DRONACHARYA
College of Engineering

LABORATORY MANUAL

B.Tech. Semester- IV

PROGRAMMING FOR DATA SCIENCE & AI LAB
Subject code: LC-CSE-258G

Prepared by:	**Checked by:**	**Approved by:**
Prof. Naveen	Dr Ritu Pahwa	Name : Prof. (Dr.) Isha Malhotra
Sign.: ……	**Sign.: ……**	**Sign.: ……**

DEPARTMENT OF CSE(AI&ML)
DRONACHARYA COLLEGE OF ENGINEERING
KHENTAWAS, FARRUKH NAGAR, GURUGRAM (HARYANA)

Table of Contents

1. Vision and Mission of the Institute
2. Vision and Mission of the Department
3. Programme Educational Objectives (PEOs)
4. Programme Outcomes (POs)
5. Programme Specific Outcomes (PSOs)
6. University Syllabus
7. Course Outcomes (COs)
8. CO- PO and CO-PSO mapping
9. Course Overview
10. List of Experiments
11. DOs and DON'Ts
12. General Safety Precautions
13. Guidelines for students for report preparation
14. Lab assessment criteria
15. Lab Experiments

Vision and Mission of the Institute

Vision:

"To impart Quality Education, to give an enviable growth to seekers of learning, to groom them as World Class Engineers and managers competent to match the expending expectations of the Corporate World has been ever enlarging vision extending to new horizons of Dronacharya College of Engineering"

Mission:

M1: To prepare students for full and ethical participation in a diverse society and encourage lifelong learning by following the principle of 'Shiksha evam Sahayata' i.e., Education & Help.

M2: To impart high-quality education, knowledge and technology through rigorous academic programs, cutting-edge research, & Industry collaborations, with a focus on producing engineers& managers who are socially responsible, globally aware, & equipped to address complex challenges.

M3: Educate students in the best practices of the field as well as integrate the latest research into the academics.

M4: Provide quality learning experiences through effective classroom practices, innovative teaching practices and opportunities for meaningful interactions between students and faculty.

M5: To devise and implement programmes of education in technology that are relevant to the changing needs of society, in terms of breadth of diversity and depth of specialization.

Vision and Mission of the Department

Vision:

To cultivate skills and make proficient engineers cum trainers in the domain of Artificial Intelligence & Machine Learning for exceptional contributions to the society.

Mission:

M1: To impart intense training and learning to generate knowledge through the state-of-the-art concepts and technologies in Artificial Intelligence and Machine Learning.

M2: To establish centres of excellence by collaborating with the leading industries to exhilarate innovative research and development in AIML and its allied technology.

M3: To inculcate regenerative self-learning abilities, team spirit, and professional ethics among the students for noble cause.

Programme Educational Objectives (PEOs)

PEO1- ANALYTICAL SKILLS:

Using a solid foundation in mathematical, scientific, engineering, and current computing principles, formulate, analyse, and resolve engineering issues in real-world domain.

PEO2- TECHNICAL SKILLS:

Apply artificial intelligence theory and concepts to analyse the requirements, realise technical specifications, and design engineering solutions.

PEO3- SOFT SKILLS:

Through inter-disciplinary projects and a variety of professional activities, demonstrate technical proficiency, AI competency, and foster collaborative learning and a sense of teamwork.

PEO4- PROFESSIONAL ETHICS:

Excel as socially responsible engineers or entrepreneurs with high moral and ethical standards, competence, and soft skills that will enable them to contribute to societal demands and achieve sustainable advancement in emerging computer technologies.

PROGRAM OUTCOMES (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

P11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

P12: Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Fundamentals and critical knowledge of the Computer System: Apply the knowledge gained pertaining to build, asses, and analyze the software and hardware aspects of the program to solve real-world business problems.

PSO2: Comprehensive and applicative knowledge of Software Development: Ability to evaluate and apply knowledge of data engineering, methodologies, and able to plan, develop, test, analyze, and manage required aspects in heterogenous platforms individually or in team work.

PSO3: **Applications in Computing Domain:** Ability to acquire computational knowledge and project development abilities using novel tools and methodologies to tackle challenges in the fields related to Deep Learning, Machine learning, Artificial Intelligence.

PSO4: Applications in Innovations and Research: Capacity to direct a team or firm that develops products and to use the knowledge learned to recognise actual research issues

University Syllabus

Course code	LC-CSE-258G				
Category	Professional core courses				
Course title	Programming for Data Science & AI Lab				
Scheme and Credits	L	T	P	Credits	Semester = 4
	0	0	2	1	
Classwork	25 Marks				
Exam	25 Marks				
Total	50 Marks				
Duration of Exam	03 Hours				

Tentative List of Experiments:

1. Python program to display details about the operating system, working directory, files and directories in the current directory, lists the files and all directories, scan and classify them as directories and files.
2. Python program to convert an array to an array of machine values and vice versa.
3. Python program to get information about the file pertaining to the file mode and to get time values with components using local time and gm time.
4. Python program to connect to Google using socket programming.
5. Python program to perform Array operations using Numpy package.
6. Python program to perform Data Manipulation operations using Pandas package.
7. Python program to display multiple types of charts using Matplotlib package.
8. Python program to perform File Operation on Excel Data Set.
9. Python program to implement with Python Sci Kit-Learn & NLTK.
10. Python program to implement with Python NLTK/Spicy/Py NLPI.

