("Both mixed-list2 and mixed_list3 point to the same object ")
```

#Output

```
[34, 45, 89, 123]
[34, 89, 123]
[34, 45, 89, 123]
89101000
89101512
89101000
Both mixed-list1 and mixed_list2 have not equal value
Both mixed-list1 and mixed_list3 have equal value
Both mixed-list2 and mixed_list3 have not equal value
Both mixed-list1 and mixed_list2 do not point to the same object
Both mixed-list1 and mixed_list3 point to the same object
Both mixed-list2 and mixed_list3 do not point to the same object
```

i) in (membership operator)

This operator checks whether the given value is present in the sequence or not. It evaluates to True if it is present else False.

Examples:

i) var_int1=459

```
mixed_list1 = [23,459,478,65]
if var_int1 in mixed_list1:
   print("Variable value is present in the given list")
else:
   print("Variable value is absent in the given list")
```

Output: Variable value is present in the given list

ii) var_int1=455

```
mixed_list1 = [23,459,478,65]
if var_int1 in mixed_list1:
   print("Variable value is present in the given list")
else:
   print("Variable value is absent in the given list")
```

Output: Variable value is absent in the given list

j) not in(membership operator)

This operator checks whether the given value is not present in the sequence or not. It evaluates to True if it is not present else False.

Examples:

i) var_int1=459

```
mixed_list1 = [23,459,478,65]
if var_int1 not in mixed_list1:
   print("Variable value is not present in the given list")
else:
   print("Variable value is present in the given list")
```

Output: Variable value is present in the given list

ii) var_int1=455

```
mixed_list1 = [23,459,478,65]
if var_int1 not in mixed_list1:
   print("Variable value is not present in the given list")
else:
   print("Variable value is present in the given list")
```

Output: Variable value is not present in the given list

2.5.1 Chaining of Comparison Operators

All the above-mentioned comparison operators gives the output in the form of True or False. While doing chaining using comparison operators, it is done arbitrarily [2].
Chaining can be better understood by the following example:
int_var1 > int_var2
int_var2 > int_var3

For above, chaining can be done like
int_var1 > int_var2 > int_var3

Syntax:
Operand1 operator1 Operand2 operator2 Operand3 operator3 Operand4 ….

Always process chaining from left to right as precedence of comparison operators is same.

Example 11: Program 1 of Chaining to understand the concept

```
int_var1 = 189
int_var2 = 174
int_var3 = 67

# Firstly check int_var1 >int_var2
print(int_var1 > int_var2)

# Now, Check int_var2 > int_var3
print(int_var2 > int_var3)

# Do chaining and check the output
print(int_var1 > int_var2 > int_var3)

# Above chaining is equivalent to
if (int_var1 > int_var2 and int_var2 > int_var3):
   print(True)
else:
   print(False)
```

#Output

```
True
True
True
True
```

Example 12: Program 2 for deeper understanding of chaining operator

```
int_var1 = 189
int_var2 = 474
int_var3 = 267

# Firstly check int_var1 >int_var2
print(int_var1 < int_var2)

# Now, Check int_var2 > int_var3
print(int_var2 > int_var3)

# Do chaining and check the output
print(int_var1 < int_var2 > int_var3)  # Perfectly Legal

# Above chaining is equivalent to
if (int_var1 < int_var2 and int_var2 > int_var3):
   print(True)
else:
   print(False)
```

#Output

```
True
True
True
True
```

Example 13: Program 3 of Chaining

```
int_var4=34
print(27<int_var4<35)
print(37<int_var4<45)

print(27<int_var4>14)
print(27<int_var4>67)

print(34==int_var4>17)

print(34==int_var4>67)
```

#Output

True
False
True
False
True
False

Example 14: Chaining operators complex programs

```
int_var1=28
int_var2=28
int_var3=37

# Here checking int_var1 and int_var2 has same address and int_var3 is having different address
print(int_var1 is int_var2 is not int_var3)

# Addresses can be checked
print(id(int_var1))
print(id(int_var2))
print(id(int_var3))

print(int_var1 is not int_var2 is int_var3)

int_var4=12
print(int_var1 is int_var2 is not int_var3 > int_var4)
# Above left to right will work
int_var5=9
print(int_var1 is int_var2 is not int_var3 > int_var4 >int_var5)

int_var6=89
print(int_var1 is int_var2 is not int_var3 > int_var4 >int_var5< int_var6)

int_var7=76
print(int_var1 is int_var2 is not int_var3 > int_var4 >int_var5< int_var6 >int_var7)
```

#Output

```
True
8790643746032
8790643746032
8790643746320
```

False
True
True
True
True

Example 15: Another program for chaining

```
# Variables with initial values
int_var1=28
int_var2=28
int_var3=37

print(int_var1==int_var2)

int_var4=0
print(int_var4 < int_var1==int_var2)

int_var5=0
print(int_var4 < int_var1==int_var2 <int_var5)

int_var6=45
print(int_var4 < int_var1==int_var2 >int_var6)
```

#Output

True
True
False
False

Example 16: Chaining with usage of ==,!=,is, is not with lists

```
#List1 is initialised with four values
mixed_list1 = [34,45,89,123]

# List2 is initialised with three values
mixed_list2 = [34,89,123]

# List 3 is assigned values of List1 (Note: Both will have same address)
mixed_list3 = mixed_list1
```

```
#Printed the elements of List1
print(mixed_list1)
#Printed the elements of List2
print(mixed_list2)
#Printed the elements of List3
print(mixed_list3)

# Here address of List1 is printed
print(id(mixed_list1))
# Here address of List1 is printed
print(id(mixed_list2))
# Here address of List1 is printed
print(id(mixed_list3))

# First if-else
if mixed_list1 != mixed_list2 is not mixed_list3:
    print("The first if condition is True")
else:
    print("The first else part is worked")

#Second if-else
if mixed_list1 != mixed_list3 is not mixed_list2:
    print("The second if condition is True")
else:
    print("The second else part is worked")

# Third if-else
if mixed_list2 != mixed_list3 is not mixed_list1:
    print("The third if condition is True")
else:
    print("The third else part is worked")

# Fourth if-else
if mixed_list1 != mixed_list2 is mixed_list3:
    print("The fourth if condition is True")
else:
    print("The fourth else part is worked")

# Fifth if-else
```

```
if mixed_list1 != mixed_list3 is mixed_list2:
    print("The fifth if condition is True")
else:
    print("The fifth else part is worked")

# sixth if-else
if mixed_list2 != mixed_list3 is mixed_list1:
    print("The sixth if condition is True")
else:
    print("The sixth else part is worked")

# seventh if-else
if mixed_list1 is not mixed_list2 == mixed_list3:
    print("The seventh if condition is True")
else:
    print("The seventh else part is worked")

# eight if-else
if mixed_list1 is not mixed_list3 == mixed_lis2:
    print("The eight if condition is True")
else:
    print("The eight else part is worked")

# ninth if-else
if mixed_list2 is not mixed_list3==mixed_list1:
    print("The ninth if condition is True")
else:
    print("The ninth else part is worked")

# tenth if-else
if mixed_list1 is mixed_list2 == mixed_list3:
    print("The tenth if condition is True")
else:
    print("The tenth else part is worked")

# eleventh if-else
if mixed_list1 is mixed_list3 == mixed_list2:
    print("The eleventh if condition is True")
else:
    print("The eleventh else part is worked")
```

```
# twelveth   if-else
if mixed_list2 is mixed_list3==mixed_list1:
   print("The twelfth if condition is True")
else:
   print("The twelfth else part is worked")
```
#Output
```
[34, 45, 89, 123]
[34, 89, 123]
[34, 45, 89, 123]
90139784
90139720
90139784
The first if condition is True
The second else part is worked
The third else part is worked
The fourth else part is worked
The fifth else part is worked
The sixth if condition is True
The seventh else part is worked
The eight else part is worked
The ninth if condition is True
The tenth else part is worked
The eleventh else part is worked
The twelfth else part is worked
```

2.6 USAGE OF BREAK, CONTINUE AND PASS STATEMENTS

2.6.1 Break Statement

The keyword break is used to interrupt the running loop. Basically, this is intentional. When break statement is encountered, the flow goes to immediate step after the loop [3].

Syntax:

```
break
```

Example 17: Usage of break with while loop

```
int_var1 = 5
while int_var1<15:
  if int_var1 == 9:
    break
  print(int_var1)
  int_var1 = int_var1+1
print("Outside loop now")
```

```
#Output
5
6
7
8
Outside loop now
```

Example 18: Usage of break with for loop

```
for int_var1 in range(5,15):
  if int_var1!=9:
    print(int_var1)
  else:
    break
print("Outside loop now")

#Output
5
6
7
8
Outside loop now
```

Example 19: Search an element in the list

```
mixed_list1 = [12, 3, 24, 45, 89, 108, 23]
item_find = int(input("Enter the item you want to find out from the list"))
num_elements = len(mixed_list1)
flag=0
for i in range(0, num_elements):
        if (mixed_list1[i] == item_find):
            flag=1
            break
if flag==0:
    print("Element is not present in array")
else:
    print("Element is present in array")
```

Test Case 1:

```
#Input
Enter the item you want to find out from the list88
#Output
```

Element is not present in array

Test Case 2:
#Input
Enter the item you want to find out from the list12
#Output
Element is present in array
Test Case 3:
#Input
Enter the item you want to find out from the list108
#Output
Element is present in array

2.6.2 Continue Statement

The keyword continue is used to end the current iteration of the loop and iterate the next iteration of the loop [4].

Syntax:
continue

Example 20: Usage of continue with while loop

```
int_var1 = 5
while int_var1<15:
   int_var1 = int_var1+1;
   if int_var1 == 9:
      continue
   print(int_var1)

print("Outside loop now")
```

#Output

```
6
7
8
10
11
12
13
14
15
Outside loop now
```

Example 21: Usage of continue with for loop

```
for int_var1 in range(5,15):
  if int_var1!=9:
    continue
  else:
    print(int_var1)
print("Outside loop now")
```

#Output

```
9
Outside loop now
```

2.6.3 Pass Statement

pass statement is simply does not do anything. It can be used in loops, functions, classes etc. It is different from comments. As interpreter ignore comments whereas interpreter does not ignore pass statement.

Syntax

```
pass
```

Example 22: Program to show Without pass statement, an error will occur

```
for int_var1 in [12,23,89,56,235]:
```

#Output

SyntaxError: unexpected EOF while parsing

Example 23: Program with relation to example 22 but with using pass

```
for int_var1 in [12,23,89,56,235]:
   pass
```

After executing the above statements, No error will be generated

Example 24: pass with function and loop

```
str_var1 = "JagatOpenUniversity"

for new_var1 in str_var1:
   if new_var1 =='O' or new_var1=='U':
      print("Pass statement is exceuted when value is O and U")
      pass
```

```
    print(new_var1)


def fun_pass():
    pass

print("Now fun_pass() is called")
fun_pass()
print("After fun_pass() calling, nothing is printed as pass statement is in fun_pass()")
```

#Output

```
J
a
g
a
t
Pass statement is exceuted when value is O and U
O
p
e
n
Pass statement is exceuted when value is O and U
U
n
i
v
e
r
s
i
t
y
Now fun_pass() is called
After fun_pass() calling, nothing is printed as pass statement is in fun_pass()
```

2.7 SELF-CHECKED QUESTIONS

A) What will be the output of the following code:

```
for int_var1 in range(3):
    for int_var2 in range(3):
        if int_var2==2:
```

```
        break
    print(int_var1,end=" ")
```

a) 0 0 1 1 2 2 b) 0 1 0 1 0 1 c) 0 1 2 0 1 2 d) 0 1 0 2 0 3

B) Select the best option for the following code:

```
for int_var1 in range(3):
  for int_var2 in range(3):
    if int_var2==2:
      break
    print(int_var2,end=" ")
```

a) 0 0 1 1 2 2 b) 0 1 0 1 0 1 c) 0 1 2 0 1 2 d) 0 1 0 2 0 3

C) Write the output of the following code

```
for int_var1 in range(6):
  for int_var2 in range(6):
    if int_var2==2:
      continue
    print(int_var2,end=" ")
```

D) Write the output of the following code

```
for int_var1 in range(6):
  for int_var2 in range(6):
    if int_var2==2:
      continue
    print(int_var1,end=" ")
```

E) Mention the output of the code

```
for int_var1 in range(3):
  for int_var2 in range(3):
    if int_var2==2:
      pass
    print(int_var2,end=" ")
```

2.8 SUMMARY

After reading this unit, students will be able to differentiate between all logical operators and they will know where they have to use which operator. When students are working on break and continue statements. There is a lot of confusion, how these statements will work. Appropriate programming examples are given to provide deeper understanding of these concepts. The unit gives programming examples related with logical operators, chaining comparison operators, pass statement. Overall, this unit helps in building complex programs. Students can differentiate pass

statement and comments. This unit targets to build logical concepts so that they can do MCQ, predict the output of the code along with programs construction.

2.9 PRACTICE QUESTIONS

A) Which of the following will not be printed when the Python3 cod3 is run

```
for str_var1 in 'JagatOpenUniversity':
    if str_var1 == 'O' or str_var1=='U':
        continue
    print('The string alphabet is :' + str_var1)
```

a) The string alphabet is :a
b) The string alphabet is :U
c) The string alphabet is :e
d) The string alphabet is :n

B) How many times '?' will be printed when the following code is executed on Python3 platform.

```
for int_var1 in [5, 8, 10]:
    for int_var2 in [5,6,7,8,9,10]:
        if int_var1!=int_var2:
            continue
        print('?')
```

a) 1
b) 3
c) 4
d) 6

C) How many times '?' will be printed when the following code is executed on Python3 platform.

```
for int_var1 in [5, 8, 10]:
    for int_var2 in [5,6,7,8,9,10]:
        if int_var1!=int_var2:
            break
        print('?')
```

a) 1
b) 3
c) 4
d) 6

D) Mention the output of the following code

```
int_var1=12
int_var2 = 37
int_var3 =23

print(int_var2> int_var1==int_var3)
```

E) Mention the output

```
int_var1=12
int_var2 = 37
int_var3 =12

print(int_var2> int_var1==int_var3 < int_var2)
```

F) Predict the output

```
int_var1 = 1
while True:
    if int_var1%9 == 0:
        break
    print("The output when this statement is executed",int_var1)
    int_var1=int_var1+4
print("At the end, you are out of the loop and this line is printed")
```

G) Predict the output

```
int_var1 = 1
while int_var1<19:
    if int_var1%9 == 0:
        int_var1=int_var1+1
        continue
    print("The output when this statement is executed",int_var1)
    int_var1=int_var1+4
print("At the end, you are out of the loop and this line is printed")
```

H) Choose the correct option after running the following code:

```
int_var1 = 1
while int_var1<19:
    if int_var1%2 == 0:
        int_var1=int_var1+1
        continue
    if int_var1%9 == 0:
```

```
        int_var1=int_var1+1
        pass
    print(int_var1,end=" ")
    int_var1=int_var1+4
```

a) 1 5 10 15 b) 1 1 3 3 5 5 7 7 9 9 11 11 13 13 15 15 17 17
c) 1 5 9 13 17 d) 1 3 5 7 9 11 13 15 17

I) Select the best option after running the following code

```
int_var1 = 1
while int_var1<19:
    if int_var1%2 == 0:
        int_var1=int_var1+1
        continue
    if int_var1%11 == 0:
        int_var1=int_var1+1
        break
    print(int_var1,end=" ")
    int_var1=int_var1+4
```

a) 1 5 9 13 17
b) 1 3 5 7 9 11 13 15 17
c) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
d) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

J) What will be the correct output of the following code

```
bool_var1 = False
while True:
    print(True)
    break
```

K) What will be the correct output of the following code

```
bool_var1 = True
while True:
    print(True)
    break
```

L) Differentiate pass and comments statement in python programming
M) Construct a program to find out whether 179 is a prime number or not.
N) With the help of logical operators, find out the largest of four numbers.
O) List out the differences between break and continue.

P) Mention the precedence level of logical operators and explain with a suitable example which shows precedence level usage.

Q) List the usage of ‘is operator’ and ‘is not operator’ with suitable example.

REFERENCES

[1] https://www.geeksforgeeks.org/difference-operator-python/

[2] https://www.tutorialspoint.com/chaining-comparison-operators-in-python

[3] https://www.programiz.com/python-programming/break-continue

[4] https://www.w3schools.com/python/ref_keyword_continue.asp

B.Sc.(DATA SCIENCE)

SEMESTER-I

PROBLEM SOLVING USING COMPUTERS

UNIT III: PROGRAM FLOW CONTROL

STRUCTURE

3.0 OBJECTIVES

- Learn in detail about conditional statements followed by syntax and coding programs.
- Shorthand notations learning with syntax and examples
- Familiarize with different types of loops.
- Detailed learning about nested loops along with coding examples.

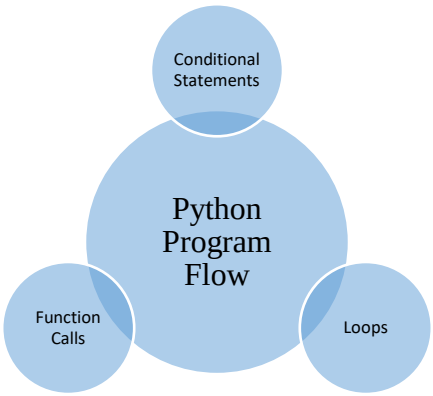
3.1 INTRODUCTION

This module targets to give deep insights into conditional statements like if, if-else, if-elif-else, nested if-else. Syntax along with examples are also mentioned. Program examples are like leap year, even-odd number, whole number or not, greatest among three numbers and four numbers, etc. Conditional statements help in knowing the execution order of code statements. The statements like if, if-else, etc. help in making the decision and repeat certain blocks to do a particular task. For repeating the tasks, loops are used extensively. For and while loops are used to iteratively repeat tasks. Syntax and examples related to loops are mentioned in detail. How else part works with for and while loop is explained. Followed by loops working on lists, iterable objects, etc. This module helps the users in getting knowledge about different conditional statements and loops. Functions calls will be explained in the next module.

3.2 FLOW CONTROL

Any program written in any programming language has a certain flow and the flow tells the order in which the programming language code flows. Generally, the programming language code control flow is controlled by the following:

1. Conditional Statements
2. Loops
3. Function Calls



3.1 Python Control Flow

3.2.1 Conditional Statements

Conditional statements are also known as conditional constructs or conditional expressions. They perform different actions depending upon the evaluation of the conditions either True or False. The conditions use different operators like arithmetic, relational, comparison, etc. The conditional statements consist of the following types:

1. Simple if statement
2. if-else statement
3. if elif else statement or Chained Conditional Statements
4. Nested Conditions

3.2.1.1 Simple if Statement

Simple if statement is one of the commonly used conditional statements in mostly all programming languages for decision making. The keyword used for this conditional statement is "*if*". The "*if"* keyword is followed with a condition and this condition is 'True' then a certain indented block of code is executed that is inside "if" block otherwise not. The syntax is mentioned below:

Syntax:

If (condition):

block of statement(s)

Rest of the statements

In the above syntax, *if condition is True then both block of statement(s) and rest of the statements will be executed and if condition is False then only rest of the statements will be executed.*

Example 1: # Check a number is even

```
number1=16                  # Variable with initialization
if(number1%2==0):           # if condition
   print("Even Number")
print("This statement of the code is always printed")    #Rest of the statements
#Output
Even Number
This statement of the code is always printed
```

Example 2: # Check a number is even

```
number1=15                  # Variable with initialization
if(number1%2==0):           # if condition
   print("Even Number")
print("This statement of the code is always printed")    #Rest of the statements
#Output
This statement of the code is always printed
```

Example 3: # Check chances of COVID

```
have_fever = True               # Variable with initialization
if(have_fever):                 # if condition
   print("Chances of COVID")
print("This statement of the code is always printed")    #Rest of the statements
#Output
Chances of COVID
```

This statement of the code is always printed

Example 4: Check Chances of COVID Variant

```
have_fever = False                  # Variable with initialization
if(have_fever):                     # if condition
   print("Chances of COVID")
print("This statement of the code is always printed")    #Rest of the statements
#Output
This statement of the code is always printed
```

3.2.1.2 if-else Statement

This conditional statement uses the keyword *if-else. if* condition is True then block of if statement(s) is executed and *if* condition is False then else part i.e. a block of else statement(s) is executed. The syntax is mentioned below:

Syntax:

```
if (condition():
   block of if statement(s)
else:
   block of else statement(s)
```

Example 5: # Check Immunization Improves for COVID or Not

```
have_injection1 = True
have_injection2 = True
if(have_injection1 == True and have_injection2==True):
   print("Immunization improves for COVID")
else:
   print("Higher chances of prone to COVID")
#Output
Immunization improves for COVID
```

Example 6: Check Number is even or odd

```
number1=15
if(number1%2==0):
   print("Number is Even")
else:
   print("Number is Odd")
#Output
Number is Odd
```

Example 7: Check whether a number is a whole number or not

```
x = 0.87
if (x - int(x) == 0):
```

```
   print("Whole Number")
else:
   print("Has decimals")
#Output
Has decimals
```

Example 8: Find the greater element between two integer elements (Assumption is that both elements cannot be equal)

```
number1=int(input())
number2=int(input())
if number1>number2:
   print("Number1 is greater")
else:
   print("Number2 is greater")
```

Test case 1:

#Input Elements
122
213
#Output
Number2 is greater

Test case 2:

#Input Elements
422
113
#Output
Number1 is greater

3.2.1.3 if elif else Statement Or Chained Conditional Statements

This statement is used to control multiple conditions in the program. The *if* condition is False then *elif* is used to control multiple conditions.

Syntax:

```
if(conidtion1):
   block of statement(s) of if
elif(condition2):
   block of statement(s) of elif
elif(condition3):
   block of statement(s) of elif
.................
.................
.................
else:
   block of statement(s) of else
```

#NOTE: *else* part is optional, the program will work if you do not use *else* part in the code too.

Example 9: # Compare two integer numbers

```
number1=int(input())
number2=int(input())
```

```
if number1>number2:
   print("Number1 is greater")
elif number1<number2:
   print("Number 2 is greater")
else:
   print("Both numbers are equal")
```

Test case 1:

```
#Inputs
122
213
#Output
Number2 is greater
```

Test case 2:

```
#Inputs
422
113
#Output
Number1 is greater
```

Test case 3:

```
#Inputs
333
333
#Output
Both numbers are equal
```

Example 10: Check whether a positive number is two digit, three digit, four digit or greater that four digit number and print an appropriate message.