Course Outcomes (COs)

Upon successful completion of the course, the students will be able to:

CO1: Implement solutions to the given assignments in Python.
CO2: Use various Python packages for solving different programming problems.
CO3: Devise solutions for complex problems of data analysis and machine learning.
CO4: Evaluate the output of data analysis and machine learning models.
CO5: Create lab records of the solutions for the given assignments.
CO6: Demonstrate the use of ethical practices, self-learning and team spirit.

CO-PO Mapping

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	1			3				3	3	3	3
CO2	2	3		3	3				3	3	3	3
CO3	3	3	1	3	3	3			3	3	3	3
CO4	3	3		3	3				3	3	3	3
CO5										3		
CO6								3	3			3

CO-PSO Mapping

CO	PSO1	PSO2	PSO3	PSO4
CO1	3	3	2	
CO2	3	3	3	3
CO3	2	3	3	3
CO4	2	2	3	2
CO5				3
CO6			2	2

*3-HIGH
*2-MEDIUM
*1-LOW

Course Overview

The Programming for Data Science & AI Lab is a practical course that focuses on enhancing students' Python programming skills and introducing them to key libraries and tools for data analysis, data visualization, natural language processing, data science and artificial intelligence (AI) applications.

This lab course serves as a practical companion to the Programming for Data Science & AI theory course and aims to provide students with practical experience in applying object-oriented programming concepts and utilizing essential Python libraries.

Through a combination of programming assignments, coding exercises, and mini-projects, students will gain proficiency in using programming languages and frameworks commonly used in data science and AI, and acquire the necessary skills to tackle real-world data-driven challenges.

List of Experiments mapped with COs

S. No.	Program Name	Course Outcomes
1.	Python program to display details about the operating system, working directory, files and directories in the current directory, lists the files and all directories, scan and classify them as directories and files.	CO1
2.	Python program to convert an array to an array of machine values and vice versa.	CO1
3.	Python program to get information about the file pertaining to the file mode and to get time values with components using local time and gm time.	CO1, CO2
4.	Python program to connect to Google using socket programming.	CO2
5.	Python program to perform Array operations using Numpy package.	CO2, CO3
6.	Python program to perform Data Manipulation operations using Pandas package.	CO2, CO4
7.	Python program to display multiple types of charts using Matplotlib package.	CO2, CO4
8.	Python program to perform File Operation on Excel Data Set.	CO2, CO3
9.	Python program to implement with Python Sci Kit-Learn & NLTK.	CO2, CO4
10.	Python program to implement with Python NLTK/Spicy/Py NLPI.	CO2, CO4

DOs and DON'Ts

DOs

1. Login-on with your username and password.
2. Log off the Computer every time when you leave the Lab.
3. Arrange your chair properly when you are leaving the lab.
4. Put your bags in the designated area.
5. Ask permission to print.

DON'Ts

1. Do not share your username and password.
2. Do not remove or disconnect cables or hardware parts.
3. Do not personalize the computer setting.
4. Do not run programs that continue to execute after you log off.
5. Do not download or install any programs, games or music on computer in Lab.
6. Personal Internet use chat room for Instant Messaging (IM) and Sites is strictly prohibited.
7. No Internet gaming activities allowed.
8. Tea, Coffee, Water & Eatables are not allowed in the Computer Lab.

General Safety Precautions

Precautions (In case of Injury or Electric Shock)

1. To break the victim with live electric source, use an insulator such as fire wood or plastic to break the contact. Do not touch the victim with bare hands to avoid the risk of electrifying yourself.

2. Unplug the risk of faulty equipment. If main circuit breaker is accessible, turn the circuit off.

3. If the victim is unconscious, start resuscitation immediately, use your hands to press the chest
in and out to continue breathing function. Use mouth-to-mouth resuscitation if necessary.

4. Immediately call medical emergency and security. Remember! Time is critical; be best.

Precautions (In case of Fire)

1. Turn the equipment off. If power switch is not immediately accessible, take plug off.

2. If fire continues, try to curb the fire, if possible, by using the fire extinguisher or by covering it with a heavy cloth if possible, isolate the burning equipment from the other surrounding equipment.

3. Sound the fire alarm by activating the nearest alarm switch located in the hallway.

4. Call security and emergency department immediately:

Emergency : 201 (Reception)
Security: 231 (Gate No.1)

Guidelines to students for report preparation

All students are required to maintain a record of the experiments conducted by them. Guidelines for its preparation are as follows: -

1) All files must contain a title page followed by an index page. ***The files will not be signed by the faculty without an entry in the index page.***
2) Student's Name, roll number and date of conduction of experiment must be written on all pages.
3) For each experiment, the record must contain the following
   (i) Aim/Objective of the experiment
   (ii) Pre-experiment work (as given by the faculty)
   (iii) Lab assignment questions and their solutions
   (iv) Test Cases (if applicable to the course)
   (v) Results/ output

Note:

1. Students must bring their lab record along with them whenever they come for the lab.
2. Students must ensure that their lab record is regularly evaluated.

Lab Assessment Criteria

An estimated 10 lab classes are conducted in a semester for each lab course. These lab classes are assessed continuously. Each lab experiment is evaluated based on 5 assessment criteria as shown in following table. Assessed performance in each experiment is used to compute CO attainment as well as internal marks in the lab course.

Grading Criteria	Exemplary (4)	Competent (3)	Needs Improvement (2)	Poor (1)
AC1: Pre-Lab written work (this may be assessed through viva)	Complete procedure with underlined concept is properly written	Underlined concept is written but procedure is incomplete	Not able to write concept and procedure	Underlined concept is not clearly understood
AC2: Program Writing/ Modeling	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out	Assigned problem is properly analyzed, correct solution designed, appropriate language constructs/ tools are applied	Assigned problem is properly analyzed & correct solution designed	Assigned problem is properly analyzed
AC3: Identification & Removal of errors/ bugs	Able to identify errors/ bugs and remove them	Able to identify errors/ bugs and remove them with little bit of guidance	Is dependent totally on someone for identification of errors/ bugs and their removal	Unable to understand the reason for errors/ bugs even after they are explicitly pointed out
AC4: Execution & Demonstration	All variants of input /output are tested, Solution is well demonstrated and implemented concept is clearly explained	All variants of input /output are not tested, However, solution is well demonstrated and implemented concept is clearly explained	Only few variants of input /output are tested, Solution is well demonstrated but implemented concept is not clearly explained	Solution is not well demonstrated and implemented concept is not clearly explained
AC5: Lab Record Assessment	All assigned problems are well recorded with objective, design constructs and solution along with Performance analysis using all variants of input and output	More than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	Less than 70 % of the assigned problems are well recorded with objective, design contracts and solution along with Performance analysis is done with all variants of input and output	

LAB EXPERIMENTS

PROGRAM NO. 1

AIM: - Python program to display details about the operating system, working directory, files, and directories in the current directory, lists the files and all directories, scan and classify them as directories and files.

SOURCE CODE: -

```
import os

# Display operating system details
print("Operating System: ", os.name)

# Display working directory
print("Working Directory: ", os.getcwd())

# Get list of files and directories in the current directory
files_and_dirs = os.listdir()

# Separate files and directories
files = []
directories = []

for item in files_and_dirs:
    if os.path.isfile(item):
        files.append(item)
    elif os.path.isdir(item):
        directories.append(item)

# Display list of files
print("\nFiles:")
for file in files:
    print(file)

# Display list of directories
print("\nDirectories:")
for directory in directories:
print(directory)
```

OUTPUT:

Operating System: posix
Current Working Directory: /path/to/current/directory

Files in the current directory:
file1.txt
file2.py
file3.jpg

Directories in the current directory:
dir1
dir2

PROGRAM NO. 2

AIM: - Python program to convert an array to an array of machine values and vice versa.

SOURCE CODE: -

```
import struct
def array_to_bytes(array):
    # Convert array to bytes
    format_string = '{}{}'.format(len(array), 'B')
    packed_data = struct.pack(format_string, *array)
    return packed_data

def bytes_to_array(bytes_data):
    # Convert bytes to array
    format_string = '{}{}'.format(len(bytes_data), 'B')
    unpacked_data = struct.unpack(format_string, bytes_data)
    return list(unpacked_data)

# Example usage
input_array = [10, 20, 30, 40, 50]

# Convert array to bytes
bytes_data = array_to_bytes(input_array)
print("Array as bytes:", bytes_data)

# Convert bytes to array
output_array = bytes_to_array(bytes_data)
print("Bytes as array:", output_array)
```

OUTPUT:

Array as bytes: b'\n\x14\x1e(2'
Bytes as array: [10, 20, 30, 40, 50]

PROGRAM NO. 3

AIM: - Python program to get information about the file pertaining to the file mode and to get time values with components using local time and gm time.

SOURCE CODE: -

```
import os
import time

# Get file information
def get_file_info(file_path):
    # Check if file exists
    if not os.path.exists(file_path):
        print("File does not exist.")
        return

    # Get file mode
    file_mode = os.stat(file_path).st_mode
    print("File Mode:", file_mode)

    # Get time values using local time
    local_time = os.path.getmtime(file_path)
    local_time_components = time.localtime(local_time)
    print("Local Time:")
    print("Year:", local_time_components.tm_year)
    print("Month:", local_time_components.tm_mon)
    print("Day:", local_time_components.tm_mday)
    print("Hour:", local_time_components.tm_hour)
    print("Minute:", local_time_components.tm_min)
    print("Second:", local_time_components.tm_sec)

    # Get time values using GMT (UTC)
    gmt_time = os.path.getmtime(file_path)
```

```
    gmt_time_components = time.gmtime(gmt_time)
    print("\nGMT (UTC) Time:")
    print("Year:", gmt_time_components.tm_year)
    print("Month:", gmt_time_components.tm_mon)
    print("Day:", gmt_time_components.tm_mday)
    print("Hour:", gmt_time_components.tm_hour)
    print("Minute:", gmt_time_components.tm_min)
    print("Second:", gmt_time_components.tm_sec)

# Example usage
file_path = "path/to/your/file.txt"
get_file_info(file_path)
```

OUTPUT:

```
File Mode: 33188
Local Time:
Year: 2023
Month: 5
Day: 15
Hour: 10
Minute: 30
Second: 45

GMT (UTC) Time:
Year: 2023
Month: 5
Day: 15
Hour: 15
Minute: 30
Second: 45
```

PROGRAM NO. 4

AIM: - Python program to connect to Google using socket programming.