```
user_number=int(input())#input from the user
if user_number>=0 and user_number <=9:
   print("Single Digit Number")
elif user_number >=10 and user_number <=99:
   print("Two Digit Number")
elif user_number >=100 and user_number <=999:
   print("Three Digit Number")
elif user_number >=1000 and user_number <=9999:
   print("Four Digit Number")
else:
   print("Number is greater than Four Digit Number")
```

Test Case 1:

```
#Input
10
```

#Output
Two Digit Number
Test Case 2:
#Input
9989
#Output
Four Digit Number
Test Case 3:
#Input
6748309
#Output
Number is greater than Four Digit Number

Example 11: Check whether a year is a leap year or not
NOTES for Basic Understanding: Leap year is a year if it is divided by 400. If it is not divided by 400 then check is it divided by 100, if it is divided by 100 then it is not a leap year. If a number is neither divisible by 100 nor 400 then only check whether a given number is divided by 4, if it is divided by 4 then it is a leap year unless it is not a leap year.

```
normal_year = int(input())
if (normal_year%400 == 0):
        print("Leap Year")
elif (normal_year%100 == 0):
        print("Not a Leap Year")
elif (normal_year%4 == 0):
        print("Leap Year")
else:
        print("Not a Leap Year")
```

Test Case 1:
#Input
2000
#Output
Leap Year
Test Case 2:
#Input
1900
#Output
Not a Leap Year
Test Case 3:
#Input
2020
#Output
Leap Year
Test Case 4:
#Input
2021
#Output
Not a Leap Year

3.2.1.4 Nested Conditions

Nested conditions consist of nested if as well as nested if-else. In the nested if, we have multiple if in outer if block. In the nested if-else block, we have if-else statements inside other *if* or *else* or both blocks. The important thing is here to take care of indentation and it is the way to know the level of nesting.

Syntax of nested if:

```
if condition1():
   if condition2():
      if condition3():
         …….
         …….
         block3 of statement(s)
      block 2 of statement(s)
   block 1 of statement(s)
else:
   block of statement(s)
```

Syntax of nested if-else:

```
if condition1():
   if condition2():          #optional if-else
      block of if statement(s)
   else:
      block of else statement(s)
else:
   if condition3():          #optional if-else
      block of if statement(s)
   else:
      block of else statement(s)
```

Example 12: Find the greatest of three integer numbers (Assume that all three numbers are distinct)

```
number1=int(input())
number2=int(input())
number3=int(input())
if number1>number2:
   if number1>number3:
      print("number1 is greatest")
   else:
      print("number3 is greatest")
else:
   if number2>number3:
```

```
        print("number2 is greatest")
    else:
        print("number3 is greatest")
```
Test case1:
#Inputs
23
36
47
#Output
number3 is greatest
Test case2:
#Inputs
67
50
56
#Output
number1 is greatest
Test case3:
#Inputs
145
567
444
#Output
number2 is greatest

Example 13: Check whether a year is a leap year or not

```
normal_year = int(input())
if normal_year%4==0:
    if normal_year%100==0:
        if normal_year%400==0:
            print("Leap Year")
        else:
            print("Not a Leap Year")
    else:
        print("Leap Year")
else:
    print("Not a Leap Year")
```

Test Case 1:
#Input
2000
#Output
Leap Year
Test Case 2:

#Input
1900
#Output
Not a Leap Year
Test Case 3:
#Input
2020
#Output
Leap Year
Test Case 4:
#Input
2021
#Output
Not a Leap Year

Example 14: Find the greatest of four integer numbers (Assume that all four numbers are distinct)

```
number1=int(input())
number2=int(input())
number3=int(input())
number4=int(input())
if number1>number2:
   if number1>number3:
      if number1>number4:
         print("number1 is greatest")
      else:
         print("numbe4 is greatest")
   else:
      if number3> number4:
         print("number3 is greatest")
      else:
         print("number4 is greatest")
else:
   if number2>number3:
      if number2>number4:
         print("number2 is greatest")
      else:
         print("numbe4 is greatest")
   else:
      if number3> number4:
         print("number3 is greatest")
      else:
         print("number4 is greatest")
```

Test Case1:
$inputs
11
21
25
45
#output
number4 is greatest
Test Case2:
$inputs

23
45
34
12
#output
number2 is greatest
Test Case3:
$inputs
104
210
250
145
#output
number3 is greatest
Test Case4:
$inputs
678
246
298
484
#output
number1 is greatest

3.2.2 Shorthand Notations:

3.2.2.1 Shorthand Notation for if

Syntax:

if (condition): if-statement

Example 15: Check a number is Even Number

```
number1=16
if number1%2==0: print("Even Number")
```

#Output

Even Number

3.3.2.2 Shorthand Notation for if else

Syntax:

if-statement(s) if (condition) else else-statement(s)

Example 16: Check whether a number is even or odd

```
number1=int(input())
print("Number is Even") if number1%2==0 else print("Number is Odd")
```

Test Case 1

#input

15

#output

Number is Odd

Test Case 2

#input

28

#Output

Number is Even

Example 17: Check whether a number is a whole number or not

```
x = 0.87
```

```
print("Whole Number") if(x - int(x) == 0) else print("Has decimals")

#Output
Has decimals
```

Example 18: Find greater of two elements

```
number1=int(input())
number2=int(input())
print("number1 is greater") if(number1>number2) else print("number2 is greater")
```

Test Case1:

```
#Inputs
25
67
#Output
number2 is greater
```

Test Case2:

```
#Inputs
475
143
#Output
number1 is greater
```

3.2.2.3 Shorthand Notation for if elif else

Example 19: Compare two integer numbers

```
number1=int(input())
number2=int(input())
print("Number1 is greater") if number1>number2 else print("Both numbers are equal") if
number1==number2 else print("Number2 is greater")
```

Test case 1:

```
#Inputs
122
213
#Output
Number2 is greater
```

Test case 2:

```
#Inputs
422
113
#Output
Number1 is greater
```

Test case 3:

```
#Inputs
```

333
333
#Output
Both numbers are equal

3.3 LOOPS

Loops execute a specific block of code that contains many statements and this block is executing repetitively. Two types of loops are there in python:

a) while loop
b) for loop

In loops, three things to be taken care and these are mentioned below:

a) Initialization
b) Condition
c) Increment/Decrement

In initialization, variables are initialized to certain values. In the condition part, conditions are specified that tells how many times the loop will be executed repetitively. The third part is increment or decrement of the values of variables that are used in the condition part so that loop will be stopped after performing a particular task.

3.3.1 While Loop

This loop iterate over a block of code repetitively or repeatedly until the condition is satisfied or True and when it becomes dissatisfied or False, the program jumps after the immediate line of the loop. The important thing is how to group many statements in a block. This is done by indentation.

Syntax:

```
initialization
while(condition):
        statement 1
        statement 2  #optional
        …….
        …….
        …….
        statement n #optional
        increment/decrement  # Order of statements and increment/decrement is not specific
```

Example 20: Print first 4 natural numbers

```
var=1          # var is a counter that will execute from 1 to 4
num=4          # Till this loop should go
while var<=num:
   print(var,end=" ")  # end=" " is used to print each element with a space
```

```
    var=var+1
#Output
1 2 3 4
```

Explanation:
Step 1. var is initialized with value 1.

Step 2. num is initialized with value 4 as the condition will go till this value.

Step 3. Condition is checked (1 <= 4) as var=1 and num=4 and it is true.

3a) 1 is printed 1st time with space.

3b) Value of var is updated to 2.

Step 4. Condition is checked (2 <= 4) as var=2 and num=4 and it is true.

4a) 2 is printed 2nd time with space.

4b) Value of var is updated to 3.

Step 5. Condition is checked (3 <= 4) as var=3 and num=4 and it is true.

4a) 3 is printed 3rd time with space.

4b) Value of var is updated to 4.

Step 6. Condition is checked (4 <= 4) as var=4 and num=4 and it is true.

4a) 4 is printed 4th time with space.

4b) Value of var is updated to 5.

Step 7. Condition is checked (5 <= 4) as var=5 and num=4 and it is false.

Step 8. The flow goes outside the while loop

Example 21: Make a program to find sum of first 15 natural numbers.

```
sum_numbers = 0 #initialise variable sum_numbers=0, it will store sum of first 15 numbers
num=15              # Till this loop should go
var=1               # var is a counter that will execute from 1 to 15
while var<=num:
    sum_numbers=sum_numbers+var #everytime loop runs var value is added to sum_numbers
    var=var+1
print("Sum is ",sum_numbers)
#output
Sum is 120
```

Example 21: Print sum of numbers from 10 to -5 using while loop.

```
var=10          # var is a counter that will execute from -5 to 10
num=-5          # Till this loop should go
sum_numbers = 0 #initialise variable sum_numbers=0
while var>=num:
   sum_numbers=sum_numbers+var #everytime loop runs var value is added to sum_numbers
   var=var-1 # decrement the var
print("Sum is ",sum_numbers)
#Output
Sum is 40
```

3.3.1.1 Use of else with While Loop

As discussed, the loop iterates over a block of code repetitively or repeatedly until the condition is satisfied or True and when it becomes dissatisfied or False, the program jumps after the immediate line of the loop. The else part is executed when the condition becomes False or dissatisfied. Only break and exceptions can hold the execution of else part.

Syntax:

```
initialization
while(condition):
        statement 1
        statement 2  #optional
        …….
        …….
        …….
        statement n #optional
        increment/decrement  # Order of statements and increment/decrement is not specific
else:                   #optional
   statement(s)
```

Example 22: Make a program to find sum of first 15 natural numbers by using else with while loop.

```
sum_numbers = 0 #initialise variable sum_numbers=0, it will store sum of first 15 numbers
num=15              # Till this loop should go
var=1               # var is a counter that will execute from 1 to 15
while var<=num:
   sum_numbers=sum_numbers+var #everytime loop runs var value is added to sum_numbers
   var=var+1
else:
   print("Sum is ",sum_numbers)
#output
Sum is 120
```

3.3.2 For Loop

In python, ‘for loop’ is used to iterate over sequences like tuple, list, string, etc. and iterable objects. It will iterate till the last value of the sequence

Syntax:

```
for variable in sequence:                    # Sequence can be list, tuple, string
                    Body of for loop
```

Example 23: # Program to find the sum of first 15 natural numbers

```
sum_numbers = 0 #initialise variable sum_numbers=0, it will store sum of first 15 numbers
num=range(1,16)
# iterate over the list
for var in num:  # take one value from 1 till 15 after every iteration
   sum_numbers = sum_numbers+var
print("Sum is", sum_numbers)
```

#Output is

Sum is 120

Explanation:

First of all, understand the inbuilt range() function. range function syntax is range(start, stop, step_size). By default, step_size is 1.

Here range(1,16) or range(1,16,1) are the same and this will generate numbers from 1 to 15, remember 16 will not be generated. It will go to 15.

Example 24: # Program to find the sum of the first 15 natural numbers stored in a list

```
num = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]  # List
sum_numbers = 0 #initialise variable sum_numbers=0, it will store sum of first 15 numbers
# iterate over the list
for var in num:
   sum_numbers = sum_numbers+var
print("Sum is", sum_numbers)
```

#Output

Sum is 120

Example 25: Program to iterate over sequences

```
# Iterating over a list
print("Iteration over List Sequence")
grad_l = ["CSE","ME","Civil","ECE"]
for var in grad_l:
   print(var,end=" ")
# Iterating over a tuple
print("\nIteration over Tuple Sequence")
```

```
grad_t = ["CSE","ME","Civil","ECE"]
for var in grad_t:
    print(var,end=" ")
 # Iterating over a String
print("\nIteration over String")
grad_s = ["Btech"]
for var in grad_s:
    print(var,end=" ")
# Iterating over Dictionary
print("\nIteration over Dictionary")
grad_d = dict()
grad_d['CSE']=1
grad_d['Civil'] =2
grad_d['ECE']=3
grad_d['ME']=4
for var in grad_d :
    print("%s  %d" %(var, grad_d[var]))
#OUTPUT
Iteration over List Sequence
CSE ME Civil ECE
Iteration over Tuple Sequence
CSE ME Civil ECE
Iteration over String
Btech
Iteration over Dictionary
CSE  1
Civil  2
ECE  3
ME  4
```

3.3.2.1 Iterating over Sequences using Index

In the sequence, the index of the elements is used for iteration. The main idea here is to first calculate the length of the sequence and then use the range function that helps in iteration till the length of the sequence.

Example 26: Iteration over a sequence using index using range and len function

```
grad_l = ["CSE","ME","Civil","ECE"]
for index in range(len(grad_l)):
    print(index, grad_l[index])
#OUTPUT
0 CSE
```

```
1 ME
2 Civil
3 ECE
```

Example 27: Iteration over a sequence using enumerate function

```
grad_l = ["CSE","ME","Civil","ECE"]
for index,value in enumerate(grad_l):
   print(index,value)
#Output
0 CSE
1 ME
2 Civil
3 ECE
```

Example 28: Iteration over a sequence using zip function

```
grad_l = ["CSE","ME","Civil","ECE"]
for index in zip(range(len(grad_l)),grad_l):
            print(index)
#Output
(0, 'CSE')
(1, 'ME')
(2, 'Civil')
(3, 'ECE')
```

3.3.2.2 Use of else with for Loop

As discussed, for loop iterate till the last element of the sequence and sequence can be list, string, tuple, and other iterable objects. The else part is executed when the block finishes the execution.

Syntax:

```
for variable in sequence:                    # Sequence can be list, tuple, string
                        Body of for loop
else:
   Body of else block
```

Example 29: Print 0 to 6 numbers

```
num=int(input())
for var in range(num): # take one value from 0 till 6 after every iteration, as range goes till num1
   print(var,end=" ")
else:
   print(“Inside else block”)
#Input
7
#Output
```

0 1 2 3 4 5 6
Inside else block

3.3.3 Nested Loops

In python, one loop can be used inside other loops. *while* loop can be used within other *while* loops, *for* loop can be used within other *for* loop, *for* loop can be used within *while* loop and vice-versa.

Syntax for nested *while* loop

```
while(condition1):  # Outer Loop
        while(condition2): # Inner Loop
                        statement(s) # Inside Inner Loop
        statement(s) #
```

Syntax for nested *for* loop:

```
for var1 in sequence: # Outer loop
        for var2 in sequence: # Inner Loop
                        statement(s)  # Inside Inner Loops
        statement(s) #
```

NOTE: Many levels of nesting can be done for loops

Example 30 Print the following sequence using while loop

```
*
*       *
*       *       *
*       *       *       *
```

Note: There is tab space between elements in a line.

Explanation: First look outer loop is running, how many times. The vertically here shows the outer loop execution. It shows four times. Now look at inner loop, try to understand the following:

when outer loop runs 1st time, inner loop runs 1 time (horizontal values), print *

when outer loop runs 2nd time, inner loop runs 2 times (horizontal values), print * two times with tab space

when outer loop runs 3rd time, inner loop runs 3 times (horizontal values), print * three times with tab space

when outer loop runs 4th time, inner loop runs 4 times (horizontal values), print * four times with tab space

Solution:

```
outer_var=1
while(outer_var<=4):
   inner_var=1
```

```
    while(inner_var<=outer_var):
        print('*',end="\t")#end="\t" for tab space between elements in a line
        inner_var=inner_var+1
    print("\n") #Next line
    outer_var=outer_var+1
```

Example 31: Print the following sequence using while loop

```
1
1       2
1       2       3
1       2       3       4
```

Note: There is tab space between elements in a line.

Explanation: First look outer loop is running, how many times. The vertically here shows the outer loop execution. It shows four times. Now, look at the inner loop, try to understand the following:

when outer loop runs 1st time, inner loop runs 1 time (horizontal values), print 1

when outer loop runs 2nd time, inner loop runs 2 times (horizontal values), print 1,2 with tab space

when outer loop runs 3rd time, inner loop runs 3 times (horizontal values), print 1,2,3 with tab space

when outer loop runs 4th time, inner loop runs 4 times (horizontal values), print 1,2,3,4 with tab space

Solution

```
outer_var=1
outer_cond=int(input())
while(outer_var<=outer_cond):
    inner_var=1
    while(inner_var<=outer_var):
        print(inner_var,end="\t")
        inner_var=inner_var+1
    print()#New line
    outer_var=outer_var+1
#Input
4
#Output
1
1       2
1       2       3
1       2       3       4
```

Example 32: Print the following sequence using for loop

1
2 3
4 5 6
7 8 9 10

Solution (Step wise procedure is explained in Table 4.1)

```
var=1 # It is used to print 1 2 3 4 5 6 7 8 9 10
for outer_var in range(1, 5): # The loop will work for fout times i.e. 5-1=4
    for inner_var in range(outer_var):
        print(var, end=' ')
        var=var+1
    print()
```

Table 3.1: Example 32 step wise procedure explained

Step Numbers	outer_var	Var	inner_var	Printing Pattern
Step 1	1	1	1	1
Step 2		2	2	
Step 3	2	2	1	1 2
Step 4		3	2	1 2 3
Step 5		4	3	
Step 6	3	4	1	1 2 3 4
Step 7		5	2	1 2 3 4 5
Step 8		6	3	1 2 3 4 5 6
Step 9		7	4	
Step 10	4	7	1	1 2 3 4 5 6 7
Step 11		8	2	1 2 3 4 5 6 7 8

Step 12		9	3	1 2 3 4 5 6 7 8 9
Step 13		10	4	1 2 3 4 5 6 7 8 9 10
Step 14		11	5	
Step 15	5			

Example 33: Print the following sequence

5 4 3 2 1
4 3 2 1
3 2 1
2 1
1

Solution (Step wise procedure is explained in Table 3.2)

```
outer_var=1
while(outer_var<=5):
    inner_var=6-outer_var
    while(inner_var>=1):
        print(inner_var, end='\t')
        inner_var-=1
    outer_var+=1
    print()
```

Explanation: In the print sequence column, due to the space limitation of a page, tab space is shown as space.

Table 3.2: Step wise procedure of example 33

Step Numbers	outer_var	outer_var<=5	inner_var	inner_var>=1	Print the sequence
Step 1	1	1<=5	5	5>=1	5
Step 2			4	4>=1	5 4
Step 3			3	3>=1	5 4 3
Step 4			2	2>=1	5 4 3 2
Step 5			1	1>=1	5 4 3 2 1
Step 6			0	0>=1	
Step 7	2	2<=5	4	4>=1	5 4 3 2 1 4

Step 8			3	3>=1	5 4 3 2 1 4 3
Step 9			2	2>=1	5 4 3 2 1 4 3 2
Step 10			1	1>=1	5 4 3 2 1 4 3 2 1
Step 11			0	0>=1	
Step 12	3	3<=5	3	3>=1	5 4 3 2 1 4 3 2 1 3
Step 13			2	2>=1	5 4 3 2 1 4 3 2 1 3 2
Step 14			1	1>=1	5 4 3 2 1 4 3 2 1 3 2 1
Step 15			0	0>=1	
Step 16	4	4<=5	2	2>=1	5 4 3 2 1 4 3 2 1 3 2 1 2
Step 17			1	1>=1	5 4 3 2 1 4 3 2 1 3 2 1 2 1
Step 18			0	0>=1	
Step 19	5	5<=5	1	1>=1	5 4 3 2 1 4 3 2 1 3 2 1 2 1 1
Step 20			0	0>=1	
Step 21	6	6<=5			

Example 34: Print Armstrong Numbers between 50 to N (Nested for and while loop)

```
start_range = 50
end_range = int(input())
for chk_num in range(start_range, end_range + 1):
    order_num = len(str(chk_num))#number of digits in a number
    sum = 0
```

```
    temp_num = chk_num
    while temp_num > 0:
        digit = temp_num % 10
        sum += digit ** order_num
        temp_num //= 10
    if chk_num == sum:
        print(chk_num)# Armstrong Number printing
```

Test Case 1:

#Input

500

#output

153

370

371

407

Test Case 2:

#Input

200

#Output

153

Explanation: A number is said to be an Armstrong number of order n if

pqr… = $p^n + q^n + r^n + \ldots\ldots$ where n is number of digits in the number

Example: $370 = 3^3 + 7^3 + 0^3 = 370$

Practice Questions

Example 35: Write a program to generate the pattern

1

10

101

1010

10101

101010

……..

…….

Sample Input 1:

4

Sample Output1:

1

10

101
1010
Sample Input 2:
3
Sample Output2:
1
10
101
Sample Input3:
6
Sample Output3
1
10
101
1010
10101
101010

Solution (Step wise procedure is explained in Table 3.3):