SOURCE CODE: -

```
import socket

def connect_to_google():
    host = "www.google.com"
    port = 80

    try:
        # Create a socket object
        client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        # Connect to Google server
        client_socket.connect((host, port))

        print("Connected to Google successfully.")

        # Close the socket connection
        client_socket.close()

    except socket.error as e:
        print("Failed to connect to Google. Error:", e)

if __name__ == "__main__":
    connect_to_google()
```

PROGRAM NO. 5

AIM: - Python program to perform Array operations using Numpy package.

SOURCE CODE: -

```
import numpy as np
# Create arrays
a = np.array([1, 2, 3, 4, 5])
b = np.array([6, 7, 8, 9, 10])

# Basic operations
print("Array a:", a)
print("Array b:", b)
print("Sum of arrays a and b:", np.add(a, b))
print("Difference of arrays a and b:", np.subtract(a, b))
print("Product of arrays a and b:", np.multiply(a, b))
print("Division of arrays a and b:", np.divide(a, b))
print("Square root of array a:", np.sqrt(a))
print("Exponential of array a:", np.exp(a))

# Aggregation operations
print("Minimum value of array a:", np.min(a))
print("Maximum value of array b:", np.max(b))
print("Mean of array a:", np.mean(a))
print("Standard deviation of array b:", np.std(b))
print("Sum of all elements in array a:", np.sum(a))

# Reshaping arrays
c = np.array([[1, 2], [3, 4], [5, 6]])
```

```
print("Array c:")
print(c)
print("Reshaped array c (2 rows, 3 columns):")
print(np.reshape(c, (2, 3)))

# Transposing arrays
d = np.array([[1, 2, 3], [4, 5, 6]])
print("Array d:")
print(d)
print("Transposed array d:")
print(np.transpose(d))
```

OUTPUT:

```
Array a: [1 2 3 4 5]
Array b: [ 6  7  8  9 10]
Sum of arrays a and b: [ 7  9 11 13 15]
Difference of arrays a and b: [-5 -5 -5 -5 -5]
Product of arrays a and b: [ 6 14 24 36 50]
Division of arrays a and b: [0.16666667 0.28571429 0.375      0.44444444 0.5       ]
Square root of array a: [1.         1.41421356 1.73205081 2.         2.23606798]
Exponential of array a: [ 2.71828183  7.3890561  20.08553692 54.59815003 148.4131591 ]
Minimum value of array a: 1
Maximum value of array b: 10
Mean of array a: 3.0
Standard deviation of array b: 1.5811388300841898
Sum of all elements in array a: 15
Array c:
[[1 2]
 [3 4]
 [5 6]]
Reshaped array c (2 rows, 3 columns):
[[1 2 3]
 [4 5 6]]
Array d:
[[1 2 3]
 [4 5 6]]
Transposed array d:
[[1 4]
 [2 5]
 [3 6]]
```

PROGRAM NO. 6

AIM: - Python program to perform Data Manipulation operations using Pandas package.

SOURCE CODE: -

```
import pandas as pd

# Create a DataFrame
data = {
    'Name': ['John', 'Emma', 'Sam', 'Lisa', 'Tom'],
    'Age': [25, 30, 28, 32, 27],
    'Country': ['USA', 'Canada', 'Australia', 'UK', 'Germany'],
    'Salary': [50000, 60000, 55000, 70000, 52000]
}
df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)

# Selecting columns
name_age = df[['Name', 'Age']]
print("\nName and Age columns:")
print(name_age)

# Filtering rows
filtered_df = df[df['Country'] == 'USA']
print("\nFiltered DataFrame (Country = 'USA'):")
print(filtered_df)

# Sorting by a column
sorted_df = df.sort_values('Salary', ascending=False)
```

```
print("\nSorted DataFrame (by Salary in descending order):")
print(sorted_df)

# Aggregating data
average_salary = df['Salary'].mean()
print("\nAverage Salary:", average_salary)

# Adding a new column
df['Experience'] = [3, 6, 4, 8, 5]
print("\nDataFrame with added Experience column:")
print(df)

# Updating values
df.loc[df['Name'] == 'Emma', 'Salary'] = 65000
print("\nDataFrame after updating Emma's Salary:")
print(df)

# Deleting a column
df = df.drop('Experience', axis=1)
print("\nDataFrame after deleting Experience column:")
print(df)
```

OUTPUT:

Original DataFrame:

	Name	Age	Country	Salary
0	John	25	USA	50000
1	Emma	30	Canada	60000
2	Sam	28	Australia	55000
3	Lisa	32	UK	70000
4	Tom	27	Germany	52000

Name and Age columns:

	Name	Age
0	John	25
1	Emma	30
2	Sam	28
3	Lisa	32
4	Tom	27

Filtered DataFrame (Country = 'USA'):

	Name	Age	Country	Salary
0	John	25	USA	50000

Sorted DataFrame (by Salary in descending order):

	Name	Age	Country	Salary
3	Lisa	32	UK	70000
1	Emma	30	Canada	60000
2	Sam	28	Australia	55000
4	Tom	27	Germany	52000
0	John	25	USA	50000

Average Salary: 57400.0

DataFrame with added Experience column:

	Name	Age	Country	Salary	Experience
0	John	25	USA	50000	3
1	Emma	30	Canada	60000	6
2	Sam	28	Australia	55000	4
3	Lisa	32	UK	70000	8
4	Tom	27	Germany	52000	5

DataFrame after updating Emma's Salary:

	Name	Age	Country	Salary	Experience
0	John	25	USA	50000	3
1	Emma	30	Canada	65000	6
2	Sam	28	Australia	55000	4
3	Lisa	32	UK	70000	8
4	Tom	27	Germany	52000	5

DataFrame after deleting Experience column:

	Name	Age	Country	Salary
0	John	25	USA	50000
1	Emma	30	Canada	65000
2	Sam	28	Australia	55000
3	Lisa	32	UK	70000
4	Tom	27	Germany	52000

PROGRAM NO. 7

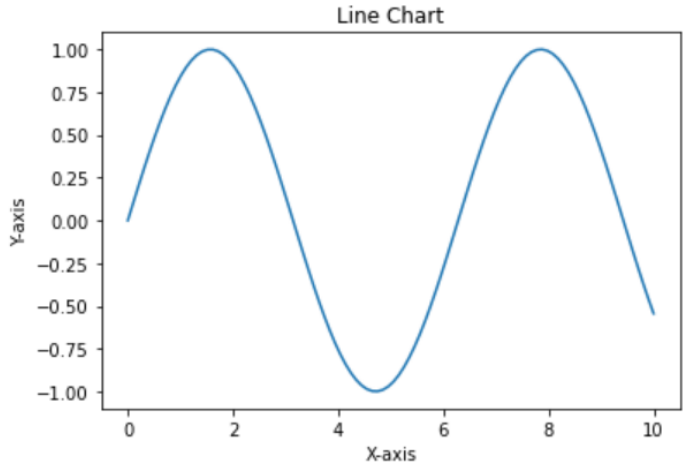
AIM: - Python program to display multiple types of charts using Matplotlib package.

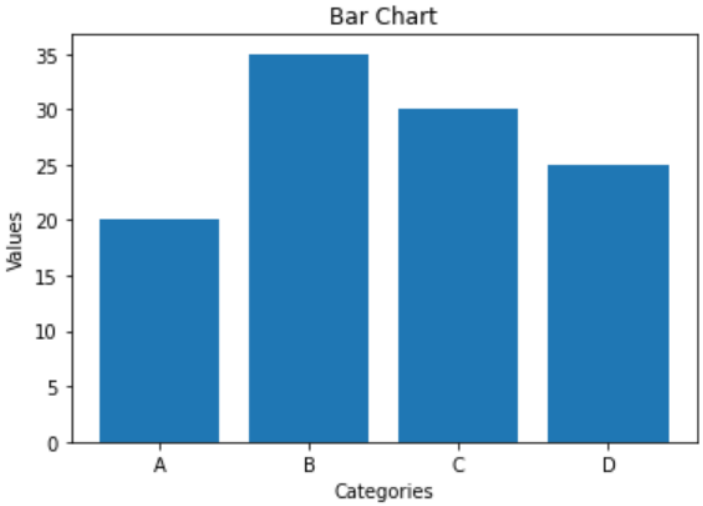
SOURCE CODE: -

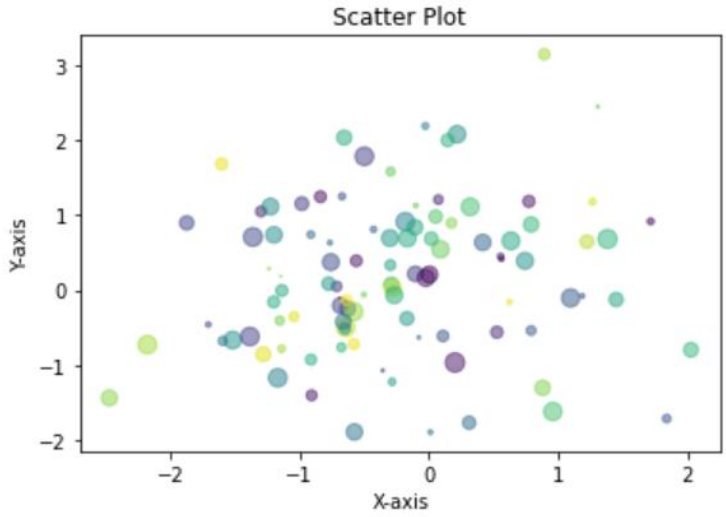
```
import matplotlib.pyplot as plt
import numpy as np

# Line chart
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.figure()
plt.plot(x, y)
plt.title("Line Chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

# Bar chart
categories = ['A', 'B', 'C', 'D']
values = [20, 35, 30, 25]
plt.figure()
plt.bar(categories, values)
plt.title("Bar Chart")
plt.xlabel("Categories")
plt.ylabel("Values")

# Scatter plot
x = np.random.randn(100)
y = np.random.randn(100)
colors = np.random.rand(100)
sizes = 100 * np.random.rand(100)
plt.figure()
plt.scatter(x, y, c=colors, s=sizes, alpha=0.5)
```