```
int_number=int(input())
outer_var=1
while outer_var<=int_number:
    count=1
    inner_var=1
    while inner_var<=outer_var:
        if count%2!=0:
            print("1",end="");
        else:
            print("0",end="");
        count=count+1
        inner_var=inner_var+1
    outer_var=outer_var+1
    print("")
```

Table 3.3: Step wise procedure of example 35 with int_numbers=5

Step Numbers	outer_var	Outer Condition	count	inner_var	Inner Condition	Printing Sequence
Step 1	1	1<=5	1	1	1<=1	1
Step 2			2	2	2<=1	
Step 3	2	2<=5	1	1	1<=2	1

						1
Step 4			2	2	2<=2	1 10
Step 5			3	3	3<=2	
Step 6	3	3<=5	1	1	1<=3	1 10 1
Step 7			2	2	2<=3	1 10 10
Step 8			3	3	3<=3	1 10 101
Step 9			4	4	4<=3	
Step 10	4	4<=5	1	1	1<=4	1 10 101 1
Step 11			2	2	2<=4	1 10 101 10
Step 12			3	3	3<=4	1 10 101 101
Step 13			4	4	4<=4	1 10 101 1010
Step 14			5	5	5<=4	
Step 15	5	5<=5	1	1	1<=5	1 10 101 1010 1
Step 16			2	2	2<=5	1 10 101 1010

						10
Step 17			3	3	3<=5	1 10 101 1010 101
Step 18			4	4	4<=5	1 10 101 1010 1010
Step 19			5	5	5<=5	1 10 101 1010 10101
Step 20			6	6	6<=5	
Step 21	6	6<=5				

Example 36: Make a program that tells whether the input positive integer is the sum of four consecutive numbers. Print ‘Thumbs Up’ when it is the sum of four consecutive numbers else print ‘Thumbs Down’.
Input: A positive Integer
Output: Thumbs Up or Thumbs Down

Test Case1:
#Input
6
#Output
Thumbs Up #0+1+2+3

Test Case2:
#Input
16
#Output
Thumbs Down

Test Case3:
#Input
86
#Output

Thumbs Up # 20+21+22+23
Solution:

```
int_number=int(input())
temp_res=int_number//4
if(((temp_res-1 + temp_res-2 + temp_res-3 + temp_res)==int_number) or ((temp_res-1 +
temp_res-2 + temp_res + temp_res+1)==int_number) or ((temp_res-1 + temp_res + temp_res+1
+ temp_res+2)==int_number) or ((temp_res + temp_res+1 + temp_res+2 +
temp_res+3)==int_number)):
   print("Thumbs Up")
else:
   print("Thumbs Down")
```

Example 37: Make a program for finding out whether a number is a prime number or not.
Test Case 1:
#Input
7
#Output
Prime number

Test Case 2:
#Input
23
#Output
Prime number

Test Case 3:
#Input
72
#Output
Not a Prime number

Test Case 4:
#Input
700
#Output
Not a Prime number

Solution:
Method 1:

```
int_number =int(input())
flag_prime = True # flag variable
if int_number > 1:
```

```
    for var in range(2, int_number):
        if (int_number % var) == 0:
            flag_prime = False
            break
if flag_prime==True:
    print("Prime number")
else:
    print("Not a Prime number")
```

Method 2:

```
int_number =int(input())
if int_number > 1:
    for var in range(2, int_number):
        if (int_number % var) == 0:
            break
if (var==(int_number-1) or (var == int_number)):
    print("Prime number")
else:
    print("Not a Prime number")
```

Method 3:

```
int_number =int(input())
if int_number > 1:
   for var in range(2,int_number):
       if (int_number % var) == 0:
           print("Not a Prime number")
           break
   else:
       print("Prime number")
else:
   print("Not a Prime number")
```

3.4 SELF-CHECK QUESTIONS

1. Is else part is mandatory while using for and while loop? True/False
2. Is for loop can be nested in while loop? True/False
3. Is while loop can be nested in for loop? True/false
4. What is the output of the following code?

```
if 10+7==17:
    print("Yes")
else:
    print("No")
```

```
print("Last Line")
```

a) Yes
   Last Line
b) No
   Last Line
c) Yes
   No
   Last Line
d) Last Line

5. What is the output of the following code?

```
var = 6
if (var > 5):
   var = var * 3;
if (var > 10):
   var = 0;
print(var)
```

a) 0
b) 18
c) 18
   0
d) None of these

3.5 SUMMARY

Conditional statements help the students in learning the flow of the program and these statements can be executed in python when certain conditions are met. Different examples are elaborated for if, if-else, if-elif-else along with nested conditional statements. The students will be able to understand different types of loops like for and while. They will be able to make programs and dry run their code. Step by step procedure has been explained with many examples of looping constructs. This module targets to inculcate the basics of loops and conditional statements in students.

3.6 Practice Questions

1. Which of the following statements will be executed in python version 3?
   a) if (7,6): print("CSE-CA")
   b) if (7,6):
      print("CSE-CA")
   c) if (7,6):

      print("CSE-CA")

d) if (7,6):
    print("CSE-CA")

2. Let two variables have been initialized as:
   var1=23
   var2=7
   Write a python program to find the remainder when you divide var1 by var2 and assign the result to a variable result_var?
3. Is the following code has valid syntax or not?
   var1=12
   var2=6
   if var1 > var2: if var1 > 10: print('CSE-CA')
4. What will be the output of the following code?
   x=12
   y=6
   if x < y: print('CSE')
   elif y < x: print('CA')
   else: print('CSE-CA')

   a) CSE
   b) CA
   c) CSE-CA
   d) Error
5. Correctly choose the options that will tell how many times the loop will work?
   var1=-5
   while var1>0:
      print("CSE")
      print("CA")
      print("CSE-CA")
      var1=var1+1
   a) 0 times
   b) -5 times
   c) 5 times
   d) Infinite times
6. What will be the output of the following code?
   x = 2.00
   if (x - int(x) == 0):
      print("Whole Number")
   else:
      print("Has decimals")

a) Whole Number
b) Has decimals
c) 2.00
d) Error

7. What will the output of the following code?

```
sum_numbers = 0
num=range(7)
for var in num:
   sum_numbers = sum_numbers+var
print(sum_numbers)
```

a) 28
b) 21
c) 15
d) 0

8. Choose the correct output

```
grad_l = ["CSE","ME","Civil","ECE"]
for index in range(len(grad_l)):
   print(len(grad_l))
```

a) 4
   4
   4
   4
b) 3
   3
   3
   3
c) CSE
   ME
   Civil
   ECE
d) None of the option is correct

9. Construct a program for doing a reverse of an N-digit number using for and while loop both.
10. Elaborate differences between while and for loop.

B.Sc.(DATA SCIENCE)

SEMESTER-I

PROBLEM SOLVING USING COMPUTERS

UNIT IV: METHODS AND FUNCTIONS

STRUCTURE

4.0 OBJECTIVES

- Familiarize how to build, define and write functions
- Learn how to use different types of arguments in the function definition
- Learn and examine how lambda functions are accessed and used.
- Usage of filter, map and reduce functions

4.1 INTRODUCTION

Functions are mainly used to perform particular tasks for programmers, users and associated stock holders. They basically used to cope with the input and output of the computer programs. The most important element in any real time projects is data and functions are the best and effective way to deal with projects that can be small, medium, large as well as complex. When we work on large and complex problems or projects, data duplication is a problem. To avoid this drawback, codes are reusable with the help of functions. This module targets to give better understanding of inbuilt functions, user-defined functions along with lambda functions. Different examples of each type of functions are mentioned along with proper output. Many inbuilt functions are explained like max(), abs(), type() etc., with proper syntax and programs. Various programs of user defined functions are also elaborated. Different types of arguments in functions passing are discussed with appropriate programs. Arguments like positional, keyword, default and special cases of variable-length arguments. At the end, lambda functions explanation along with filter function, map function and reduce function are described and discussed extensively followed with self-check questions, summary and unit end questions.

4.2 FUNCTIONS

A function in python is a block of statements that do a particular work or task, this work or task can be related to any logical, any computational or any evaluation task. The idea is to combine the common statements such that code is reusable and helps in avoiding writing the same code again and again. Functions help in dividing the larger programs into smaller blocks that help in managing the whole program easily.

Functions are divided into two types:

1. Inbuilt functions
2. User-defined functions
3. Anonymous functions

4.2.1 Inbuilt functions

The python interpreter has many functions built into it and these functions have their pre-defined functionalities. Some of these inbuilt functions are listed below:

a) abs()
b) print()
c) input()

d) chr()
e) max()
f) min()
g) int()
h) float()
i) type()
j) round()

Inbuilt Functions with Examples

a) abs() – This inbuilt function returns the absolute value of a number and this number can be an integer, complex number or float. This function takes only one argument and it returns the absolute value if the passed number is an integer or float number and it returns the magnitude of the number if the passed number is complex.

Example 1: # Python code for abs() built-in function

```
int_var     = -101 #Initialising variable with an integer value
float_var   = -46.87 #Initialising variable with an float value
complex_var = (5 - 12j) #Initialising variable with an complex value
print("Absolute Value",abs(int_var))
print("Absolute Value",abs(float_var))
print("Absolute Value",abs(complex_var))
#Output
Absolute Value 101
Absolute Value 46.87
Absolute Value 13.0
```

b) print() – It generates the output from a passed value or many values. The output can be in the form of a screen/standard output device or text stream file. In the print() function, you can have more zero or more expressions and all these expressions are separated by using a comma operator. print() functions have five arguments as expressions and these are mentioned in syntax along with its explanation. All these arguments are optional and keyword arguments.

Syntax:

```
print(*object, sep=' ', end='\n', file=sys.stdout, flush=False)
```

where

*object – object denotes the screen output and * indicates the number of objects as screen output.
sep – object denotes the screen output and is separated using the sep value. The default value of sep=’ ‘
end – It is used to print at the last

file – By default, its value is sys.stdout and must use import sys in the program if you use sys.stdout and it helps in printing objects on the screen. You can also use any object having a write(string) method instead of the default value.
flush – The internal buffer is forcibly flushed if its value is True. By default its value is False.

Example 2: Usage of print() with many objects

```
print("CSE","CA")
print("Btech")
print()
print("ECE","ME")
#Output
CSE CA
Btech

ECE ME
```

NOTE: At the end of the print() line, the new line is inserted as end='\n' by default. One more thing to be noted in the first and fourth print() function, multiple objects are mentioned and when they are printed there is space between them as sep=' ' by default.

Example 3: Usage of print() function with sep argument

```
print("CSE","CA",sep="\t")
print("Btech")
print()
print("ECE","ME",sep="**")

#Output
CSE	CA
Btech

ECE**ME
```

Example 4: Usage of print() function with sep and end arguments

```
print("CSE","CA",sep="\t",end='\n\n')
print("Btech")
print()
print("ECE","ME",sep="**",end='&&')
#Output
CSE	CA

Btech
```

ECE**ME&&

Example 4: Usage of print() function with file argument

```
Python_src_file =open('python_basics.txt', 'w')
print("Jagat Guru Nanak Dev PSOU", file= Python_src_file)
Python_src_file.close()
```

c) input() – In this inbuilt function, the user enters the input and then this inbuilt function assesses this input expression whether it is correct or not. If it is correct then it is assigned to the variable unless it generates a syntax error or an exception is raised.

Syntax:

```
input([prompt])   # here, prompt is optional
```

Example 5:

```
str_var = input("Enter Official name")
print(str_var)
int_var=int(input(" Enter Emp-Id"))  #typecasting to integer, here int() is another inbuilt function
print(int_var)
float_var=float(input("Enter Gross Salary"))
# typecasting to float, here float() is another inbuilt function
print(float_var)
#Output
Enter Official nameVinay
Vinay
Enter Emp-Id178
178
Enter Gross Salary26734.56
26734.56
```

d) chr() – This inbuilt function takes the parameter of integer type (valid Unicode) and it returns a character corresponding to that passed integer.

Example 6: Print the characters using chr()

```
print(chr(79))
print(chr(107))
print(chr(1178))
print(chr(45))
#Output
O
k
```

Ķ
-

e) max() – It takes a python object or many objects (optional) as argument(s) with another argument ‘key’ and it is optional. Key is the function that compares the objects. One more argument is default and it is also optional. If an object is empty then the default value is used. This function returns the maximum value if it is an integer or float object. If it is a string then it returns a lexographic value. If the objects are iterable like lists, tuples, or dictionary then it returns the largest item of the iterable.

Syntax:
max(object1,object 2, … object n, key, default)

Example 7: Usage of max() function with many objects as arguments

```
int_var1=45
float_var2=78.23
float_var3=34.893
max_var=max(int_var1,float_var2,float_var3)
print(max_var)
#Output
78.23
```

Example 8: Usage of max() function with list as an argument

```
list_var=[45,78.23,34.893]
max_var=max(list_var)
print(max_var)
#Output
78.23
```

Example 9: Usage of max() function with key=len as an argument

```
str_var1 = "CSE-CA"
str_var2 = "Btech"
str_var3 = "CSE-ME-ECE-CA"
max_var = max(str_var1, str_var2, str_var3,key = len)
print(max_var)
#Output
CSE-ME-ECE-CA
```

f) min() – This function returns the minimum value if integer or float values are passed as argument(s). And it returns lexographic smallest value if the strings are passed as argument(s).

Syntax:
min(object1, object2 …… object n, key, default)
where object 1 is integer, float, string, list, tuple, dictionary
object 2, object 3….. object n are optional
key is optional and this function takes all the passed objects and comparison is performed
the default value is allotted if the given objects are empty

Example 10: Usage of min() function with many objects as arguments

```
int_var1=45
float_var2=78.23
float_var3=34.893
min_var=min(int_var1,float_var2,float_var3)
print(min_var)
#Output
34.893
```

Example 11: Usage of min() function with list as an argument

```
list_var=[45,78.23,34.893]
min_var=min(list_var)
print(min_var)
#Output
34.893
```

Example 12: Usage of min() function with key=len as an argument

```
str_var1 = "CSE-CA"
str_var2 = "Btech"
str_var3 = "CSE-ME-ECE-CA"
min_var = min(str_var1, str_var2, str_var3,key = len)
print(min_var)
#Output
Btech
```

g) int() – This functions converts the specified value as argument into an integer number [1].
Syntax:
int(string, any_base)
where the string is a combination of elements of 1's and 0's
any_base indicates any base of the number

Example 13: Usage of int()

```
binary_var="110"
octal_var="110"
hexa_var="A0A"
```

```
decimal_convertfrom_binary=int(binary_var,2)
print(decimal_convertfrom_binary)
decimal_convertfrom_octal=int(octal_var,8)
print(decimal_convertfrom_octal)
decimal_convertfrom_hexa=int(hexa_var,16)
print(decimal_convertfrom_hexa)
$Output
6
72
2570
```

Example 14: Converting string to int()

```
str_var=input("Enter the string variable value")
print(str_var)
print(type(str_var))
int_var=int(input("Enter the string variable value"))
print(int_var)
print(101+int_var)
print(type(int_var))
#Output
Enter the string variable value123
123
<class 'str'>
Enter the string variable value123
123
224
<class 'int'>
```

h) float() – This inbuilt function converts the argument value of number or string into float value [2], [3].

Syntax:

```
float([string or number])
```

Example 15: float() function usage

```
float_var1= (float(101))
print(float_var1)
print(type(float_var1))
float_var2= (float(101.67))
print(float_var2)
```

```
print(type(float_var2))

float_var3= (float("101.87"))
print(float_var3)
print(type(float_var3))
float_var4= (float(5e003))
print(float_var4)
print(type(float_var4))
float_var5= (float(5e-003))
print(float_var5)
print(type(float_var5))
float_var6= (float(False))
print(float_var6)
print(type(float_var6))
float_var7= (float(True))
print(float_var7)
print(type(float_var7))
float_var8= (float('abc'))
print(float_var8)
print(type(float_var8))
#output
101.0
<class 'float'>
101.67
<class 'float'>
101.87
<class 'float'>
5000.0
<class 'float'>
0.005
<class 'float'>
0.0
<class 'float'>
1.0
<class 'float'>
ValueError
```

i) type() – It is one method that is widely used for debugging and it returns the class type of the object that Is passed as an argument to the type() method.

Example 16: Usage of type() inbuilt function

```
int_var=10
print(type(int_var) is int)
float_var = 10.45
print(type(float_var) is float)
list_var=[1,3,5]
print(type(list_var) is list)
tuple_var=(1,3,5)
print(type(tuple_var) is list)
float_var = 10.45
print(type(float_var) is not float)
tuple_var=(1,3,5)
print(type(tuple_var) is not list)
#Output
True
True
True
False
False
True
```

j) round() - This is one of the inbuilt functions in Python and it rounds off the number to the parameter value (i.e. ndigits decimal) and if no parameter value is mentioned then it rounds off to the nearest integer [5].

Syntax:
round(number, [ndigits decimal])
where
number is to whom which rounded to be done
ndigits decimal tell up to what decimals rounding of is required and it is optional

Example 17: round() function basic example without optional parameter

```
# Use Case for integers
print(round(123))

# Use Case for floating-point numbers
print(round(123.7))

# Use Case for floating-point numbers
print(round(123.2))

# Use Case for floating-point numbers
```

```
print(round(123.5))

#Output
123
124
123
124
```

Example 18: round() function basic example with optional parameter

```
print(round(15.455,2))
print(round(15.453,2))
print(round(15.457,2))
#Output
15.46
15.45
15.46
```

Example 19

```
print(round("CSE",2))

#Output
TypeError
```

NOTE: If any input other than the number is given in parameter value 'number' then it generates an error i.e. Type Error. If any input other than a number is given in parameter value 'ndigits decimal' then it generates and error i.e. TypeError.

4.2.2 User-Defined Functions

The functions that are defined by the user are known as user-defined functions. The user-defined functions can have any name excluding space, pre-defined keywords, and any special character.

Syntax:

```
def function-name(parameters):
        statement(1)
        statement(2) #optional
        ……. #optional
        ……. #optional
        statement(n) #optional
        """optional documentation string"""
        return #Optional return statement
```

where

def is the keyword and it tells the start of the function header

function-name differentiate the functions used in the program and it follows naming conventions of identifiers and it is unique in nature.
parameters- they are also known as arguments, the values to the function are passed using parameters or arguments and they are optional in the user-defined functions.
: colon tells the end of the function header
Optional documentation string- It is used to tell what the functions do in the program and they are optional to define in the functions
statement(s) – the python user-defined function consists of one or more than one valid statements
return – This helps in returning the value from the function and it is optional in a user-defined function.

Example 20: First Simple example of a function

```
# Construct a user-defined function to print COVID message when it is called
def bye_COVID():
    print("Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID")
    print("Do follow social distancing")
    print("Properly use mask")
    print("Avoid unnecessary shopping and walking-out")

#Call the function to print COVID message
bye_COVID()
```

```
#Output
Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID
Do follow social distancing
Properly use mask
Avoid unnecessary shopping and walking-out
```

Example 21: Simple example with a parameter

```
# Construct a user-defined function to print COVID message when it is called with a parameter
def bye_COVID(name):
    print("Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID")
    print("Do follow social distancing")
    print("Properly use mask")
    print("Avoid unnecessary shopping and walking-out")

# Take input from the user
year_name=input("Enter the year in which it came into existence\n")

#Call the function to print COVID message
bye_COVID(year_name)
```

```
#Output
Enter the year in which it came into existence
19
Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID
Do follow social distancing
Properly use mask
Avoid unnecessary shopping and walking-out
```

Example 21: Simple example without passing an appropriate parameter

```
# Construct a user-defined function to print COVID message when it is called with a parameter
def bye_COVID(name):
    print("Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID")
    print("Do follow social distancing")
    print("Properly use mask")
    print("Avoid unnecessary shopping and walking-out")

# Take input from the user
year_name=input("Enter the year in which it came into existence\n")

#Call the function to print COVID message
bye_COVID()

#Output
Enter the year in which it came into existence
19
TypeError: bye_COVID() missing 1 required positional argument: 'name'
```

Example 22: Use of return statement in function

```
def sum_of_numbers(num_var1,num_var2,num_var3):
    variables_sum=num_var1+num_var2+num_var3
    return(variables_sum)

variables_summation=sum_of_numbers(13,45,78)
print("Sum is depicted as", variables_summation)

#Output
Sum is depicted as 136
```

Example 23: Use of return when strings are passed as arguments

```
# Construct a user-defined function to print COVID message when it is called with a parameter
def bye_COVID(name,new_variant):
```

```
    print("Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID")
    print("Do follow social distancing")
    print("Properly use mask")
    print("Avoid unnecessary shopping and walking-out")
    total=name+new_variant
    return(total)

# Take input from the user
year_name=input("Enter the year in which it came into existence\n")
variant_number=input("Enter the latest variant that is coming\n")
#Call the function to print COVID message
result=bye_COVID(year_name,variant_number)
print(result)

#Output
Enter the year in which it came into existence
2019
Enter the latest variant that is coming
2nd-variant
Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID
Do follow social distancing
Properly use mask
Avoid unnecessary shopping and walking-out
20192nd-variant

Example 25: Use of return without returning any value
def sum_of_numbers(num_var1,num_var2,num_var3):
    variables_sum=num_var1+num_var2+num_var3
    return

variables_summation=sum_of_numbers(13,45,78)
print(variables_summation)

#Output
None
```

NOTE: The arguments/parameters which are specified in the function definition are called formal arguments whereas the arguments/parameters which are specified in the function call are called actual arguments.

4.2.2.1 Function Definition Arguments or Function Formal Arguments

Four types of formal arguments are specified in user-defined functions and these are mentioned below:

A) Positional or Required Arguments
B) Keyword Arguments
C) Default Arguments
D) Variable-length Arguments

A) Positional or Required Arguments

The correct position and an exact number of the arguments are passed to a function or in other words function call and function definition number of arguments must be the same along with the correct position of these arguments is required.

Example 26: Positional Arguments Program