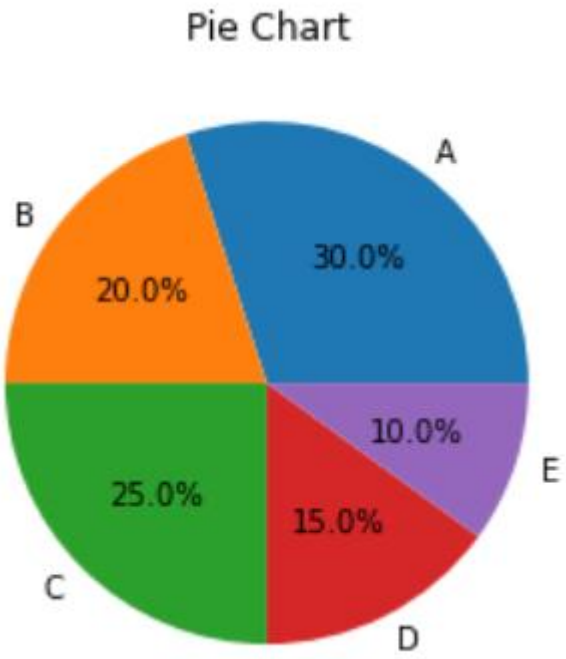
```
plt.title("Scatter Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")

# Pie chart
sizes = [30, 20, 25, 15, 10]
labels = ['A', 'B', 'C', 'D', 'E']
plt.figure()
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title("Pie Chart")

# Show all the charts
plt.show()
```

OUTPUT:


Line Chart
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Y-axis
0
2
4
6
8
10
X-axis


Bar Chart
35
30
25
20
15
10
5
0
Values
A
B
C
D
Categories


Scatter Plot
Y-axis
X-axis
3
2
1
0
−1
−2
−2
−1
0
1
2


Pie Chart
A
B
C
D
E
30.0%
20.0%
25.0%
15.0%
10.0%

PROGRAM NO. 8

AIM: - Python program to perform File Operation on Excel Data Set.

SOURCE CODE: -

```
import pandas as pd
# Read Excel file
df = pd.read_excel('data.xlsx')

# Display first few rows
print("First few rows:")
print(df.head())

# Get summary statistics
print("\nSummary statistics:")
print(df.describe())

# Filter data
filtered_data = df[df['Age'] > 30]
print("\nFiltered data (Age > 30):")
print(filtered_data)

# Sort data
sorted_data = df.sort_values(by='Salary', ascending=False)
print("\nSorted data (by Salary):")
print(sorted_data)

# Add a new column
df['Bonus'] = df['Salary'] * 0.1
print("\nData with new column (Bonus):")
print(df)

# Write to Excel file
df.to_excel('output.xlsx', index=False)
```

```
print("\nData written to output.xlsx")
```

OUTPUT:

```
First few rows:
   Name  Age  Salary
0  John   25   50000
1  Emma   30   60000
2   Sam   28   55000
3  Lisa   32   70000
4   Tom   27   52000

Summary statistics:
             Age        Salary
count   5.000000      5.000000
mean   28.400000  57400.000000
std     2.701851   8001.661438
min    25.000000  50000.000000
25%    27.000000  52000.000000
50%    28.000000  55000.000000
75%    30.000000  60000.000000
max    32.000000  70000.000000

Filtered data (Age > 30):
  Name  Age  Salary
3  Lisa   32   70000

Sorted data (by Salary):
   Name  Age  Salary
3  Lisa   32   70000
1  Emma   30   60000
2   Sam   28   55000
4   Tom   27   52000
```

0 John 25 50000

Data with new column (Bonus):

Name Age Salary Bonus

0 John 25 50000 5000.0

1 Emma 30 60000 6000.0

2 Sam 28 55000 5500.0

3 Lisa 32 70000 7000.0

4 Tom 27 52000 5200.0

Data written to output.xlsx

PROGRAM NO. 9

AIM: - Python program to implement with Python Sci Kit-Learn & NLTK.