```
def bye_COVID(injection1,injection2):
# Look carefully, first argument have value of "True" and Second have "Yes" values
   if injection1=="True" and injection2=="Yes":
      print("Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID")
      print("Do follow social distancing")
      print("Properly use mask")
      print("Avoid unnecessary shopping and walking-out")

# Take input from the user
var_vaccination1=input("Enter Yes or No if vaccination1 has been done or not\n")
var_vaccination2=input("Enter True or False if vaccination2 has been done or not\n")
#Call the function to print COVID message
bye_COVID(var_vaccination2,var_vaccination1)  # Look arguments calling

#Output
Enter Yes or No if vaccination1 has been done or not
Yes
Enter True or False if vaccination2 has been done or not
True
Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID
Do follow social distancing
Properly use mask
Avoid unnecessary shopping and walking-out
```

Example 27: No Positional Arguments in Function Definition but Two Values Are Specified in Function Calling

```
def bye_COVID(): # Look carefully, No formal argument
```

```
    if injection1=="True" and injection2=="Yes":
        print("Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID")
        print("Do follow social distancing")
        print("Properly use mask")
        print("Avoid unnecessary shopping and walking-out")

# Take input from the user
var_vaccination1=input("Enter Yes or No if vaccination1 has been done or not\n")
var_vaccination2=input("Enter True or False if vaccination2 has been done or not\n")
#Call the function to print COVID message
bye_COVID(var_vaccination2,var_vaccination1)  # Look arguments calling

#Output
Enter Yes or No if vaccination1 has been done or not
Yes
Enter True or False if vaccination2 has been done or not
True
TypeError: bye_COVID() takes 0 positional arguments but 2 were given
```

Example 28: Positional Arguments in Function Definition but no values are specified in Function Calling

```
def bye_COVID(injection1,injection2): # Look carefully, Two Positional Arguments
    if injection1=="True" and injection2=="Yes":
        print("Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID")
        print("Do follow social distancing")
        print("Properly use mask")
        print("Avoid unnecessary shopping and walking-out")

# Take input from the user
var_vaccination1=input("Enter Yes or No if vaccination1 has been done or not\n")
var_vaccination2=input("Enter True or False if vaccination2 has been done or not\n")
#Call the function to print COVID message
bye_COVID()  # Look arguments calling

#Output
Enter Yes or No if vaccination1 has been done or not
Yes
Enter True or False if vaccination2 has been done or not
True
TypeError: bye_COVID() missing 2 required positional arguments: 'injection1' and 'injection2'
```

B) Keyword Arguments

When arguments are passed in the function call, they can or cannot be in the order as formal arguments defined in the function definition. These things are achieved through keyword arguments. Remember that keyword argument must match the arguments of formal arguments [6].

Example 29: Keyword Arguments basic example 1

```
def BYE_COVID(var_string):
    print(var_string)
    return

# function calling
BYE_COVID(var_string = "Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID")
```

#Output

Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID

Example 30: Keyword Arguments basic example 2

```
def BYE_COVID(var_string1,var_string2):
    print(var_string1)
    print(var_string2)
    return

# function calling
BYE_COVID(var_string2 = "Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID", var_string1="Properly use mask")
```

#Output

Properly use mask

Take both vaccinations with a normal 4 to 6 weeks gap and say bye to COVID

C) Default Arguments

If in a function call, no explicit values are given then the formal arguments take default values [8].

Example 31: Keyword Arguments usage in calling function

```
def BYE_COVID(precaution1, precaution2="injection2", precaution3="Mask", precaution4="Social Distance",disease="COVID"):
    print("Take", precaution1, "followed with", precaution2, ",Wear", precaution3, "and Follow rules of", precaution4,"that helps in avoiding", disease)
```

```
# Keyword argument
BYE_COVID(precaution1="injection1")

# Keyword arguments
BYE_COVID(precaution1="injection1",precaution3="N95 or Fully Covered Mouth/Nose
Mask")

# Way of changing arguments
BYE_COVID(precaution3="N95 or Fully Covered Mouth/Nose
Mask",precaution1="injection1")

#Output
Take injection1 followed with injection2 ,Wear Mask and Follow rules of Social Distance that he
lps in avoiding COVID
Take injection1 followed with injection2 ,Wear N95 or Fully Covered Mouth/Nose Mask and Fo
llow rules of Social Distance that helps in avoiding COVID
Take injection1 followed with injection2 ,Wear N95 or Fully Covered Mouth/Nose Mask and Fo
llow rules of Social Distance that helps in avoiding COVID
```

Example 32: Deeper understanding of keyword arguments

```
def BYE_COVID(precaution1, precaution2="injection2", precaution3="Mask",
precaution4="Social Distance",disease="COVID"):
    print("Take", precaution1, "followed with", precaution2, ",Wear", precaution3, "and Follow
rules of", precaution4,"that helps in avoiding", disease)

# Invalid keyword
BYE_COVID(do_precaution1="injection1")    # Run this line individually

# No argument passes
BYE_COVID()  # Run this line individually

BYE_COVID(precaution2="inject","injection1") # Run this line individually

# Output
TypeError: BYE_COVID() got an unexpected keyword argument 'do_precaution1'
TypeError: BYE_COVID() missing 1 required positional argument: 'precaution1'
SyntaxError: positional argument follows keyword argument
```

4.2.2.2 Special Case of Keyword Arguments

In this **kwargs as an argument in the function definition is used and it means you can pass any number of keyword arguments along with variable length [7]. Remember the keyword name that is passed in function call must match the keyword name of the function definition.

Example 33: Basic Example of variable length arguments

```
def usage_of_kwargs(**kwargs):
   print(kwargs)

usage_of_kwargs(injection1=True, Mask="Yes", injection2=True, age=67)

#Output
{'injection1': True, 'Mask': 'Yes', 'injection2': True, 'age': 67}
```

Example 34: Another way of printing elements of method defined in previous example 31

```
def usage_of_kwargs(**kwargs):
   for var_key, var_value in kwargs.items():
      print ("%s == %s" %(var_key, var_value))

usage_of_kwargs(injection1=True, Mask="Yes", injection2=True, age=67)

#Output
injection1 == True
Mask == Yes
injection2 == True
age == 67
```

Example 35: Another way of printing elements in keyword arguments

```
def usage_of_kwargs(**kwargs):
   for var_key, var_value in kwargs.items():
      print("The key is {} and its associated value is {}".format(var_key, var_value))

usage_of_kwargs(injection1=True, Mask="Yes", injection2=True, age=67)

#Output
The key is injection1 and its associated value is True
The key is Mask and its associated value is Yes
The key is injection2 and its associated value is True
The key is age and its associated value is 67
```

4.2.2.3 Special Case of Positional Arguments

In this *var_arguments are used in function definition and the var_argumentss are non-keyword variable length arguments [6]. The * denotes to have any number of arguments.

Example 36: Usage of *var_arguments

```
def BYE_COVID(*var_arguments):
   for var_arg in var_arguments:
      print (var_arg)

BYE_COVID('Take injection1', 'Followed with injection2', 'Wear Mask Properly', 'Follow the rules of Social Distance')

#Output
Take injection1
Followed with injection2
Wear Mask Properly
Follow the rules of Social Distance
```

Example 37: Program to show *var_arguments having extra arguments.

```
def BYE_COVID(var_argument1,var_argument2,*var_arguments):
   print("The first argument value is",var_argument1)
   print("The second argument value is",var_argument2)
   for var_arg in var_arguments:
      print (var_arg)

BYE_COVID('COVID GUIDELINES','READ PROPERLY','Take injection1', 'Followed with injection2', 'Wear Mask Properly', 'Follow the rules of Social Distance')
#Output
The first argument value is COVID GUIDELINES
The second argument value is READ PROPERLY
Take injection1
Followed with injection2
Wear Mask Properly
Follow the rules of Social Distance
```

4.2.2.4 Pass by Reference or Pass by value

In python language, every variable is a reference. It means when we pass a variable to the function, every time a new object is created with a new reference.

Example 38: Largest of three numbers

```
def max_three_numbers(var_number1,var_number2,var_number3):
```

```
    if var_number1>var_number2:
        if var_number1>var_number3:
            var_largest=var_number1
        else:
            var_largest=var_number3
    else:
        if var_number2>var_number3:
            var_largest=var_number2
        else:
            var_largest=var_number3
    return(var_largest)
var_number1=int(input())
var_number2=int(input())
var_number3=int(input())
print(max_three_numbers(var_number1,var_number2,var_number3))

#Output
```
Test case1:
```
#Inputs
23
36
47
#Output
47
```
Test case1:
```
#Inputs
67
50
56
#Output
67
```
Test case1:
```
#Inputs
145
567
444
#Output
567
```

Example 39: Program to know whether a year is leap year or not a leap year

```
def leap_or_notleap(normal_year):
    if normal_year%4==0:
        if normal_year%100==0:
            if normal_year%400==0:
                flag=1
            else:
                flag=0
        else:
            flag=1
    else:
        flag=0

    return(flag)

normal_year = int(input())
flag_value=leap_or_notleap(normal_year)
if flag_value==1:
    print("Leap Year")
else:
    print("Not a Leap Year")

#Output
Test Case 1:
#Input
2000
#Output
Leap Year
Test Case 2:
#Input
1900
#Output
Not a Leap Year
Test Case 3:
#Input
2020
#Output
Leap Year
Test Case 4:
#Input
2021
```

#Output
Not a Leap Year

Example 40: Find factorial of a given number using user defined functions

```
def num_factorial(formal_number):
    if formal_number == 0:
        return 1
    else:
        return formal_number * num_factorial(formal_number-1)

var_number=int(input("Enter the number for factorial computation\n"))
print("The factorial is", num_factorial(var_number))
```

#Output
Enter the number for factorial computation
7
The factorial is 5040

Example 41: Write a program to generate the pattern using functions
A
AB
ABA
ABAB
ABABA
ABABAB
……..
…….
Sample Input 1:
4
Sample Output1:
A
AB
ABA
ABAB
Sample Input 2:
3
Sample Output2:
A
AB
ABA
Sample Input3:

6
Sample Output3
A
AB
ABA
ABAB
ABABA
ABABAB

Solution (Step wise procedure is explained in table 4.1):

```
def pattern_making(int_number):
    outer_var=1
    while outer_var<=int_number:
        count=1
        inner_var=1
        while inner_var<=outer_var:
            if count%2!=0:
                print("A",end="");
            else:
                print("B",end="");
            count=count+1
            inner_var=inner_var+1
        outer_var=outer_var+1
        print("")

int_number=int(input())
pattern_making(int_number)
```

Table 4.1: Pattern making by assuming int_number=5

Step Numbers	outer_var	Outer Condition	count	inner_var	Inner Condition	Printing Sequence
Step 1	1	1<=5	1	1	1<=1	A
Step 2			2	2	2<=1	
Step 3	2	2<=5	1	1	1<=2	A A
Step 4			2	2	2<=2	A AB
Step 5			3	3	3<=2	
Step 6	3	3<=5	1	1	1<=3	A AB

						A
Step 7			2	2	2<=3	A AB AB
Step 8			3	3	3<=3	A AB ABA
Step 9			4	4	4<=3	
Step 10	4	4<=5	1	1	1<=4	A AB ABA A
Step 11			2	2	2<=4	A AB ABA AB
Step 12			3	3	3<=4	A AB ABA ABA
Step 13			4	4	4<=4	A AB ABA ABAB
Step 14			5	5	5<=4	
Step 15	5	5<=5	1	1	1<=5	A AB ABA ABAB A
Step 16			2	2	2<=5	A AB ABA ABAB AB
Step 17			3	3	3<=5	A AB ABA ABAB ABA

Step 18			4	4	4<=5	1 A AB ABA ABAB ABAB
Step 19			5	5	5<=5	A AB ABA ABAB ABABA
Step 20			6	6	6<=5	
Step 21	6	6<=5				

4.2.3 Anonymous function or Lambda Function

The keyword used in the creation of lambda function is 'lambda' and they are popularly known as single line function. They are known as anonymous functions (a function without having a name). They are different from normal functions as this function does not use keywords like 'return' and 'def' [9].

Syntax:
Lambda multiple_arguments: expression
where expression returns an object and it is only one in the whole function
multiple_arguments used a comma to separate multiple arguments

Example 42: Example to differentiate between normal function and lambda function

```
def multiply_25(var_int1): # Normal function definition
   return var_int1*25

lambda_25 = lambda var_int1:var_int1*25 #lambda function definition

var_int1=int(input("User is entering the number which he/she is required to be multiply by 25 is
"))
print(multiply_25(var_int1))#Normal function calling
print(lambda_25(var_int1))#lambda function calling
```

#Input
User is entering the number which he/she is required to be multiply by 25 is 8
#Output
200
200

Example 43: Sum of three numbers for differentiating between def and lambda functions.

```
def summation_of_3num(var_int1,var_int2,var_int3):
    return var_int1+var_int2+var_int3

lambda_3num = lambda var_int1,var_int2,var_int3:var_int1+var_int2+var_int3

var_int1=int(input("User is entering the first integer number"))
var_int2=int(input("User is entering the second integer number"))
var_int3=int(input("User is entering the third integer number"))
print(summation_of_3num(var_int1,var_int2,var_int3))
print(lambda_3num(var_int1,var_int2,var_int3))

#Input
User is entering the first integer number23
User is entering the second integer number67
User is entering the third integer number40

#Output
130
130
```

4.2.3.1 Lambda with filter function

filter function is having two arguments, one argument is a function that helps in filtering and the other argument is iterator like list, tuples, sets, etc. [10]. It returns values that are passing the filtering condition. Only one iterator is passed in the filter function.

Syntax:

```
filter(lambda_function,iterator)
```

Example 44: Program to find out elements greater than 25 in a list

```
int_list1=[24,26,32,18,10,75]
result_greater_25 = filter(lambda var_int: var_int>25, int_list1)
print(type(result_greater_25))
print(list(result_greater_25))

#Output
<class 'filter'>
[26, 32, 75]
```

Explanation

1. In the first line, a list is defined with integer numbers.

2. In the second line, the result_greater_25 variable will store the values returned by the filter function.
3. In the second line, list each element is run by lambda function, and when filtering criteria is True (i.e. when list element > 25) then it returns its value.
4. In the third line, the type of the returned values are printed.
5. In the last line, the results are printed that are returned by the filter function.

Example 45: Program to multiple every element of list using filter function (look carefully at the output)

```
int_list1=[24,26,32,18,10,75]
result_multiply_3 = filter(lambda var_int: var_int*3, int_list1)
print(list(result_multiply_3))

#Output
[24, 26, 32, 18, 10, 75]# Here value is not modified
```

4.2.3.2 Lambda function with Map Function

map function is having two arguments, one argument is a function and the other arguments are iterator like list, tuples, sets, etc. The map function has one or more iterators. It returns the modified values to the resultant variable.

Syntax:

```
map(lambda_function,iterator)
```

Example 46: Program to multiple every element of list using map function

```
int_list1=[24,26,32,18,10,75]
result_multiply_3 = map(lambda var_int: var_int*3, int_list1)
print(type(result_multiply_3))
print(list(result_multiply_3))

#Output
<class 'map'>
[72, 78, 96, 54, 30, 225]
```

Example 47: Map function with lists (swapcase converts lower to upper and vice-versa)

```
str_list1=['take InjecTion1','Followed with Injection2','wEAr MasK','FoLLOW RULES of social distancing']

result_swapcase_alphabets = map(lambda var_chr: str.swapcase(var_chr), str_list1)
print(list(result_swapcase_alphabets))
```

#Output
['TAKE iNJECtION1', 'fOLLOWED WITH iNJECTION2', 'WeaR mASk', 'fOllow rules OF SO CIAL DISTANCING']

4.2.3.3 Lambda with Reduce Function

The reduce function belongs to the functools module and this function is having two arguments, one argument is lambda function and the other argument is iterator like list, tuple, sets, etc. This function performs repetitive operation over the iterable elements in pairs and the new reduced result is returned.

Syntax:
reduce(lambda_function, iterator)

Example 48: Factorial of a number using reduce function

```
from functools import reduce
int_list1 = [1,2,3,4,5,6]
print(reduce(lambda var1,var2:var1*var2, int_list1))
```

#Output
720

Explanation: Here, firstly 1 is multiplied with 2, then this result 2 is multiplied with 3, then this result 6 is multiplied with 4, then this result 24 is multiplied with 5, then this result 120 is multiplied with 6, and the final result of 720 is returned.

Example 49: Find smallest element in the list using reduce function

```
from functools import reduce
int_list1 = [-56,2,89,234,-78,-452]
print(reduce(lambda var1,var2:var1 if var1<var2 else var2, int_list1))
```

#Output
-452

4.3 SELF-CHECK QUESTIONS

1. Predict the following code output:

```
def output_func(var_int1,var_int2 = -17):
   print(var_int1,var_int2)
output_func(-89)
```

   a) -89 -17 b) -17 c) -89 d) -17 -89

2. Look at the below code and what line says?

```
def output_func(var_int1,var_int2 = -17):
   print(var_int1,var_int2)
output_func(-89)
```

a) Function definition
b) Function calling
c) Function header
d) Function tail

3. If Function does not have a return statement, it automatically returns None (True/False)
4. In functions, positional arguments must follow keyword arguments (True/False)
5. Tell the output of the following code

```
var_int1=10
def summation_with_five(var_int1):
  var_int1=var_int1+5
  return var_int1
summation_with_five(5)
print("Value is=",var_int1)
```

a) Value is= 10
b) Value is= 15
c) Value is= 20
d) Error

4.4 SUMMARY

The students will be able to make different programs with the help of functions. This module helps the students in understanding the concepts of inbuilt, user-defined and anonymous functions. Different types of arguments that are used in functions are well explained with suitable examples. Different inbuilt functions are mentioned, lambda function usage with map, filter and reduce functions are elaborated. This chapter helps the students to know about the reusability concept is important and functions play an important role in the reusability of code. The students can now proceed with advanced concepts of python that will help them in making projects.

4.5 PRACTICE QUESTIONS

1. The output of the following inbuilt function is
   A. print(round(-14.2378,2))
   B. print(type(type(type(float))))
   C. print(max([-34,78,-89,-12,12]))
2. If you are not aware about the number of arguments to be passed in function definition, then arbitrary arguments are used. (True / False)
3. Differentiate lambda function with map and reduce function
4. How *var_arguments and **kwargs are different and elaborate with an suitable program?
5. What is the code output?

```
var_int1 = 3
var_int3 = lambda var_int2: var_int2*var_int1**var_int2
print(var_int3(4))
```

a) 324 b) 36 c) 24 d) 496

6. A number of statements are included in anonymous function like lambda
   a) True b) False
7. The correct output of the code is

```
print(float('1e-003'))
```

a) 0.001 b) 0.003 c) 0.01 d) 0.03

8. What will be printed?

```
def bye_COVID():
  print('Take injections')

bye_COVID()
bye_COVID()
```

9. Use functions to construct a program for figuring out a number is prime or not.
10. Use functions to construct a program that helps in finding out the smallest of four integer numbers.

REFERENCES

[1] https://www.geeksforgeeks.org/python-int-function/
[2] https://www.techbeamers.com/python-float-function/
[3] https://www.tutorialsteacher.com/python/float-method
[4] https://www.geeksforgeeks.org/python-type-function/
[5] https://www.programiz.com/python-programming/methods/built-in/round
[6] https://levelup.gitconnected.com/5-types-of-arguments-in-python-function-definition-e0e2a2cafd29
[7] https://www.digitalocean.com/community/tutorials/how-to-use-args-and-kwargs-in-python-3
[8] https://www.geeksforgeeks.org/default-arguments-in-python/
[9] https://www.geeksforgeeks.org/python-lambda-anonymous-functions-filter-map-reduce/
[10] https://www.guru99.com/python-lambda-function.html

B.Sc.(DATA SCIENCE)

SEMESTER-I

PROBLEM SOLVING USING COMPUTERS

UNIT V: OBJECT-ORIENTED PROGRAMMING

STRUCTURE

5.5.4 Packages in Python

5.5.5 Python Packages vs Python Modules

5.5.6 Installing Python packages (pip-PyPi)

5.0 OBJECTIVES

In this chapter we address Python classes that can be used in a way similar to C structures, but also in a complete object-oriented way. In the first two subsection, we shall explore the use of classes as constructs for the benefit of readers who are not object-oriented programmers. The rest of the chapter deals with Python's OOPs. This is just a summary of Python's structures and constructs which is clear demonstration of object-oriented programming itself.

5.1 OBJECT-ORIENTED PROGRAMMING (OOP)

Object-oriented programming is a methodology in the programming sector that offers a tool for structuring programmes to package properties and its corresponding behaviors. OOP is often used by Python programmers because it makes programming more reusable and makes working with bigger programs easier. The methodology is used to bind properties and privileges in order to structure the program into individual objects. As a result, OOP finds it easy to follow the principle of "Don't Repeat Yourself" (DRY).

OOPs often benefit from the ability of users to represent data as a single relationship to evolving market issues. For e.g., if you were to build the programme representing the management structure of your employees in such a way that the classes involved represent something such as an employee and then objects are specific example or otherwise properties involving each employee. Likewise, one can think of another example by considering the software as a kind of factory mounting line such that a device part processes some raw material at each stage of the assembly line and eventually with additional functionalities transforms it into a finished product. Therefore, it is eventual that an object will include information such as raw or previously manufactured materials on a line at each point of time, and actions, such as the movement corresponding to every part on an assembly line.

Another traditional model for programming is procedural programming, which structures a programme, in that it offers sequentially a series of steps in the form of functions and code blocks to complete a task. The main takeover is that in Python OOP is not only represented by the details, but also by the general structure of the program.

5.2 BUILDING BLOCKS OF OOPS IN PYTHON

5.2.1 Defining a Class

There are many reasons why Python allows developers to define new classes. Classes tailored to a certain program would allow the application software to be created, debugged, read and maintained more intuitively and easily.

A prototype specified by the user or user-defined prototype for an object which sets out a number of attributes characterize the object belonging to the specific class. Since a class can be specified once and repeated several times, OOP programmes keep you from repeating code. The attributes include its data members and processes, which are obtained through the dot notation. Both

Python data types are classes, and Python gives you powerful tools to handle all aspects of the actions of a class. With the class keyword, you can define a class as:

```
class ClassName:
    'Optional string for documentation purposes'
    body
```

Here body is a set of Python statements, usually function definitions and variable assignments. However, the assignments or function definitions are not necessary and the defined body may only be a single line statement.

Many Python users, on the other hand, are unaware of Python's strong reliance on classes under the hood until they learn what a class is, just as we have progressed in this class so far without learning what classes are. The below code is an example of creating a null operation using pass statement while defining *Player* class such that nothing happens on the execution of the below sample code:

```
class Player:
    pass
```

By standard the class identifiers are in CapCase, i.e. the first letter is capitalized on each part word to differentiate and making it much easier for the developer to read around.

5.2.2 Object Instantiation

Only the object's definition or outline is created when we define a class. There will be no memory allocation until the object is created. True data or information is stored in the objector instance inst as:

inst = ClassName()

Likewise, after defining the class you can build a new class type by calling the class name as a function and this is generally referred to as a class instance.

One can create (we will name it *new_player*) an instance of the *Player* class and print the type of the variable *player_type* as:

Player.py

```
class Player:
      player_type=”batsman”
```

```
new_player=Player()
print(type(new_player))
```

Running the program will return the type of the variable as:

```
Output:
<class '__main__.Player'>
```

Further the value corresponding to instance of defined class can be done through the usage of dot notation as:

Player.py

```
class Player:

        #class attribute

        player_type=”batsman”

new_player=Player()
print(new_player.player_type)
```

Running the program will return the value of the variable as:

```
Output:
batsman
```

5.2.3 Invoking Methods

Classes may also have features such as methods that are only applicable to objects of that type. These functions are specified within the class and perform any behavior that is beneficial to that particular object category. Methods have two distinctions, much like functions:

- In order to make clear the relationship between the class and the method the methods are specified in a class description.
- The method invoking syntax is distinct from the function call syntax

Player.py

```
class Player:

        #instance attributes

        def __init__(self, matches, name):
            self.matches = matches
            self.name=name

        def odi_history(self, runs):
             return "{} has scored {} number of runs".format(self.name, runs)

         def t20_history(self, runs):
             return "{} has scored {} number of runs".format(self.name, runs)

        def tennis(self):
             return "{} loves playing tennis".format(self.name)

#object instantiation

new_player= Player(1987, "Rishabh Pant")

#instance methods calling

print(new_player.odi_history(687))
print(new_player.t20_history(1056))
print(new_player.tennis())
```

5.2.4 Class Variable Vs Instance Variable

Class Variable is a variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are. A class variable or instance variable that holds data associated with a class and its objects. Defined apart from any method, class variables are usually placed below the class header and before the constructor method and other methods by default.

An instance is an object that is constructed from a class and includes actual data, while a class is the blueprint. It may refer to as a variable that is defined inside a method and belongs only to the current instance of a class.

```
class Player:
```

```
        #class attribute

        player_type=”batsman”

       #instance attribute

       def __init__(self, matches, name):
            self.matches = matches
            self.name=name

#parametric instantiation of Player class

player1=Player(124, “Rohit Sharma”)
player2=Player(132, “Virat Kohli”)

# class variable access

print(“Rohit Sharma is an opening {}”.format(player1.__class__.player_type))
print(“Virat Kohli is also a {}”.format(player2.__class__.player_type))

# instance variable access

print(“{} has played {} matches”.format(player1.matches,player1.name))
print(“{} has played {} matches”.format(player2.matches,player2.name))
```

5.3 FOUR PRINCIPLES OF OOPS

5.3.1 Encapsulation

Encapsulation is performed by maintaining a private state of any object within a class. Other objects will only call a list of public functions if this state is not accessible directly instead. Via these functions the object maintains its own state and no other class can modify it unless specifically permitted. You will have to use the methods presented to communicate with the object.

5.3.2 Abstraction

Abstraction is an encapsulation expansion. Data are selected from a wider pool to only provide the corresponding information.

5.3.3 Inheritance

Inheritance is one object's ability to acquire one or more of those properties corresponding to another object. For example, a child inherits his/her parents' traits as well as behavior. Reusability is a significant advantage of inheritance. You may reuse the existing class fields and

methods. There are different types of inheritances in Java: single, multiple, multi-level, hierarchical, and hybrids.

We need a parent class and a child class to incorporate inheritance in the Python programme. Let's see how these two classes can be formed:

```
#creating parent class

class Sports:

        channel=”star productions”

        def details(self):

            print(“The channel broadcasts the live updates from sports corner”)
```

As in the above program, we have created a parent class which was similar to that of defining the normal class. So next, we create child class whose syntax is like similar to method calling where Parent Class is an argument as represented below:

```
class childClass(parentClass):
      #body of child class
```

So, the example code of inheritance where child class is able to access the data members of itself as well as the parent class is shown below:

```
# creating parent class

class Sports:

        channel=”star productions”

        def details(self):

            print(“The channel broadcasts the live updates from sports corner”)

#creating child class

class Cricket(Sports):
```

```
        def no_hrs(self):

            print("Cricket has broadcast time of 10 hours a day")

#main() method

new_chn = Cricket()
print(new_chn.details())
print(new_chn.no_hrs())
```

Types of Inheritance in Python

The five types of inheritance is supported by python namely

a) Single Inheritance

A child class inherits all the features of a parent class in a single inheritance.

```
# creating parent class

class Sports:
        channel="star productions"
        def details(self):
            print("The channel broadcasts the live updates from sports corner")

#creating child class

class Cricket(Sports):
        def no_hrs(self):
             print("Cricket has broadcast time of 10 hours a day")

#main() method

new_chn = Cricket()
print(new_chn.details())
print(new_chn.no_hrs())
```

b) Multiple Inheritance

A class in Python can be derived from one base class and this type of inheritance is referred to as multiple inheritance.

```
# creating parent class

class Sports:
        channel=”star productions”
        def details(self):
            print(“The channel broadcasts the live updates from sports corner”)

#creating child class

class News:
        def no_hrs(self):
             print(“News has broadcast time of 10 hours a day”)

class Match(Sports,News):
        def time(self):
             print(“Match has broadcast time of 7 hours a day”)


#main() method

new_chn = Match()
print(new_chn.details())
print(new_chn.no_hrs())
print(new_chn.time())
```

c) Multilevel Inheritance

We also can inherit a derivative class and the succession of such process is known as multilevel inheritance. In Python, it can be of some depth as:

```
# creating parent class

class Sports:
        channel=”star productions”
        def details(self):
             print(“The channel broadcasts the live updates from sports corner”)

          #creating child class

          class News(Sports):
                def no_hrs(self):
                     print(“Sports News has broadcast time of 10 hours a day”)

          class Cricket(News):
```

```
        def time(self):
            print("Cricket News has broadcast time of 7 hours a day")


    #main() method

    new_chn = Cricket()
    print(new_chn.details())
    print(new_chn.no_hrs())
    print(new_chn.time())
```

d) Hierarchical Inheritance

It is considered as hierarchical inheritance if there are more than one class inherited from the base class. The characteristics typical in child class are included in the base class in hierarchical inheritance.

```
# creating parent class

class Sports:
    channel="star productions"
    def details(self):
        print("The channel broadcasts the live updates from sports corner")

    #creating child class

    class News(Sports):
        def no_hrs(self):
            print("Sports News has broadcast time of 10 hours a day")

    class Match(Sports):
        def time(self):
            print("Cricket News has broadcast time of 7 hours a day")


    #main() method

    new_chn1 = News()
    new_chn2 = Match()
    print(new_chn.details())
    print(new_chn1.no_hrs())
    print(new_chn2.time())
```

e) Hybrid Inheritance

The combination of various types of above-mentioned inheritance is called hybrid inheritance.

```
# creating parent class

  class Sports:
      channel=”star productions”
      def details(self):
          print(“The channel broadcasts the live updates from sports corner”)

        #creating child class

        class News(Sports):
            def no_hrs(self):
                print(“Sports News has broadcast time of 10 hours a day”)

        class Match(Sports):
            def time(self):
                print(“Cricket News has broadcast time of 7 hours a day”)

        class TalkShow(Sports, Match):
            def time(self):
                print(“Talk Show of match has broadcast time of 7 hours a day”)


        #main() method

        new_chn1 = TalkShow()
        print(new_chn.details())
```

The super() function can be applied to in the inherited subclass of a parent class. The super function returns a temporary superclass object, allowing access to its child class through all its methods with an example as:

```
# creating parent class

class Sports:
      channel=”star productions”
      def details(self):
          print(“The channel broadcasts the live updates from sports corner”)

#creating child class
```

```
class Cricket(Sports):
        def no_hrs(self):
            Super().details()
            print("Cricket has broadcast time of 10 hours a day")

#main() method

new_chn = Cricket()
print(new_chn.no_hrs())
```

5.3.4 Polymorphism

Polymorphism has a means for us of using a class much as its parent such that combining forms are not confused. That being said, any subclass of children maintains its own functions/methods. Polymorphism is an object-oriented programming term, meaning multiple forms or different types. Polymorphism allows for one interface with the input of several data types, classes and various inputs. On such process is method overriding or function overloading is a kind of polymorphism in which a variety of methods can be declared with the same name but different parameters and with different parameters types. These strategies may play a similar or different role.

For example, the len function below takes string and list as an argument showing multi-purpose form of the function:

```
len("hello")
len([1,2,3,4,5])
```

Polymorphic Classes and Methods

However, it is well known that there are different forms of cricket i.e., ODI, T20 and Test but there are some attributes as well as methods which might remain same under such regard. Now we will employ the technique of method overloading in such scenario where function will act same as that of another class as shown below:

```
class Batsman:
        runs_scored=2000
        matches=40
        def calculate_average(self):
            return self.run_scored / self.matches

class Bowler:
```

```
        wickets_taken=50
        matches=30
        def calculate_average(self):
            return self.wickets_taken / self.matches

bat= Batsman()
bowl=Bowler()

print("Average of a batsman: ", bat.calculate_average())
print("Average of a bowler: ", bowl.calculate_average())

#Another form of writing

bat1= Batsman()
bowl1=Bowler()

for(obj in (bat1,bowl1)):
    obj.calculate_average()
```

5.4 SPECIAL METHODS IN OOPS

A set of special object methods is used in all integrated data forms. Double underscores (__) often precede the names of special methods. The interpreter triggers these methods automatically as the programme runs. For instance, a + b is mapped to an internal process, and a.__add__(b), and likewise an indexing operation for defined list a[i], is mapped to a.__getitem__(i). Each data type's behavior depends entirely on the selection of particular methods it utilizes. For example, the __new()__ method in python is called implicitly before the call of __init()__ method such that a new object is returned by __new()__ method and then initialized by __init()__ method.

```
class Sports:

    def __new__(mtd):
        print ("Here__new()__ magic method is being invoked")
        inst = object.__new__(mtd)
            return inst

    def __init__(self):
        print ("Here __init__ magic method is being invoked")
        self.name='Virat'

spr = Sports()
```

The output on the execution of above program is :

```
Here__new()__ magic method is being invoked
Here __init__ magic method is being invoked
```

Moreover, for the case of string operation, the special method __str()__ is useful. The usage for the same is represented as below:

```
j=6

print(str(j))

print(int.__str(j)__)
```

The output on the execution of above program is :

```
Output:

‘6’
‘6’
```

Below table entails the use of special methods in OOPS:

Most commonly used special methods in object-oriented python programming

Methods	**Description**
__new__(cls,args)	The method is used as an alternative method of object instantiation
__init__(self,args)	This method is to be called by the above __new__ method for the intialisation purposes
__del__(self)	Self-Destructor method

5.5 MODULES AND PACKAGES

In reality, in Python, there are three different ways of defining a module:

- Self-writing of the module itself in Python language
- A module like the re- (regular expression) module can be written in C and loaded dynamically at runtime.
- An integrated module like the module of itertools is inherently contained in the interpreter.

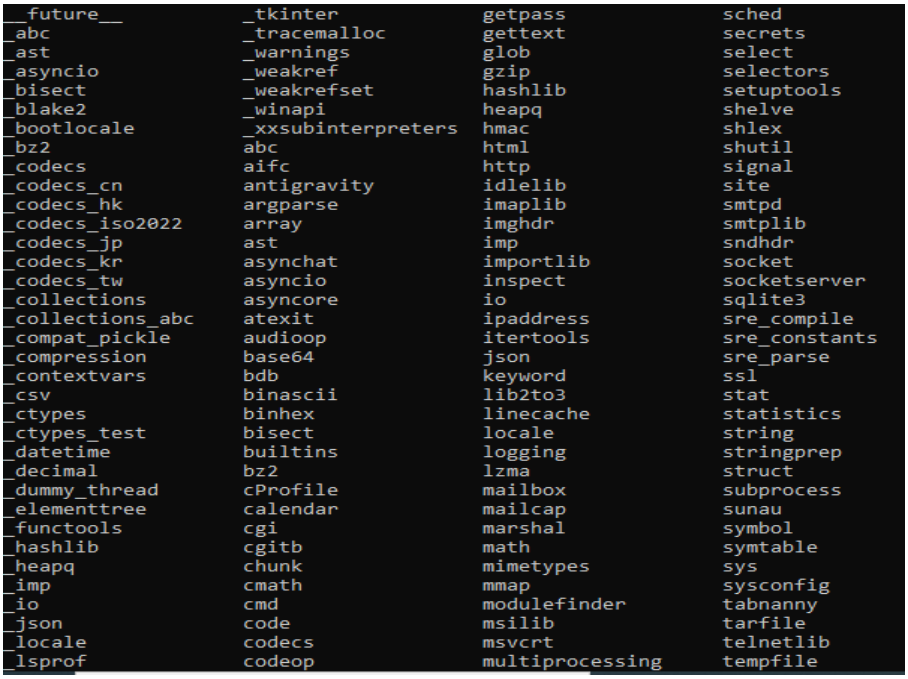
In all three cases: with the import statement, the content of a module are accessed in the same manner.

5.5.1 In-built Modules

Generally, the in-built modules are stored in the directory where the python has been installed and likewise all modules can be displayed using the following command on Python IDLE as:

```
>>> help('modules')
```

On running in the python IDLE, the output of the command is as:



The import statement as shown below is used for calling the in-built modules, such as an example of system module (sys) in this case

```
import sys
```

Likewise, the resulting search path where all these modules are being located can be assembled altogether from their respective sources of location as:

- The directory that was used for the input script or the actual directory whether the interpreter is interactively running
- The directory list in the environment variable PYTHONPATH, if it has been set. (PYTHONPATH's format depends on the OS but should be imitated as variable PATH environment.)
- An installation-based directory list set up on the installation of Python

Below python code (path_mod.py) lists the resulting search path corresponding to the sys module as:

path_mod.py

```
import sys #importing in-built module

print(sys.path)
```

The output on the execution of the code can be found as:

```
Output:

['', 'C:\\Users\\DELL\\AppData\\Local\\Programs\\Python\\Python38\\python38.zip',
'C:\\Users\\DELL\\AppData\\Local\\Programs\\Python\\Python38\\DLLs',
'C:\\Users\\DELL\\AppData\\Local\\Programs\\Python\\Python38\\lib',
'C:\\Users\\DELL\\AppData\\Local\\Programs\\Python\\Python38',
'C:\\Users\\DELL\\AppData\\Local\\Programs\\Python\\Python38\\lib\\site-packages']
```

5.5.2 User-Defined Modules

For implementing the case of self-written modules, the file (pmod.py) with extension .py needs to be created containing string (str), list of earnings (ear), self-defined method and class (with no specific operation)

pmod.py

```
#initialising string
str = “Ram is a student and does a part time job to meet his earnings”

#initialising list
l_ear = [1000,2000,3000]

#self-defined method
def savings(val):
    print(f’val={val}’)

class Test:
      pass
```

If mod.py is saved in a suitable location that you can eventually understand about importing the self-written modules by creating another file (mod_test.py) at the same location of the previous one.

mod_test.py

```
import pmod

print(pmod.str)

print(pmod.l_ear)

print(savings([‘10000’,’20000’,’24000’]))

x= pmod.Test()

print(x)
```

Once a module is imported, the location of its existence can be determined using the __file__ attribute as:

mod_exs.py

```
import re

print(re.__file__)
```

The output on the execution of the code can be found as:

```
Output:

'C:\\Users\\DELL\\AppData\\Local\\Programs\\Python\\Python38\\lib\\re.py'
```

5.5.3 Alternative form of Import Statement

An alternative type of the import statement may be imported directly to the caller's symbol table by individual artefacts of the module with syntax as:

```
from <module-name> import <name(s)>
```

In such cases for the implementation of the same we can invoke the file pmod.py (used in 1.5.2) as a module and create another python code file (anthr_import.py) for implementing the various methods of using such an alternative form of import statement as:

anthr_import.py

```
from pmod import str, savings #module as defined in section 1.5.2

print(str)

print(savings ([‘10000’,’20000’,’24000’]))
```

Even the class being defined in the self-written module can be imported using this alternative form of import statement as:

anthr_import.py

```
from pmod import Test #module as defined in section 1.5.2

x=Test()

print(x)
```

Furthermore, if user wishes to import the name of all objects being used in the specific module, then he/she can opt for using * operation after import keyword in the aforementioned alternative form of import statement as:

anthr_import.py

```
from pmod import * #module as defined in section 1.5.2

print(str)

print(savings (['10000','20000','24000']))

x=Test()

print(x)
```

It is also possible to import the name of any module using user-defined alternative name. This is done with an objective of avoiding any conflicts with previously defined names. The import statement in such cases can be represented as:

```
from <module-name> import <name> as <alt-name>[, <name> as <alt-name> ….]
```

The entire module can also be imported using alternative name such that:

alt_mod.py

```
import pmod as ud_module

print(ud_module.str)

print(ud_module.savings (['10000','20000','24000']))
```

5.5.4 Packages in Python

Essentially, a package is like a directory containing subpacks and modules. We can also use one of the Python Package Index (PyPI) for our own projects when creating our own packages. Suppose below is the directory structure for the user-defined modules or subpacks to be used

```
cricket
|-- batsman
|   |-- run.py
|   |-- __init__.py
|   |-- matches.py
|   `-- mom.py
|-- bowler
|   |-- wickets.py
|   |-- __init__.py
|   |-- average.py
|   `-- economy.py
```

A package in python must include the __init .py file, though this file may be an empty file. However, only the immediate components are shipped while we import a package, not the sub packages. It will raise an AttributeError if you want to access them.

So as an example, we type the following code to import a package:

package_exmp1.py

```
import cricket

print(cricket)
print(cricket.batsman)
```

5.5.5 Python Packages vs Python Modules

Now that all modules and packages have been revamped, let's see how differing they are:

- A module is a Python-coded format. However, a package is like a directory containing subpackages and modules.
- The __init__.py file must be held by a package. This is not true of modules.

- We use the wildcard * to import anything from a module. But in packages, such wildcard doesn't work.

5.5.6 Installing Python packages (pip-PyPi)

We learn how to use a pip to install and handle packages for Python in this section. Pip is the default Python package manager. In the Python Standard Library, we can load additional packages with pip.

On Python versions 3.4 or higher, pip comes pre-installed. However, if there are two versions of python in the system, then there might be chance that there are two pips – one pip corresponding to Python2 version (pip) and another pip corresponding to Python3 version. However, python2 is near to the deprecation in near future such that pip alone refers to third version of Python i.e., Python3. Likewise, the following command in the console may be used to check for the existence of pip:

```
>>> pip –version
```

Pip is a programme on the command line. A pip command will be added to be used with the command prompt after launch. The standard pip syntax is:

```
pip <pip-arguments>
```

Various arguments linking to the pip command can be implemented where the examples for such cases from installation to usage to removal of any package are defined in the block below:

```
pip install pandas #command to install pandas package
```

User can install the specified version of the package as:

```
pip install pandas==1.2.4 #installing specified version of pandas

Successfully installed pandas-1.2.4
```

For rechecking purposes, the user can run for the same command as during the installation of any package as:

```
pip install pandas

Requirement already satisfied
```

To uninstall a package with PIP, enter the following command in the prompt (don't forget to enter this command with a path of Python Scripts):

```
pip uninstall pandas

Successfully uninstalled pandas-1.2.4
```

In the era of data science, one is in hurry where user doesn’t want to enter manually for every single package but wants automation in such process. Therefore, pip allows the use of the requirements file which contains all the name of python packages to be installed. For example, let us consider the requirements.txt file containing the name of packages to be installed in the system

requirements.txt

```
librosa
pandas
keras
tensorflow
```

Now user can invoke pip command and install all the packages and its corresponding dependencies using single command

```
pip install -r requirements.txt
```

Likewise, one can search for the installed packages such that all packages containing the name or similar identity as:

```
pip search pygame
```

B.Sc.(DATA SCIENCE)

SEMESTER-I

PROBLEM SOLVING USING COMPUTERS

UNIT VI: ERRORS AND EXCEPTION HANDLING

STRUCTURE

6.0 OBJECTIVES

Sometimes the programme does not work at all while running a Python programme or the programme runs but produces unexpected output or is strangely behavioral. This is when one of the syntax or runtimes or logical errors arise in the code. Exceptions in Python are errors which are automatically triggered. Exceptions can, however, be strongly triggered and managed by computer code. We will be studying in this chapter about Python's exceptions such that the error can be managed and caught easily rather than debugging the whole code again and again.

6.1 INTRODUCTION

The reason for an exception is typically outside of the programme itself. Incorrect input, an improper IO device etc. are examples for the origination of such exceptions. Since the programme ends suddenly with an exception, the system resources, e.g. files, might be compromised. The exceptions should thus be properly dealt with so that the application is not abruptly shut down.

It is crucial to understand before we grasp why exception handling and forms of built-in exceptions supported by Python is needed to understand that an error and an exception are different. Therefore, let us make an effort to grasp what errors are in Python before explaining how to deal with problems. Errors are merely code errors that can be damaging leading to loss of useful information/content available in the files/systems.

In Python, there are two kinds of errors:

6.1.1 Syntax Errors

If we have not followed the rules for the programming language when developing a programme, syntax errors are identified. These errors are also called parsing errors. The interpreter does not run the programme on a syntax error unless the mistakes have been corrected, the programme is saved and re-started. If during shell mode, a syntax error occurs, Python shows the name of the error and a little explanation of the mistake.

Errors cannot be managed whereas exceptions to Python may be dealt with at run time. An error may be a (parse) syntax error, but many kinds of exceptions may happen during performance and are not unreservedly inoperative. An error can point to significant flaws that should not be detected by a sensible programme, whereas an exception might identify circumstances for an application to attempt to capture. Errors are a sort of uncontrolled exception, and they can be irretrievably handled by a programmer like an OutOfMemoryError.

6.1.2 Exceptions

Even though an expression or statement is syntactically accurate, an error might occur during its implementation. For instance, try to open a non-existent file, zero-divide, etc. Such mistakes can disturb regular programme execution and are known as exemptions.

An exception is an object from Python that is an error. If a mistake occurs during programme execution, an exception is reported. The programmer must deal with this exception so that the programme is not abnormally terminated. Therefore, a programmer may foresee and resolve such

erroneous scenarios in the design of a programme by the inclusion of the proper code for this exception.

6.2 BUILT-IN EXCEPTIONS

In the compiler/interpreter, common exceptions are often defined. These are referred to as built-in exceptions.

The standard library in Python is a comprehensive collection of built-in exceptions which address common faults (exceptions) by giving the specified remedies for these mistakes. In the case of any included exceptions, a relevant exception handler code is called that shows the reason and the exception name raised. The programmer must take suitable measures to deal with it. Some of the common built-in exceptions in Python are discussed in the table below:

Exception	Description
SyntaxError	Raised when a Python syntax error occurs.
ValueError	Raised when an in-built data-type function has the appropriate type of argument, arguments, but incorrect values are utilized corresponding to the argument
IOError	Raised for errors linked to the operating system.
KeyboardInterrupt	Raised, generally by hitting Ctrl+c or Ctrl+z, after the user stops running the programme.
SystemExit	Raised using the in-built function of sys.exit().
ArithmeticError	Base class for all numerical calculation mistakes.
OverflowError	Raised when the maximum limit for a number type exceeds a computation.
FloatingPointError	Raised if a computation of the floating point fails.
ZeroDivisionError	For all numeric types, its value is raised when division or modulo by zero occurs.
AssertionError	If the Assert statement fails, this exception is raised.
EOFError	When the end of the file is reached and there is no input from either the raw_input() or input() function, this exception is raised.
IndexError	Raised if a sequence does not find an index.
NameError	Raised when a local or global name space does not include an identifier.
IndentationError	Raised if not correctly provided indentation.
TypeError	Raised when attempting an operation or function which is invalid for the data type given.

6.3 RAISING EXCEPTIONS

The Python interpreter raises (throws) an exception every time an error is identified in a programme. Exception managers are intended to run when there is a particular exception. Program makers may also use raise and assert statements to forcibly raise exceptions in a

programme. When there is a derogation, no more statement is performed in the current code block. An exception therefore means that the usual flow of the programme is interrupted and that section of the programme is jumped into (exclusion handling code).

6.3.1 Raise Statement

The raise keyword, on the other hand, is used to raise an exception, whereas the try and except blocks are used to handle exceptions. The syntax of raise statement is as:

```
raise [Exception [, args [, trback]]]
```

Here, in the above syntax, Exception is the exception type (e.g., NameError) and the argument is the exception value. The argument is optional; the exception is None if not provided. Otherwise, trback(traceback) is likewise an optional (and often seldom used) input, and the traceback object for the exception is utilized if provided.

Only an exception handler (or a procedure called directly or indirectly by an exception handler) can use raise without any expressions. The identical exception object that the handler got is re-raised by a simple raise command. The handler is terminated, and the exception propagation mechanism continues to look for other handlers that are appropriate. When a handler realizes that it is unable to handle an exception it gets, and the exception should continue to propagate, it is beneficial to use a raise without expressions.

Likewise, a string, a class or an object can be an exception. The classes with an argument that is a class instance are mostly exceptions raised by the Python core. It is relatively straightforward to define new exceptions and may be done as follows:

```
class SalaryRangeError(Exception):
    """Exception raised for errors in the input salary.
    """
    def __init__(self, sal, mes="Salary here is not in (2000, 10000) range"):
        self.sal = sal
        self.mes = mes
        super().__init__(self.mes)


sal = int(input("Enter the amount of salary: "))
if not 2000 < sal < 10000:
    raise SalaryRangeError(sal)
```

The output of the above code is produced as:

```
Output:

Enter the amount of salary: 12000
Traceback (most recent call last):
```

```
  File "<string>", line 13, in <module>
__main__.SalaryRangeError: Salary here is not in (2000, 10000) range
```

6.3.2 Assert Statement

The declaration of *assert* is used in Python to continue the execution if the particular condition is true. If the condition assesses False, then with the supplied error message, the exception AssertionError will be raised with syntax as follow:

```
assert condition [, ErrorMessage]
```

In Python an expression in the programme code is tested using a declaration. If the test result is wrong, the exception is raised. This statement is usually used to validate the correct entry at the beginning of the function or after a function call

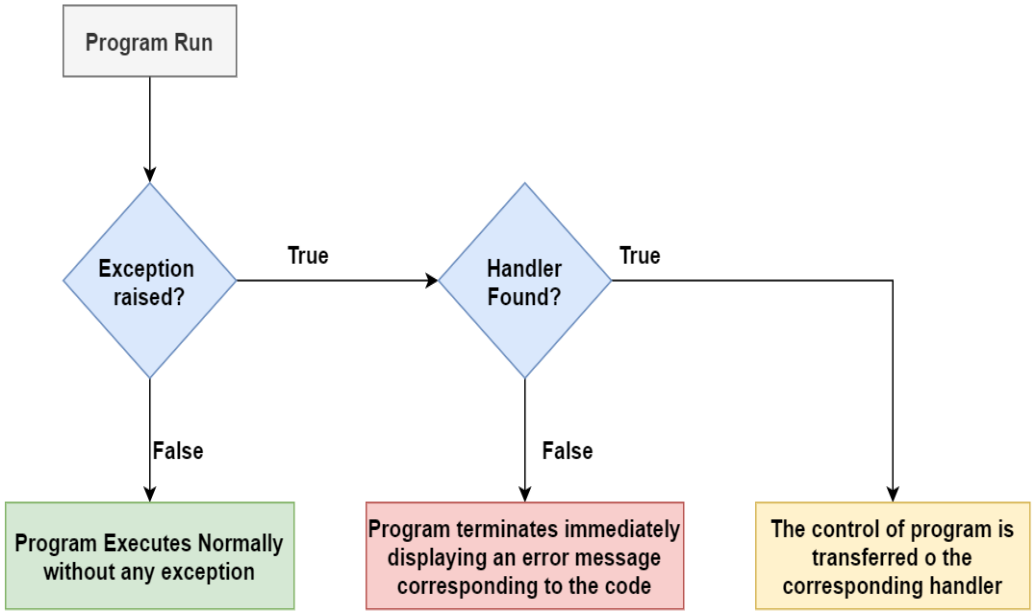
```
def discount_offer(pr, dis):
    new_pr = int(pr['price'] * (1.0 - dis))
    assert 0 <= new_pr <= pr['price']
    return new_pr

clothes = {'name': 'Moda Clothes', 'price': 14000}

print(discount_offer(clothes,0.35)) #35% discount

discount_offer(clothes,2.00) #200% discount
```

As you can see, attempting an invalid discount generates an exception to AssertionError, which links to the violated assumption. If one of these mistakes is ever encountered during the testing of our online shop, it is easy to find out by looking at the trace with output as shown below:

```
Output:

9100
Traceback (most recent call last):
  File "<string>", line 12, in <module>
  File "<string>", line 3, in discount_offer
AssertionError
```

6.4 HANDLING EXCEPTIONS

The flow chart for the process of handling exception in python programming language is as follows:



You may safeguard your programme by placing the suspended code in a block if you have a questionable code which might produce an exception, it is recommended to include a statement after the try: block, followed by an elegant bit of code to deal with the situation.

An exceptional except for a class that is the same class or the same base class (but not the other way round – an exclusively clause listing a derived class is not consistent with a basic class) is compatible with an except for a clause. For instance, in this order, the following code will display B, C, D:

```
class Sports(Exception):
    pass

class Cricket(Sports):
    pass

class News(Cricket):
    pass

for cls in [Sports, Cricket, News]:
    try:
        raise cls()
    except News:
        print("This is news section of cricket")
    except Cricket:
```

```
        print("This is section of cricket game")
    except Sports:
        print("This is section of sports center")
```

On running the above code, the following output is being obtained:

```
Output:

This is section of sports center
This is section of cricket game
This is news section of cricket
```

There may be several test statements except statements for one attempt. The test block contains statements that might produce different kinds of exceptions. This is considered as useful for the developer's prospective such that a general excluding clause should cover any exception that may also be provided. Moreover, one can put a different clause after such exception(s). The code in the other block executes if the block tries: no exception is raised. Therefore, the use of other block is a suitable area for code that needs little effort, thus ensuring protection of blocks while coding for bigger enterprise applications. Below is an example of exception handling in file systems:

```
#Exception handling alongside the use of file handling

try:
   f_obj = open("newfile.txt", "w") #opening file in writing mode
   f_obj.write("This is a check for exception handling!!") #writing file
except IOError:
   print "Error: Unable to find file or read data"
else:
   print "Successfully written content into the file"
   f_obj.close()
```

On running the above code, the following output is being produced:

```
Output:

Successfully written content into the file
```

However, exception handling was not useful in the above case as legitimate conditions for opening the file were met as well as adequate. However, when we specify the wrong mode and

then start writing the file, then the try block executes to an exception. The illustration of such case in file handling is shown below:

```
#exception handling alongside the use of file handling

try:
   f_obj = open("newfile.txt", "w") #opening file in writing mode
   f_obj.write("This is a check for exception handling!!") #writing file
except IOError:
   print "Error: Unable to find file or read data"
else:
   print "Successfully written content into the file"
   f_obj.close()
```

On running the above code, the following output is being produced:

```
Output:

Error: Unable to find file or read data
```

The try ... except the statement contains else optional clause, which must follow everything except the provisions when present. It is advantageous if a trial clause does not trigger an exception for code which must be performed. For instance, the below code details the use of exception handling in such scenarios:

```
for i in sys.argv[1:]:
    try:
        f = open(i, 'r')
    except OSError:
        print('cannot open the file', arg)
    else:
        print(i, 'has', len(f.readlines()), ' number of lines')
        f.close()
```

Instead of adding more code to the test clause, the usage of the other clause is better, since it avoids mistakenly catching an exception not produced by the code that is shielded from the test unless statement. If an exception occurs, a value, also known as an argument for the exception, may occur. The type and existence of the argument will depend on the kind of exception. The exception clause can specify an exception name variable. The variable is linked with the argument specified in instance.args to an exception instance.

6.5 USING PYLINT IN PYTHON

The knowledge of the other programmes is one of the main issues of the day. Even worse, situations when there is no informative stuff in the code, such as comments or docstrings. As a programmer, our code should be legible and comprehensible. This is the tale of a Pylint tool that discovered a production-impacting problem the day before the code was deployed. This is also the storey of a tool that, for good reason, no one uses. By the end of this section, one can easily understand why this tool is valuable, why it isn't, and how to utilize it with your Python project.

Pylint is a library for third parties which is not in Python by default and is quite simple to set up. Since Pylint is not part of the standard Python library, we have to separately install it. The pip package can simply accomplish this. Pip is Python's standard Package Manager, enabling packages not included in the Python Standard Library to be installed and managed. Simply run the install command through pip packages, and Pylint and all of its dependencies will be installed.

```
pip install pylint
```

When we have pylint installed, the command pylint with the file name is easy to use by running the following code utilizing pylint package:

```
pylint exp_file.py
```

6.5.1 Detecting Multi-statement with Implicit Continuation

In Python, inside parathesis we utilize implicit line, brackets ([]) and braces ({{}). Implicit indicates that the line continuation character (\) is not written so that a statement is extended over many lines.

The wrapped element should be vertically aligned or hanging by use of the implied continuation lines. Hanging indentation in Python indicates that the start of a parenthesized declaration is the ultimate non-whitespace character of the line, and successive lines are indented until the closing parentheses. Let’s take an example where we will see handing indentation while defining the function and its corresponding arguments:

test.py

```
#multi-line statement

def mult_func(a1, a2, a3,
  a4, a5):
    """The function performs the numerical operations on numbers"""
    return a1 * a2 - a3 * a4 + a5
```

```
mult_func(2,4,1,6,5)
```

On running the above code using the output comes out to be

```
> python3 mult_pylint.py

7
```

The code will run but as said with pylint one can discover a production-impacting problem such that the readability of code for cross-platform deployments can be tested as:

```
> pylint test.py

************** Module test
test.py:1:0: C0114: Missing module docstring (missing-module-docstring)
test.py:2:0: C0103: Argument name "a1" doesn't conform to snake_case naming style (invalid-name)
test.py:2:0: C0103: Argument name "a2" doesn't conform to snake_case naming style (invalid-name)
test.py:2:0: C0103: Argument name "a3" doesn't conform to snake_case naming style (invalid-name)
test.py:2:0: C0103: Argument name "a4" doesn't conform to snake_case naming style (invalid-name)
test.py:2:0: C0103: Argument name "a5" doesn't conform to snake_case naming style (invalid-name)
test.py:2:0: C0116: Missing function or method docstring (missing-function-docstring)

-----------------------------------
Your code has been rated at -13.33/10
```

6.5.2 Using Operators

A warning at the beginning of the code may be easily deactivated by inserting a statement (#pylint; disable=C0114)

The warning C0114 shows the absence of a module docstring. A docstring is a string in module, function, class, or method declaration which is the initial statement. Under PEP 257, all modules should include a docstring stating what the module performs in the starting point. We will not put a docstring to the top of each module in order to make things simpler. However, the practise of writing doctrines is strongly advised.

Below code shows the use of operators where python recommends the use employing single space on either side:

test1.py

```
# pylint: disable=C0114

#Comparison operators
print(9<8)

# Membership operator
if 5 in [1, 2, 3, 4, 5]:
    print('element is present in this list')
else:
    print('element is not present in this list')
```

```
> python3 test1.py
False
element is present in this list

> pylint test1.py
********** Module test1
test1.py:11:0: C0305: Trailing newlines (trailing-newlines)

--------------------------------
Your code has been rated at 7.50/10
```

6.5.3 Whitespaces following the use of comma, semicolon, or colon

The following problematic practices can be observed in the code:

- After the comma separating each item in the list, a whitespace is lacking.
- Following the colon (:), which divides the key value pair in the dictionary, there is no whitespace.
- There is a whitespace just before the comma separating each tuple element.

test2.py

```
# pylint: disable=C0103
# pylint: disable=C0114

# list
num=[1,2,3,4,5] #trailing whitespace at the end
```

```
# dictionary - players and grades
score_grade=   {'Rohit':10,   'Virat':2.5, 'Pant':8.5} #trailing whitespace at the end

# tuple - strike rate and average
perf =   (128.5 , 51.576) #trailing whitespace at the end

print(score_grade, end=",") #trailing whitespace at the end
print(perf) #trailing whitespace at the end
```

```
>python3 test2.py
{'Rohit': 10, 'Virat': 2.5, 'Pant': 8.5},(128.5, 51.576)

> pylint test2.py

************** Module test2
test2.py:5:15: C0303: Trailing whitespace (trailing-whitespace)
test2.py:8:54: C0303: Trailing whitespace (trailing-whitespace)
test2.py:11:25: C0303: Trailing whitespace (trailing-whitespace)
test2.py:13:27: C0303: Trailing whitespace (trailing-whitespace)
test2.py:14:11: C0303: Trailing whitespace (trailing-whitespace)

--------------------------------------------------------------------
Your code has been rated at 0.00/10
```

B.Sc.(DATA SCIENCE)

SEMESTER-I

PROBLEM SOLVING USING COMPUTERS

UNIT VII: PYTHON GENERATORS

STRUCTURE

7.0 OBJECTIVES

In this chapter, we study the importance of effective utilization of large sets of results files without the allocation of the memory for all the outcomes simultaneously. Likewise, this can be achieved in Python through providing our own iterator method by employing a generator. A generator is a specific function type that does not return a single value but rather returns a stream of values for an iterator object or in instances when the generator utilizes or consumes another generator, and when it is done as early as feasible, it is more convenient. A yield statement is used instead of a return statement in a generator function. A basic generating feature and functions are further detailed in this chapter

7.1 Generators in Python Programming

Have you ever had to read big datasets or files in a circumstance that was too overwhelming to put into memory? Or perhaps you intended to make an iterator but the manufacture was simple enough to create the iterator, as opposed to creating the needed values?

Keeping in mind the growing outrage of data science in normal life scenarios, the effective use of generator under some of these situations can be quite helpful and simple. The prospects of positivity are generalized on both the sides whether the person is developer or the reader of beautifully written code.

The generator functions introduced with PEP 255 are a specific type of function which returns some form of lazy iterator. Objects can be looped over like a list, although lazy iterators do not save its contents in the memory, unlike lists. The quantity of code required for code is one of the advantages of employing generator functions for using iterators.

7.1.1 Using yield keyword in Python

The keyword yield functions as similar to that of yield in python where the only difference is that it returns a generator object to the caller instead of returning a value.

The function execution stops at the line itself when a function is called and the running thread discovers an output keyword in the function, and returns a generator object to the caller.

If the statement begins an iteration over a collection of items, the generator is running. When the function code of the generator reaches a statement "render," the generator returns its execution to the loop, returning a new value from the set. The generator function can produce as many (maybe unlimited) values as it wishes, each of which in turn results.

Let's see some instances of generators and yield in action after this introduction:

test_yield.py

```
def yield_func():
      yield “This is the way of using yield similar to that of return function”

print(yield_func())
```

Above was the basic syntax covering yield keyword where on running the program, the corresponding output obtained is as following:

Output:

```
<generator object yield_func at 0x7fb3f0ea6510>
```

The essential points considering the usage of yield are as follows:

- A yield produces a function exit but we start 'where we have left off' next time the function is called, i.e., on the line after the output rather than at the starting of the function.
- All local variables values that existed at the time of the yield action are kept intact at the time of resuming.
- The same generator can have many yield lines.
- There are also return statements, however if a StopIteration exception is generated if the next() function is called once again, the execution of a such statement will occur.
- Returns one argument to yield (or none). That can be a tuple parameter, however.

7.1.2 Using Generator in Python

As of now, we are very familiar with syntax employing yield keyword in python. Therefore, the next task is to fetch the corresponding values from generator object. One has to remember that these generator objects are able to be fetched one at a time instead of the whole list together. So for carrying out operations where the whole list is required to be fetched, we can use loop, next() or preferably the list() method.

Following are the examples of the generators corresponding to their fetched values in each scenario

test_generator.py

```
def generator_func():
      yield “first statement”
      yield “second statement”
```

```
    yield “third statement”
    yield “fourth statement”
    yield “fifth statement”

gen = generator_func()

print(gen)

for i in gen:
    print(i)
```

On running the above program, one can clearly see the use of for loop to get the values stored at particular address corresponding to that generator. Thus, the output of the above code is as:

```
Output:

<generator object generator_func at 0x7f2fdb56d510>
first statement
second statement
third statement
fourth statement
fifth statement
```

7.1.3 Difference of normal function and generator function

The use of generator function sounds similar to that of normal function employed in python programming. Therefore, with an example below, we try to investigate the difference between the two such that the sample code ought to return only the value back which is the form of string.

```
#Generator Function

def gen_func():
     yield “This is the way of using yield in generator function”

#Normal Function

def nor_func():
    return “This is the way of using return in normal function”

print(gen_func()) #calling generator function

print(nor_func()) #calling normal function
```

On running the program, the output will clearly define the difference of using yield and return statement such that the yield keyword corresponds to the address instead normal function returns the string.