SOURCE CODE: -

```
import nltk
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score

# Prepare sample data
sentences = ['I love Python programming.',
        'I dislike writing code.',
        'Machine learning is fascinating.',
        'Natural language processing is challenging.']

labels = ['positive', 'negative', 'positive', 'negative']

# Tokenization and preprocessing
nltk.download('punkt')
corpus = [nltk.word_tokenize(sentence) for sentence in sentences]

# Convert corpus to feature vectors
vectorizer = CountVectorizer()
X = vectorizer.fit_transform([' '.join(sentence) for sentence in corpus])

# Apply TF-IDF transformation
transformer = TfidfTransformer()
X = transformer.fit_transform(X)

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, labels, test_size=0.2, random_state=42)
```

```
# Train a Linear SVM classifier
classifier = LinearSVC()
classifier.fit(X_train, y_train)

# Make predictions on test data
y_pred = classifier.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy:', accuracy)
```

OUTPUT:

```
Accuracy: 0.5
```

PROGRAM NO. 10

AIM: - Python program to implement with Python NLTK/Spicy/Py NLPI.

SOURCE CODE: -

```
import nltk
import spacy
import pynlpi

# NLTK Example: Tokenization
nltk.download('punkt')
sentence = "Natural Language Processing is awesome!"
tokens = nltk.word_tokenize(sentence)
print("NLTK Tokenization:")
print(tokens)

# SpaCy Example: Part-of-Speech Tagging
nlp = spacy.load('en_core_web_sm')
doc = nlp(sentence)
print("\nSpaCy Part-of-Speech Tagging:")
for token in doc:
    print(token.text, token.pos_)

# PyNLPI Example: Sentiment Analysis
sentiment_score = pynlpi.sentiment_score(sentence)
print("\nPyNLPI Sentiment Analysis Score:")
print(sentiment_score)
```

OUTPUT:

NLTK Tokenization:

['Natural', 'Language', 'Processing', 'is', 'awesome', '!']

SpaCy Part-of-Speech Tagging:

Natural ADJ

Language PROPN

Processing NOUN

is AUX

awesome ADJ

! PUNCT

PyNLPI Sentiment Analysis Score:

0.53125

Viva – Voice Questions

1. What is Python? What are the benefits of using Python?

Python is a high-level, interpreted, general-purpose programming language. Being a general-purpose language, it can be used to build almost any type of application with the right tools/libraries. Additionally, python supports objects, modules, threads, exception-handling, and automatic memory management which help in modelling real-world problems and building applications to solve these problems.

Benefits of using Python:

- Python is a general-purpose programming language that has a simple, easy-to-learn syntax that emphasizes readability and therefore reduces the cost of program maintenance. Moreover, the language is capable of scripting, is completely open-source, and supports third-party packages encouraging modularity and code reuse.
- Its high-level data structures, combined with dynamic typing and dynamic binding, attract a huge community of developers for Rapid Application Development and deployment.

2. What is docstring in Python?

- Documentation string or docstring is a multiline string used to document a specific code segment.
- The docstring should describe what the function or method does.

3. What is Scope in Python?

Every object in Python functions within a scope. A scope is a block of code where an object in Python remains relevant. Namespaces uniquely identify all the objects inside a program. However, these namespaces also have a scope defined for them where you could use their objects without any prefix. A few examples of scope created during code execution in Python are as follows:

- A **local scope** refers to the local objects available in the current function.
- A **global scope** refers to the objects available throughout the code execution since their inception.
- A **module-level scope** refers to the global objects of the current module accessible in the program.
- An **outermost scope** refers to all the built-in names callable in the program. The objects in this scope are searched last to find the name referenced.

Note: Local scope objects can be synced with global scope objects using keywords such as **global**.

4. What is slicing in Python?

- As the name suggests, 'slicing' is taking parts of.
- Syntax for slicing is **[start : stop : step]**
- **start** is the starting index from where to slice a list or tuple
- **stop** is the ending index or where to sop.
- **step** is the number of steps to jump.
- Default value for **start** is 0, **stop** is number of items, **step** is 1.
- Slicing can be done on **strings, arrays, lists**, and **tuples**.

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(numbers[1 : : 2])  #output : [2, 4, 6, 8, 10]
```

5. What are global, protected and private attributes in Python?

- **Global** variables are public variables that are defined in the global scope. To use the variable in the global scope inside a function, we use the global keyword.
- **Protected** attributes are attributes defined with an underscore prefixed to their identifier eg. _sara. They can still be accessed and modified from outside the class they are defined in but a responsible developer should refrain from doing so.
- **Private** attributes are attributes with double underscore prefixed to their identifier eg. __ansh. They cannot be accessed or modified from the outside directly and will result in an AttributeError if such an attempt is made.

6. What is the use of self in Python?

Self is used to represent the instance of the class. With this keyword, you can access the attributes and methods of the class in python. It binds the attributes with the given arguments. self is used in different places and often thought to be a keyword. But unlike in C++, self is not a keyword in Python.

7. What is __init__?

__init__ is a contructor method in Python and is automatically called to allocate memory when a new object/instance is created. All classes have a **__init__** method associated with them. It helps in distinguishing methods and attributes of a class from local variables.