```
Output:

<generator object gen_func at 0x7f2fdb56d510>
This is the way of using return in normal function
```

7.2 USING YIELD FROM IN PYTHON GENERATOR

Let's first get out of the way one item. The rationale that the yield of *val* equals *for val in g* is because yield v is not even eq-ual to what yield is. Well let's face it, if the output of the entire loop expands, then the adding of the output of the language does not merit the addition of all new features in a Python 2.x.

What yield from keyword used in generator does is that it sets up a transparent two-way link between the caller and the sub-generator such that:

- The link is "transparent" in that it also spreads everything appropriately, not just the pieces that are created (e.g. exceptions are propagated).
- The link is "bidirectional" because data may be transmitted from and to a generator.

Let’s illustrate an example where does the actual role of yield from keywords originate and how will it help in solving the rationale problem equivalent to that of for loop. In the below code, we designate the use of manual iteration over read_value() function as:

```
def read_value():
    for i in range(4):
        yield '<< %s' % i

def read_wrap(g):
    # Manually iterate over data produced by reader
    for v in g:
        yield v

wrap = read_wrap(read_value())
for i in wrap:
    print(i)
```

Likewise, what recommened here is to use yield from keyword in read_value() function instead of iterating manually over the function. Therefore, below changes in the code clearly depicts the role of yield from keyword and moreover, the readability of code increases through elimination of one line of code as:

```
def read_value():
    yield from g

def read_wrap(g):
    # Manually iterate over data produced by reader
    for v in g:
        yield v

wrap = read_wrapper(read_value())
for i in wrap:
    print(i)
```

7.3 REAL-LIFE USE CASES OF PYTHON

Generators often work with large files or data streams, such as CSV files considering real-life applications. For implementing such scenario of handling larger files, the code assumes how many rows on a text file we have to count for which the python program may look like:

csv_read.py

```
def read_file (name):
    f = open(name)
    res = f.read().split("\n")
    return res

#csv_reader for reading large text file
gen_csv = read_file("largefile.txt")

#intializing count of the rows
count = 0

for i in gen_csv:
    count += 1

print(f"The number of rows in document are {count}")
```

The above code will probably work on any modern machine if the file contains a few thousand lines, but if the file is large enough, then we will have a few problems. The issues can start to slow down from the machine, until the programme kills the machine, so that the programme must be terminated, to the end.

We supplied large number of files with thousand number of rows and we had to stop the code manually. So due to long-time processing the code eventually on manual trigger resulted in:

```
Output:

Traceback (most recent call last):
Memory Error
```

So we made use of generator in such scenarios and modified the above written code csv_read.py as:

csv_read_gen.py

```
def read_file (name):
    for i in open(file_name, “r”):
        yield i

#csv_reader for reading large text file
gen_csv = read_file("largefile.txt")

#intializing count of the rows
count = 0

for i in gen_csv:
    count += 1

print(f"The number of rows in the given file are {count}")
```

So, after using the yield in the given function read_file(name) in csv_read_gen.py file, the output of the code yielded as:

```
Output:

The number of rows in the given file are 55182343
```

But that's not the end of a tale, there's even easier and more fascinating ways of implementing a generator expression (also known as a generator comprehension) which has a syntax that makes it look like list comprehension.

```
gen_csv = (i for i in open("largefile.txt"))
```

7.4 MAKING AN ITERABLE FROM A GENERATOR

Even although the object-oriented technique to create an iterator is really fascinating, it is not a computationally efficient approach. A generator function is the most common and easiest way to build a iterator in Python. So in order to discuss such a method will implement iterator through generator i.e. an effort will be made to make an iterable from generator

As we had had already studied for the use of for loop while getting the value of generator object, the next() function can also be employed to get the value in this regard. When a generator function is invoked, a generator object is returned without even starting the function. When the next method is initially invoked, the function begins to execute until the return statement is reached. The value received is returned by the next call.

```
#Generator Function

def gen_func():
    yield "This is the way of using yield in generator function"

print(next(gen_func))
```

On running the above program, the next() makes the generator function to return the value instead of the address with output as shown below:

```
Output:

This is the way of using yield in generator function
```

First of all, we would be looking for the process where we will implement an iterator as a Class. An iterable is an object in Python that defines an iterator or an index (index) using __init__ or the __next__ method. In brief, every object that can provide us an iterator may be iterable. What's an iterator then? Such a way can be used to cycle over an iterable object forever through the implementation as:

```
class Sports(str):

    def __init__(self, itrb):
        self.itrb = itrb
```

```
            self.obj_iter = iter(iter)

        def __iter__(self):
            return self

        def __next__(self):
            while True:
             try:
                  obj_next = next(self.obj_iter)
                  return obj_next
              except StopIteration:
                   self.obj_iter = iter(self.itrb)

obj=Sports("Cricket")

print(obj)

for i in range(20):
      print(next(obj), end = ", ")
```

7.5 RECURSIVE GENERATOR

We know in Python that one function can call another function can call yet at the same time, the function can even be called by itself. These building types are called recursive functions.

Now you need to ask if recursion in Python generators may be used?

The below is the code demonstrating the use of recursive generator function. The code aims to print the even numbers in the series till 20 *(num<20)* through recursive calling of generator function *even_num(arg)*as detailed below:

recur_gen.py

```
#using recursive generator function

def even_num(begin):
   yield begin
   yield from even_num(begin+2)

#using loop for printing even numbers till 20 from 1
for num in even_num(2):
   if num<20:
     print (num)
```

```
  else:
    break
```

```
Output:

2
4
6
8
10
12
14
16
18
```

7.6 GENERATOR EXPRESSIONS

Using generator expressions, simple generators can be created on fly easily. It facilitates the construction of generators. Similar to lambda functions, generator expressions create anonymous generator functions.

The syntax is similar to the Python list comprehension for the generator expression. But round brackets replace the square brackets. The biggest difference between a list understanding and a generator statement is that a list understanding produces the entire list and the generator statement produces one item at a time. Below is example of it:

```
iterator = ('Iterator' for i in range(5))
```

The above-mentioned generator expression produces the sequence of values that we developed in my generator lesson when it was iterated. Again, here, your memory is refreshed:

```
def repeating_val(val, count):
   for i in range(count):
      yield val

iterator = repeating_val('Iterator', 5)
```

Generator Expressions vs List Comprehensions

As you might say, generator terms are pretty comparable to list comprehensions:

```
list_comp = ['Iterator' for i in range(5)]

gen_expr = ('Iterator' for i in range(5))

print(list_comp)

print(gen_expr)
```

However, generator expressions do not build list objects, as opposed to list understandings. Rather, they create "on time" data, such as a class-based iterator or generator function. All you obtain is an iterable "generator object" by assigning a generator expression to a variable with output of above program as:

```
Output:

['Iterator', 'Iterator', 'Iterator', 'Iterator', 'Iterator']
<generator object <genexpr> at 0x7f2af112b580>
```

You must call next() in the same way as you want in any other iterator to get the values generated by the generator expression:

```
gen_expr = ('Iterator' for i in range(5))

print(next(gen_expr))
print(next(gen_expr))
print(next(gen_expr))
print(next(gen_expr))
print(next(gen_expr))
```

```
Output:

Iterator
Iterator
Iterator
Iterator
Iterator
```

Alternately, in a generator expression, you can also invoke the list() method to build a list object that contains all the values generated:

```
gen_expr = ('Iterator' for i in range(5))

print(list(gen_expr))
```

```
Output:
['Iterator', 'Iterator', 'Iterator', 'Iterator', 'Iterator']
```

In-line Generator Expressions

As generator expressions are, well... expressions, you may utilize them in-line with additional statements. You may for example define and use an iterator with a for-loop immediately:

```
for x in (‘Iterator’ for i in range(5)):
    print(x)
```

```
Output:

Iterator
Iterator
Iterator
Iterator
Iterator
```

7.7 SUMMARY

Generators allow you, in a pythonic way, to create iterators. Iterators only generate the next element of a requested iterable object, and allow lazy evaluation. For really big data sets, this is beneficial. Only over one time, Iterators and generators can be iterated over the inputs. It's better than iterators to employ generator functions. The expression of generators is better than iterators (for simple cases only). Generator expressions are comparable to list comprehensions. They don't build list objects, though. Generator expressions instead create 'time-only' values such as a class-based iterator or generator function. It cannot be restarted or reused after a generating expression is spent. For implementing basic adhoc iterators, generator expressions are optimal. It is best to create a generator or a class-based iterator for complicated iterators.

B.Sc.(DATA SCIENCE)

SEMESTER-I

PROBLEM SOLVING USING COMPUTERS

UNIT VIII: FILE HANDLING

STRUCTURE

8.0 OBJECTIVES

Python offers us a key functionality to read file data and write data to a file. So, in this chapter, the objective is to study the important aspects of using various kinds of files and their methods. However, all values or data in programming languages are saved in certain volatile variables. Because data is only saved in such variables during runtime and is lost once the running of the program is over. It is therefore best to save these data with files permanently.

8.1 Introduction to File Handling

So far, we've written Python programmes that receive input, alter it, and show the results. However, that output is only available while the application is running, and input must be supplied using the keyboard. This is due to the fact that the variables used in a programme have a lifespan that lasts until the programme is executed. What if we wanted to save the data that was entered as well as the generated output indefinitely so that we could utilise it again later? Typically, businesses would wish to save information on personnel, inventory, sales, and other items indefinitely to prevent having to enter the same information over and over again. As a result, data is permanently preserved on secondary storage devices for reusability. With a .py extension, we save Python programmes produced in script mode. Each programme is saved as a file on the secondary device. Similarly, the data input and the result can be saved to a file indefinitely.

Files are identified locations on disc where associated data is stored. They're used to keep data in a non-volatile memory for a long time (e.g., hard disk). We utilize files for future usage of the data by permanently saving it since Random Access Memory (RAM) is volatile (it loses its contents when the machine is switched off). We must first open a file before we can read from or write to it. When we're finished, it has to be closed so that the file's resources may be released. As a result, a file operation in Python is performed in the following order:

- Opening a file
- Reading or writing
- Closing the file

8.2 TYPES OF FILES AND FORMATS

Every file on a computer is stored as just a series of 0s and 1s, or in binary form, as we all know. As a result, each file is really nothing more than a sequence of bytes saved one after the other. Text files and binary files are the two most common forms of data files. Any text editor can open a text file, which is made up of human readable characters. Binary files, on the other hand, are made up of non-human readable letters and symbols that must be accessed using special tools.

8.2.1 Text Files

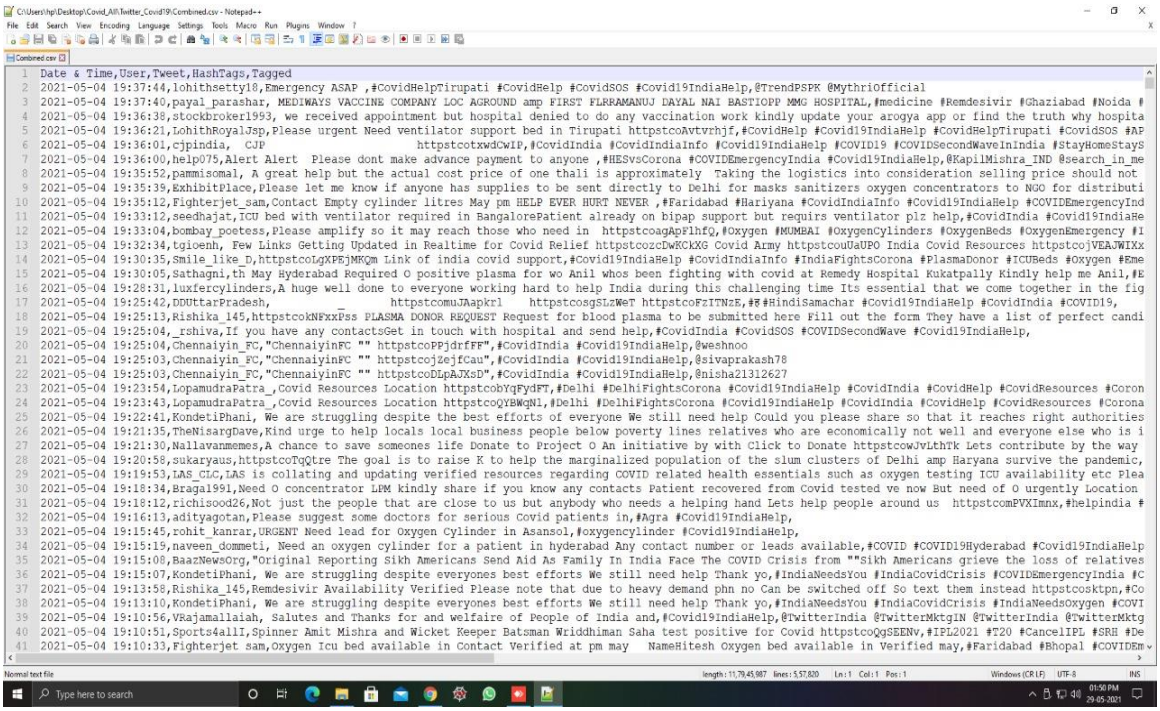
A text file is a sequence of characters that encompasses alphabets, numerals, and other special symbols. Text files include files with extensions such as.txt,.py,.csv, and others. We see many lines of text when we open a text file in a text editor (e.g., Notepad). Internally, however, the file

contents are not saved in this manner. Rather, they are saved as a series of 0s and 1s in a byte sequence. The value of each character in a text file is recorded as bytes in ASCII, UNICODE, or any other encoding system. As a result, when we open a text file, the text editor converts each ASCII value and displays the human-readable comparable character.

Comma Separated Files

A CSV file has some data as a text file. A CSV file is usually used for transferring data across applications. A CSV file holds data, including numbers and text in a simple form, for clarification. The plain text compresses and enables text formatting, as you would recall.

Typically, when there are a bunch of data that is to be transmitted to another programme, an extension of CSV is employed. The file extension, however, assists an operating system to determine which software the file is connected with, in particular. The usage of a spreadsheet application can also better suit the user's needs, as it features cells in which data is arranged in rows and columns.

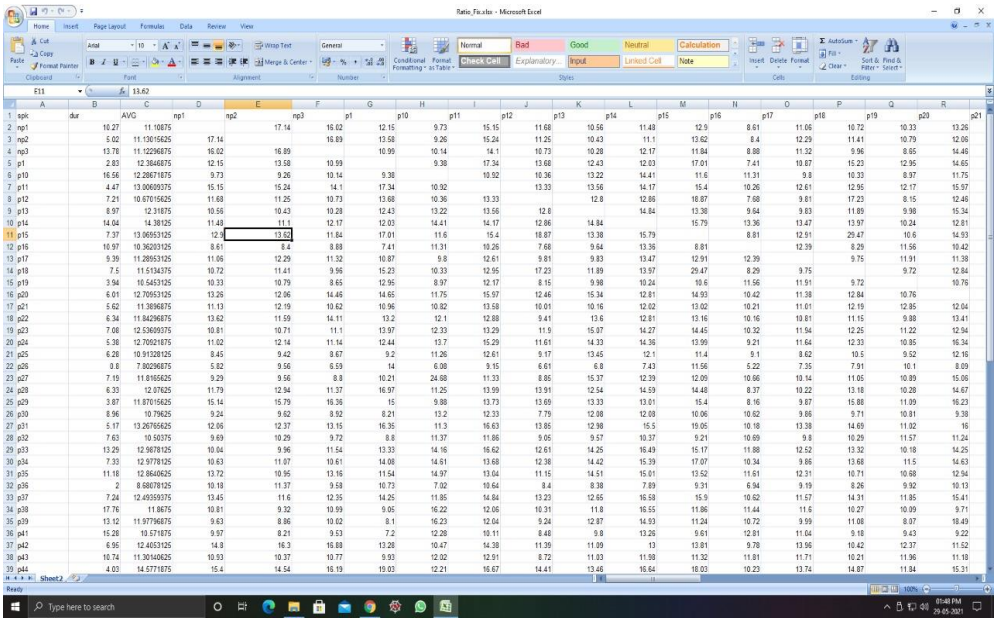


XLSX Files

A Microsoft Excel Open XML Format Spreadsheet file is an XLSX file extension. It's a ZIP Compressed, Microsoft Excel 2007 and later XML-based spreadsheet file.

XLSX files manage data in cells stored in worksheets and saved in workbooks (files that contain multiple worksheets). The cells in a table are positioned according to rows and columns and are capable of including designs, formatting, math and more.

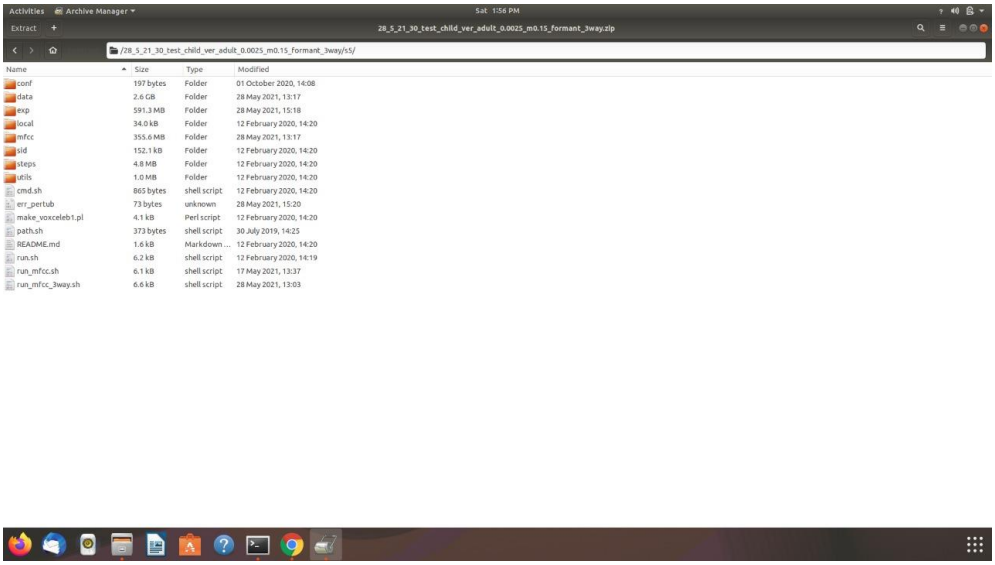
Files produced in previous Excel versions are kept in XLS format. XLSM files are Excel supporting macros.



Zip Files

A zip file is a means to gather many files or archive them in order to function as one file. Let's imagine for instance you would like to send someone with a folder containing Word documents. A zip file is a means to gather many files or archive them in order to function as one file. It would take a long time to attach each file, especially if a large number of papers exist. It might be best to put all the files in a zip file then attach your e-mail to the zip file. Software downloads are the most frequent usage of ZIP files. Zipping software saves the server space, reduces the time required for your computer download and preserves hundreds or thousands of files in a simple ZIP file well structured.

Let's imagine for instance you would like to send someone with a folder containing Word documents. It would take a long time to attach each file, especially if a large number of papers exist. It might be best to put all the files in a zip file then attach your e-mail to the zip file. ZIP file is one of the most frequently used archive formats in which you run, as a ZIP file extension. A file ZIP is merely a collection of one or more files and/or directories, but a single file is compressed for simple transmission and compression like other file formats.

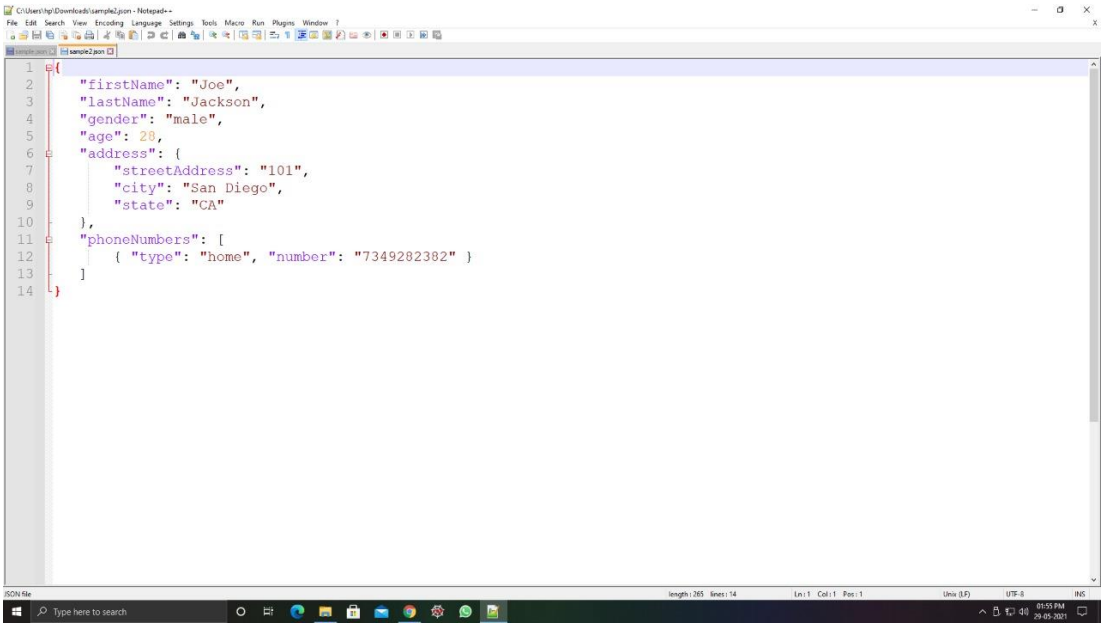


JSON Files

A JSON file is a file that holds basic data structures and its corresponding objects in mostly utilized JavaScript Object Notation (JSON), a standard format for data exchange. It is mostly used for data transmission between a web application and a server. JSON files are small, text-based, and readable by people and can also be changed with a text editor.

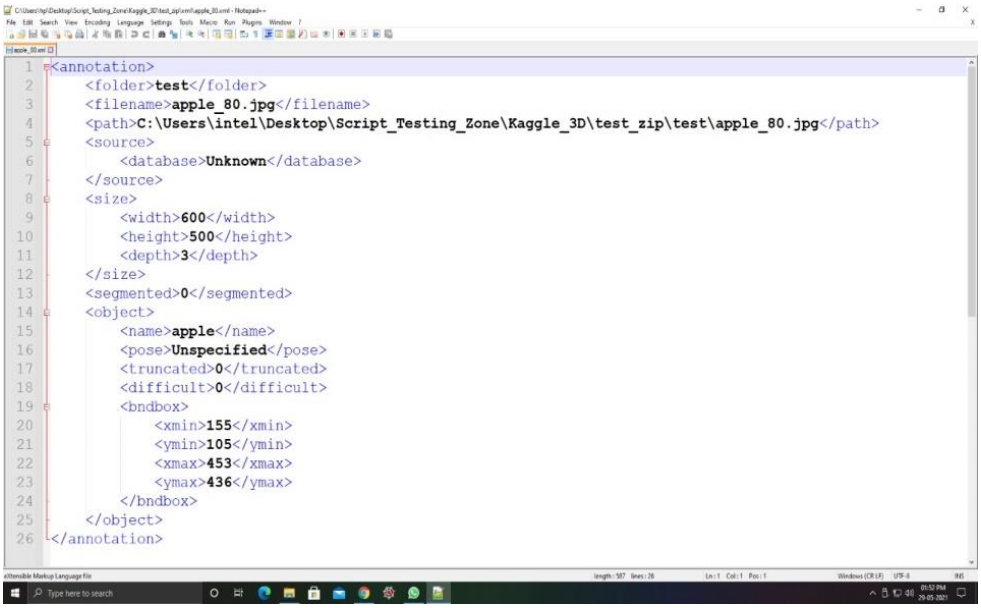
While many apps employ JSON for data exchange, files with the .json cannot effectively be saved on local disk, as there is data exchange between linked machines. But users can save .json files in some programmes. One example is Google+, where JSON files are used to store profile data. You may pick "Data liberation" after login and pick "Download Profile Data."

An XML file for storage and transmission is an extensible marking language file. Tags and text are available in an XML file. The tags supply the data structure. The text in the file you want to keep is enclosed by the tags that comply with certain syntactic requirements. At the heart of an XML file is a conventional text file with specific tags to specify the document's structure, storage and transmission



XML Files

XML is a markup language, which implies that it is a computer language employing tags to specify file components. Instead of writing a syntax this markup language incorporates real words. Likewise, HTML and XML are the most common markup languages.



8.2.2 Binary Files

Binary files, like text files, are contained in bytes (0s and 1s), however these bytes do not reflect the ASCII values of characters. Instead, they represent actual content like images, audio, video, compressed copies of other files, executable files, and so on. These files are not readable by

humans. As a result, using a text editor to access a binary file will result in some junk values. To read or write the contents of a binary file, we'll need specialized software.

Furthermore, binary formats have advantages in terms of access speed. While the underlying unit of information in a plain text file is simple (one byte = one character), locating the actual data values is frequently more difficult. To discover the third data value on the tenth row of a CSV file, for example, the reader programme must continue reading bytes until nine end-of-line characters are detected, followed by two delimiter characters. This implies that in order to discover a certain value in a text file, you must generally read the entire file.

To locate the position (and meaning) of any value in a binary format, some type of format description, or map, is necessary. However, having such a map has the advantage of allowing any value inside the file to be retrieved without bothering to read the entire file.

8.3 OPENING FILES IN PYTHON

To open a file in Python, the built-in open() function is being used. This method returns a file object (commonly known as a handle) that may be used to read or change a file. The syntax of how to open file in python using file object (file_obj) is shown below:

```
file_obj = open(“file”,”mode”)
```

Here file corresponds to the name of the corresponding file that needs to opened or on which any operations need to be performed. Likewise, mode represents the attribute telling the mode in which the actual file needs to be open. When we open a file, we may define the mode. We specify whether we want to read (r), write (w), or append (a) to the file with the mode parameter. We may additionally indicate whether the file should be opened in text or binary form.

For example, one would like to read the contents of the already existing file in your system, then first of all we need to create the existing exmpl.txt as shown below.

exmpl.txt

```
Hello, this is an example of just reading the file using the default mode which is r mode.

To accomplish this, consistent grammar, pronunciation, and more common terms would be required.

When many languages merge, the new language's grammar is more basic and regular than the separate languages.

The new common language will be more straightforward and consistent than current European languages.
```

Now, the task is to read the file such that we assume that the location of the file is similar to that where the python code is being saved, otherwise instead of file name, we need to provide the absolute or relative path in order to open the file. Likewise, for reading the contents inside the file, we opt to use in-built python's read() function as shown in file_open.py below:

file_open.py

```
file_obj = open("exmpl.txt", "r")

'''
or we may also write file_obj=open("exmpl.text")
as by default it open() function opens file in read mode
'''

print(file_obj.read()) #in-built read() function
```

On running the above program, the contents present in the file are displayed as:

Output:

```
Hello, this is an example of just reading the file using the default mode which is r mode.

To accomplish this, consistent grammar, pronunciation, and more common terms would be required.

When many languages merge, the new language's grammar is more basic and regular than the separate languages.

The new common language will be more straightforward and consistent than current European languages.
```

The read() function employed above provides the entire text stored in the file by default, but user can also specify the number of characters he/she wishes to return:

```
file_obj= open("exmpl.txt", "r")

print(file_obj.read(5))
```

The output of the above code will be as:

```
Output:

Hello
```

Likewise, there are other alternative ways of reading the file including using readline() function which returns single line in the file and secondly using for loop through file line by line. Both the functionalities are being represented on the same plain text file () as we had used earlier:

file_readline.py

```
file_obj = open("exmpl.txt","r")

print(file_obj.readline()) #print first line of file
print(file_obj.readline()) #print second line of file
```

```
Output:

Hello, this is an example of just reading the file using the default mode which is r mode.

To accomplish this, consistent grammar, pronunciation, and more common terms would be required.
```

file_loop.py

```
file_obj = open("exmpl.txt","r")

for line in file_obj:
    print(line)
```

```
Output:

Hello, this is an example of just reading the file using the default mode which is r mode.

To accomplish this, consistent grammar, pronunciation, and more common terms would be required.
```

When many languages merge, the new language's grammar is more basic and regular than the separate languages.

The new common language will be more straightforward and consistent than current European languages.

readline() vs readlines() in Python

Let’s assume the inbuilt text file, listdata.txt, with the contents as shown below:

listdata.txt

```
Rohit
Virat
Rishabh
Hardik
```

readline() reads a file line till the end of that line is reached. In the string is maintained a trailing newline character (\n). Likewise, readlines() in contrast produces a list with all the lines of the file (strings). The syntax is this:

file_readlines.py

```
f ile_obj = open("listdata.txt")

print(file_obj.readlines()) #readlines() return list of files
```

The output of running the file is shown as below:

Output:

```
[‘Rohit\n’,’Virat\n’,’Rishabh\n’,’Hardik\n’]
```

8.4 MODES OF OPENING FILE IN PYTHON

It's a number that indicates the file's opening mode, such as read, write, append, and so on. It's a non-mandatory parameter. It is set to read-only by default (r). After reading from the file, we obtain data in text form in this mode. Below you'll find a table providing a list of the many

access options for the case of opening text file and their corresponding modes. Likewise, the binary mode, on the other hand, returns bytes. It's better for accessing non-text files like images and executable files. Here, you'll find a table. It provides a list of the many access options for the case of opening binary files and their corresponding modes.

Modes	**Description**
r	Opens a file in a reading mode
rb	Opens a binary file in a reading mode
w	Opens a writable file. Create a new file if it doesn't exist or if it already exists, truncate the file.
wb	Opens a writable binary file and also like mode w, it will create a new file if it doesn't exist or truncates on the existence
a	Opens a file without truncating at the end of a file. Create a new file if not available.
ab	Opens a binary file without truncating at the end of a file. Create a new binary file if not available.
+	Opens a file for both modes of reading and writing

In case of object-oriented case, open() function when employed in python have several linked attributes which run down as shown in below table.

Attributes	**Description**
files.closed	A Boolean attribute telling if the file is closed or not.
file.mode	Returns the file-opened access mode.
file.name	Returns file name
file.softspace	Returns false if space with print is expressly necessary, else true.

8.5 FILE POSITIONING

We notice that a newline returns as a '\n' in the read() function. When we reach the end of the file, the next time we read, we obtain a blank string. Using the seek() function, we may modify our current cursor file (position) with syntax as:

```
file_obj.seek(offset, pos)
```

where you are dealing with *file_obj* is really the file pointer; *offset* indicates how many places you move; your point of reference is defined by *pos*:

- 0: implies the start of the file is your reference point
- 1: signifies that the current file location is your reference point.
- 2: implies that the end of your file is your reference point

If *pos* argument is missed, the default is 0. Suppose we have a file with contents as shown below

```
filepos.txt

First Line of the file
```

Now, we want to read the character at the suitable position. The example of such implementation is shown below:

```
file_obj=open(“filepos.txt”,”r”)
file_obj.seek(3)
print(file_obj.readline())
```

So we have jumped three bytes over the character such that the output of the above program on running is as:

```
Output:

st Line of the file
```

Likewise, the tell() function also returns our current position (in number of bytes). The example code of the tell() function is as below considering the same file filepos.txt:

```
f_obj = open("filepos.txt", "rw+")
print("File Name: ", f_obj.name)

line = f_obj.read(4)
print(line)

# Get the current position of the file.
pos = f_obj.tell()
print(pos)

# Close opened file
fo.close()
```

8.6 CLOSING FILE IN PYTHON

The file object's closing method i.e. in-built function close() flushes any unwritten information and closes the file object, so it is no longer possible to write. However, when a reference object of a file is reassigned to a different file, Python automatically closes a file. Therefore, it is

believed that the close() function to end a file is always a good practice specially while solving real-life market related problems.

close_exmpl.py

```
file_obj=open(“filename.txt”,”wb”)
print(“The file name is: “, file_obj.name)

#closing file that was opened

file_obj.close()
```

8.7 CREATING AND APPENDING TEXT FILE IN PYTHON

We have to open it in writing w, add an or exclusive x modes to write a file in Python. We must be careful with w mode, since if it already exists, it is overwritten onto to the file. This causes the deletion of all prior data. The write() function is used to write a byte string or sequence (for binary files). This returns the number of characters in the file. Also in this example, we will study the another way opening file employing *with* keyword as:

```
with open(“newfile.txt ”,”w”) as file_obj:
        for lines in range(5):
            print(“This is way of writing file with print method”, file=file_obj)
            file_obj.write(“Rohit\n”)

file_obj.close()

with open(“newfile.txt”,”r”) as file_read:
        print(file_read.readlines())

file_read.close()
```

In the above code, we have explored the two ways i.e. using print() and write() function of writing a new file. Therefore, the output of the above program is as:

Output:

['This is way of writing file with print method\n', 'Rohit\n', 'This is way of writing file with print method\n', 'Rohit\n', 'This is way of writing file with print method\n', 'Rohit\n', 'This is way of writing file with print method\n', 'Rohit\n', 'This is way of writing file with print method\n', 'Rohit\n']

The file handle position is also defined in these modes. The handle of the file is like a cursor defining from which the information is to be read or written into the file. Open the file as a new line in append mode, with 'a' or 'a+' as an access mode, to add a new line to the existing file as:

```
#creating file in python

f_obj1 = open("newtextfile.txt", "w")
list1 = ["Rohit\n", "Virat \n", "Sharma"]
file1.writelines(L) #adding list in file using writelines() function
file1.close()

# Append-adds at last

f_obj2 = open("newtextfile.txt", "a")  # append mode
file1.write("Pant \n")   #adding contents at the bottom of file
file1.close()

f_obj3 = open("newtextfile.txt", "r")   #reading file for printing
print("Output after appending into the file ", f_obj3.name)
print()
print(f_obj3.read())
print()
f_obj3.close()
```

```
f_obj1 = open("newtextfile.txt", "w") # write mode
f_obj1.write("Hardik \n")
f_obj1.close()

f_obj1 = open("newtextfile.txt", "r")
print("Output after writing the already existing file in python ")
print(f_obj1.read())
print()
f_obj1.close()
```

8.8 FILE METHODS IN PYTHON

The file object contains numerous methods. Most of them are not much utilized in the context of python programming but eventually plays critical role. Therefore, below is the full set of methods with their respective brief description corresponding to the file in python:

Methods	Description
detach()	Separates from the TextIOBase the underlying binary buffer and returns
fileno()	Returns the file descriptor (integer number)
isatty()	Returns True if interactive file stream.
flush()	Flushes the file stream writing buffer.
readable()	Returns true on reading from the stream of the file.
seekable()	Returns True when random access is supported by the file stream.
truncate(size=None)	Resize the stream file to bytes in size. If not given, the size will be resized to the current location
writable()	Returns True if you can write the file stream to.

8.9 WORKING WITH RESPONSE DATA FILES

Whether you develop a thin client or a thick client (client server application), you definitely ask for a Web server and require a proper data format to answer your questions at some point. There are three primary types of data that are currently utilized to deliver data to a client from a web server: CSV, XML, and JSON. It is a good idea to grasp the difference between each format in order to design an application with a robust architecture and know when to utilize it. This post is intended to outline every data format, to explain the advantages and disadvantages for every single format and to find out which conditions work best.

8.9.1 Working with CSV File

You must use the reader feature to construct a reader object to read data from CSV files. The reader function is built to produce a list of all columns for each row of the file. The column for which the variable data is required is then chosen. It sounds much more complex than it is. Let us look at this CSV code in Python, and we will find out that it is not too difficult for us to deal with csv file.

Writing to CSV File

```
import csv #import modules for csv file

with open(‘dataset.csv', 'w') as f_obj:
   data_write = csv.writer(f_obj, delimiter=';', quotechar="", quoting =
csv.QUOTE_MINIMAL)

   #way to write to csv file
   data_writerow([‘Shirt No’;’Player Name’; ‘Team’; ‘Scores’]
   data_writerow([’45’;’Rohit’;’Mumbai’;’99’])
   data_writerow([’18’;’Virat’;’Bangalore’;’23’])
```

```
    data_writerow([‘7’;’Dhoni’;’Chennai’;’4’])
```

Reading from CSV File

```
import csv #import modules for csv files

with open('datatest.csv','rt')as f_obj:
  data_test = csv.reader(f_obj)
  for line in data_test:
        print(line)
```

On running the above code, the output is:

```
Output:

[‘Shirt No;Player Name; Team; Scores’]
[’45;Rohit;Mumbai;99’]
[’18;Virat;Bangalore;23’]
[‘7;Dhoni;Chennai;4’]
```

8.9.2 Working with XML File

We will use the next XML file in the examples below that we save as "players.xml":

```
                                        players.xml
<data>
    <players>
        <player name="rohit">rohit2india</player>
        <player name="virat">virat2india</player>
    </players>
</data>
```

Writing XML file

ElementTree is suitable for write XML file data. The following code illustrates how to generate an XML file with the same structure as the file in earlier instances. The following steps:

- Create an element that acts as our root. The tag for this element is "data" in our example.
- Use the SubElement function to construct sub-elements after we have our root element.

The syntax of this function is:

write_xml.py

```
import xml.etree.ElementTree as etree

# creating the structure of the XML File

data = etree.Element('data')
players = etree.SubElement(data, ‘players’)
player1 = etree.SubElement(players, ‘player’)
player2 = etree.SubElement(players, ‘player’)
player1.set('name',’rohit’)
player2.set('name','virat’)
player1.text = 'rohit2india'
player2.text = 'virat2india'

# create a new XML file with the results

my_data = etree.tostring(data)
my_file = open(''players.xml'', "w")
myfile.write(mydata)
```

Reading XML File

The *minidom* is a reduced implementation of the document object model (DOM). DOM is a programming interface for application, which handles XML as a tree structure, where each tree node represents an object. Therefore, we must be aware of the capabilities of this module.

```
from xml.dom import minidom  #importing module for xml

# parsing an xml file by name
mydoc = minidom.parse('players.xml')

players = mydoc.getElementsByTagName('player')

# printing for one specific player attribute
print('Player #2 attribute:')
print(players[1].attributes['name'].value)

# all player attributes
print('\nAll attributes:')
for elem in players:
    print(elem.attributes['name'].value)

# printing one specific player's data
print('\Player #2 data:')
```

```
print(players[1].firstChild.data)
print(players[1].childNodes[0].data)

# all players data
print('\nAll players data:')
for elem in players:
    print(elem.firstChild.data)
```

A more "Pythonic" interface to XMl is provided in ElementTree module and is an excellent solution for people not aware of the DOM. It is also probably best for more rookie programmers because of its basic interface, as you will see in this post.

```
import xml.etree.ElementTree as etree
tree = etree.parse('players.xml')
root = tree.getroot()

# printing one specific player attribute
print('Player #2 attribute:')
print(root[0][1].attrib)

# all player attributes
print('\nAll attributes:')
for elem in root:
    for sube in elem:
        print(sube.attrib)

# one specific player’s data
print('\nPlayer #2 data:')
print(root[0][1].text)

# all player data
print('\nAll player data:')
for elem in root:
    for sube in elem:
        print(sube.text)
```

The output of the both the above programs is as:

```
Output:

Player #2 attribute:
player2

All attributes:
```

```
player1
player2

Player #2 data:
virat2india

All player data:
rohit2india
virat2india
```

8.9.3 Working with JSON File

The JSON format has been one of the common ways, if not the most, to serialize data for the previous 5-10 years. You will probably be encountered with JSON especially in the web development industry through one of the various REST APIs, application settings or simply basic data storage.

Writing JSON file

You may easily send your data to a Python file in JSON format by storing your data in a dict object that may include additional nesting dicts, lists, booleans or other primitive kinds, such as integers or string. A comprehensive list of supported data types may be found here.

```
import json #imporitng json module

data = {}
data[‘player’] = []
data['player'].append({
    'name': 'Rohit',
    'Team': 'Mumbai',
    'from': 'India'
})
data['player'].append({
    'name': 'Maxwell',
    'Team': 'Bangalore',
    'from': 'Australia'
})
data['player’].append({
    'name': 'Gayle',
    'Team': 'Punjab',
    'from': 'WestIndies'
})

with open('playerdata.txt', 'w') as out_file:
    json.dump(data, out_file)
```

We create some simple data to publish to our file once we load the json library. The key section ends when we use the statement to open our target file and use json.dump to write the data object to the outfile file. Every file object, even if it is not a real file, can be passed on to the second parameter. The socket that can be opened, closed, and written like a file would be an excellent example. This is another scenario that you can meet with, as JSON is widespread all over the web.

Reading JSON file

On the other hand, it is just as straightforward to read JSON data from a file. We can extract and parse the JSON string from a file object using the same json package again. We do precisely this and then publish the data we have received in this example:

```
import json

with open('playerdata.txt') as json_file:
    data = json.load(json_file)
    for p in data['player']:
        print('Name: ' + p['name'])
        print('Team: ' + p['Team'])
        print('From: ' + p['from'])
        print('')
```

The quality and effective to note here is json.load. It will read the file string, scanning the JSON contents, adding the contents to a Python Dict and returning it.

Pretty-print in JSON file

It is as easy as supplying the integer value on the *indent* option, for JSON human readable (although "pretty printing"):

```
import json
data = {'player':[{'name': 'Rohit', 'Team': 'Mumbai', 'from': 'India'}]}
json.dumps(data, indent=4)
```

Output:

```
{
   "player": [
         "Team": "Mumbai",
         "from": "India",
         "name": "Rohit"
      }
   ]
}
```

ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇ ਨੂਰ ਚ ਰੌਸ਼ਨ ਹੈ ਇਹ ਵਿਸ਼ਵ ਵਿਦਿਆਲਾ
ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਸ਼ਬਦ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਕਿਰਤ ਕਰਮ ਦੀ
ਸ਼ਬਦ ਸੁਰਤ ਦੀ

ਸੰਗਤ ਪੰਗਤ
ਵੰਡ ਛਕਣ ਦੀ

ਖੋਜ, ਵਿਵੇਕ ਅਤੇ ਸਿਰਜਣ ਦੀ
ਕਰਤਾ ਪੁਰਖ ਰਹੱਸ ਦਰਸ਼ਨ ਦੀ

ਸਿੱਖਿਆ ਦੇਵਣ ਵਾਲਾ
ਰੌਸ਼ਨ ਹੈ ਇਹ ਵਿਸ਼ਵ ਵਿਦਿਆਲਾ

ਗਗਨ ਮੰਡਲ ਵਿਚ ਜਗਦੇ ਤਾਰੇ
ਦੀਪਕ ਸੋਹਣ ਦੁਆਰੇ ਦੁਆਰੇ

ਕਾਇਆ ਕਾਗਦ ਅੱਖਰ ਜਗਦੇ
ਪੁਸ਼ਪ ਸੁਹਾਵਣ ਧਰਤੀ ਹਿਰਦੇ

ਇਹ ਤੇਰੀ ਲੀਲਾ ਵਿਸਮਾਦੀ
ਨਿਤ ਨਵੇਲੀ ਆਦਿ ਜੁਗਾਦੀ

ਇਸ ਲੀਲਾ ਦੇ ਕਰਮ ਖੰਡ ਵਿਚ
ਤੇਰਾ ਸ਼ਬਦ ਸਵਾਰਨਹਾਰਾ
ਤੇਰਾ ਨਾਦ ਉਜਾਲਾ

ਰੌਸ਼ਨ ਹੈ ਇਹ ਵਿਸ਼ਵ ਵਿਦਿਆਲਾ

ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਸ਼ਬਦ ਗੁਰੂ ਨਾਨਕ ਦੇਵ

ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਪੰਜਾਬ ਸਟੇਟ ਓਪਨ ਯੂਨੀਵਰਸਿਟੀ
ਪਟਿਆਲਾ

JAGAT GURU NANAK DEV
PUNJAB STATE OPEN UNIVERSITY, PATIALA
(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)