```
# class definition
class Student:
  def __init__(self, fname, lname, age, section):
     self.firstname = fname
     self.lastname = lname
     self.age = age
     self.section = section
# creating a new object
stu1 = Student("Sara", "Ansh", 22, "A2")
```

8. What are Python Modules?

Files containing Python codes are referred to as Python modules. This code can be of different types like classes, functions, or variables. This saves the programmer's time by providing predefined functionalities when needed. It is a file with ".py" extension containing an executable code.

Commonly used built modules are listed below:

- os
- sys
- data time
- math
- random
- JSON

Which is faster, python list or Numpy arrays, and why?
A. NumPy arrays are faster than Python lists for numerical operations. NumPy is a library for working with arrays in Python, and it provides a number of functions for performing operations on arrays efficiently.
One reason why NumPy arrays are faster than Python lists is that NumPy arrays are implemented in C, while Python lists are implemented in Python. This means that operations on NumPy arrays are implemented in a compiled language, which makes them faster than operations on Python lists, which are implemented in an interpreted language.

9. Read csv data in Python.
We can read a csv file using commands:
import pandas as pd
Df1 = pd.read_csv('filename.csv')

10. Explain zip() function in Python.
Zip()- Takes two iterables and combines them to form a single iterable.
lst1 = [15, 14, 13, 12, 11, 10]
lst2 = [10, 11, 12, 13, 14, 15]
x= zip(lst1, lst2)
print(set(x))
Output:
{(14, 11), (13, 12), (12, 13), (15, 10), (10, 15), (11, 14)}

11. What are ways of creating 1D, 2D and 3D arrays in NumPy?
Consider you have a normal python list. From this, we can create NumPy arrays by making use of the array function as follows:

- One-Dimensional array

```
import numpy as np

arr = [1,2,3,4]       #python list
numpy_arr = np.array(arr)    #numpy array
```

- Two-Dimensional array

```
import numpy as np

arr = [[1,2,3,4],[4,5,6,7]]
numpy_arr = np.array(arr)
```

- Three-Dimensional array

```
import numpy as np

arr = [[[1,2,3,4],[4,5,6,7],[7,8,9,10]]]
numpy_arr = np.array(arr)
```

Using the np.array() function, we can create NumPy arrays of any dimensions.

12. How do you find the data type of the elements stored in the NumPy arrays?
NumPy supports the following datatypes:

- i - integer
- S - string
- b - boolean
- f - float
- u - unsigned integer
- c - complex float
- m - timedelta
- M - datetime
- O - object
- U - unicode string
- V - fixed memory chunk for types such as void

13. How is np.mean() different from np.average() in NumPy?

- np.mean() method calculates the arithmetic mean and provides additional options for input and results. For example, it has the option to specify what data types have to be taken, where the result has to be placed etc.
- np.average() computes the weighted average if the weights parameter is specified. In the case of weighted average, instead of considering that each data point is contributing equally to the final average, it considers that some data points have more weightage than the others (unequal contribution).

14. Mention the different types of Data Structures in Pandas?

Pandas provide two data structures, which are supported by the pandas library, **Series,** and **DataFrames.** Both of these data structures are built on top of the NumPy.
A **Series** is a one-dimensional data structure in pandas, whereas the **DataFrame** is the two-dimensional data structure in pandas.

15. Define DataFrame in Pandas?

A DataFrame is a widely used data structure of pandas and works with a two-dimensional array with labeled axes (rows and columns) DataFrame is defined as a standard way to store data and has two different indexes, i.e., row index and column index. It consists of the following properties:

- The columns can be heterogeneous types like int and bool.
- It can be seen as a dictionary of Series structure where both the rows and columns are indexed. It is denoted as "columns" in the case of columns and "index" in case of rows.

16. What are the significant features of the pandas Library?

The key features of the panda's library are as follows:

- Memory Efficient
- Data Alignment
- Reshaping
- Merge and join
- Time Series

17. How can we create a copy of the series in Pandas?

We can create the copy of series by using the following syntax:
pandas.Series.copy
Series.copy(deep=True)
The above statements make a deep copy that includes a copy of the data and the indices. If we set the value of deep to **False**, it will neither copy the indices nor the data.

18. How to Rename the Index or Columns of a Pandas DataFrame?
You can use the **.rename** method to give different values to the columns or the index values of DataFrame.

19. How can we convert DataFrame into a NumPy array?
For performing some high-level mathematical functions, we can convert Pandas DataFrame to numpy arrays. It uses the **DataFrame.to_numpy()** function.
The **DataFrame.to_numpy()** function is applied to the DataFrame that returns the numpy ndarray.

```
DataFrame.to_numpy(dtype=None, copy=False)
```

20. What is Matplotlib?
Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

21. What is the difference between pyplot and pylab?
Pyplot is a module in matplotlib that provides a set of functions for creating and manipulating plots. Pylab is a module that combines pyplot with numpy, providing a set of tools for numerical calculations and plotting.

22. What's the best way to draw multiple lines in a single figure?
The best way to draw multiple lines in a single figure is to use the plot() function. This function can take multiple arguments, each of which will be plotted as a separate line. For example, to plot two lines, you would use the following code:
plot(x1, y1, x2, y2)

23. Is it possible to modify the tick label size for all plots in Matplotlib? If yes, then how?
Yes, it is possible to modify the tick label size for all plots in Matplotlib. This can be done by using the rcParams command.