Index

Practical -1

Aim: Write a python script to print a statement.

Program:

```
# This is a Python script that prints a statement

print("I am a student of SRMIST")
```

Output:

```
print("I am a student of SRMIST")
```

Practical -2

Program: Write a python Program to check that the given number is even or odd.

Program:

```
number = int(input("Enter a number: "))
if number % 2 == 0:
    print("The number is even")
else:
    print("The number is odd")
```

Output:

```
Enter a number: 5
The number is odd
```

Practical -3

Aim: Write a python program for swapping two numbers.

```
x = int( input("Please enter value for Ist No: "))

y = int( input("Please enter value for IInd No: "))

  # To swap the value of two variables

# we will user third variable which is a temporary variable

temp = x

x = y

y = temp

print ("The Value of Ist No. after swapping: ", x)

print ("The Value of IInd No. after swapping: ", y)
```

Output:

```
Please enter value for Ist No: 2
Please enter value for IInd No: 3
The Value of Ist No. after swapping:  3
The Value of IInd No. after swapping:  2
```

Practical -4

Aim: Write a python script to Explore iris data set.

Program:

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import load_iris


# Load the Iris dataset

iris = load_iris()

data = pd.DataFrame(data=iris.data, columns=iris.feature_names)

data['target'] = iris.target


# Basic exploration

print("First 5 rows of the dataset:")

print(data.head())


print("\nSummary statistics of the dataset:")

print(data.describe())
```

Output:

```
First 5 rows of the dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
```

3 0
4 0

Summary statistics of the dataset:

	sepal length (cm)	sepal width (cm)	petal length (cm)	\
count	150.000000	150.000000	150.000000	
mean	5.843333	3.057333	3.758000	
std	0.828066	0.435866	1.765298	
min	4.300000	2.000000	1.000000	
25%	5.100000	2.800000	1.600000	
50%	5.800000	3.000000	4.350000	
75%	6.400000	3.300000	5.100000	
max	7.900000	4.400000	6.900000	

	petal width (cm)	target
count	150.000000	150.000000
mean	1.199333	1.000000
std	0.762238	0.819232
min	0.100000	0.000000
25%	0.300000	0.000000
50%	1.300000	1.000000
75%	1.800000	2.000000
max	2.500000	2.000000

Practical -5

Aim: Reading different types of datasets (.txt,csv) from web and disk and save file in specific disk location.

Program:

```
import pandas as pd

csv_url = "https://raw.githubusercontent.com/cs109/2014_data/master/countries.csv"

df = pd.read_csv(csv_url)

    # Now, 'df' is a DataFrame containing the data from the CSV file

    # You can perform various operations and analysis on this DataFrame

print(df)

# Save the DataFrame to a local CSV file

df.to_csv("local_dataset.csv", index=False)  # Specify the desired file name
```

Output:

```
Country         Region
0      Algeria         AFRICA
1       Angola         AFRICA
2        Benin         AFRICA
3     Botswana          AFRICA
4      Burkina         AFRICA
..         ...            ...
189   Paraguay  SOUTH AMERICA
190       Peru  SOUTH AMERICA
191   Suriname  SOUTH AMERICA
192    Uruguay  SOUTH AMERICA
193  Venezuela  SOUTH AMERICA

[194 rows x 2 columns]
```

Practical -6

Aim: Python Program to implement recursion by using user defined function.

Program:

```
# Function to calculate the factorial of a number using recursion

def factorial(n):

    if n == 0:

        return 1

    else:

        return n * factorial(n - 1)

# Input from the user

num = int(input("Enter a non-negative integer: "))

# Check if the input is non-negative

if num < 0:

    print("Factorial is undefined for negative numbers.")

else:

    result = factorial(num)

    print(f"The factorial of {num} is {result}")
```

Output:

```
Enter a non-negative integer: 3
The factorial of 3 is 6
```

Practical -7

Aim: Python Program to print the table of any number

Program:

```
# Multiplication table (from 1 to 10) in Python

num = int(input("Multiplication table of= "))

# Iterate 10 times from i = 1 to 10

for i in range(1, 11):

   print(num, 'x', i, '=', num*i)
```

Output:

```
Multiplication table of= 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

Practical -8

Aim: Python Program to implement list and it operations.

Program:

```
# Create a list of fruits

fruits = ['apple', 'banana', 'cherry', 'date']

# print List

print(fruits)

# Access and print elements in the list

print("First fruit:", fruits[0])

print("Third fruit:", fruits[2])

# Modify the second element

fruits[1] = 'kiwi'

# Append an element to the end of the list

fruits.append('grape')

# Remove an element from the list

fruits.remove('cherry')

# Find the length of the list

num_fruits = len(fruits)

print("Number of fruits:", num_fruits)
```

Output:

```
['apple', 'banana', 'cherry', 'date']
First fruit: apple
Third fruit: cherry
Number of fruits: 4
```

Practical -9

Aim: Python Program to implement CSR Matrix using Scipy.

Program:

```
import numpy as np

from scipy.sparse import csr_matrix

arr = np.array([0, 0, 0, 0, 0, 1, 1, 0, 2])

print(csr_matrix(arr))
```

Output:

```
(0, 5)  1
(0, 6)  1
(0, 8)  2
```

Practical -10

Aim: Install and perform a numerical array processing using Numpy

Program:

```
import numpy as np

# Creating NumPy arrays

arr1 = np.array([1, 2, 3, 4, 5])

arr2 = np.array([6, 7, 8, 9, 10])


# Element-wise operations

addition = arr1 + arr2

subtraction = arr1 - arr2

multiplication = arr1 * arr2

division = arr1 / arr2


print("Array 1:", arr1)

print("Array 2:", arr2)


print("Addition:", addition)

print("Subtraction:", subtraction)

print("Multiplication:", multiplication)

print("Division:", division)


# Creating a NumPy array using arange

arr3 = np.arange(1, 11)  # Creates an array from 1 to 10

print("Array 3 (using arange):", arr3)


# Reshaping arrays

reshaped_arr = arr3.reshape(2, 5)
```

```
print("Reshaped Array:")

print(reshaped_arr)


# Transposing an array

transposed_arr = reshaped_arr.T

print("Transposed Array:")

print(transposed_arr)


# Basic array statistics

mean_value = np.mean(arr3)

max_value = np.max(arr3)

min_value = np.min(arr3)

sum_value = np.sum(arr3)

print("Mean:", mean_value)

print("Max:", max_value)

print("Min:", min_value)

print("Sum:", sum_value)
```

Output:

```
Array 1: [1 2 3 4 5]
Array 2: [ 6  7  8  9 10]
Addition: [ 7  9 11 13 15]
Subtraction: [-5 -5 -5 -5 -5]
Multiplication: [ 6 14 24 36 50]
Division: [0.16666667 0.28571429 0.375      0.44444444 0.5       ]
Array 3 (using arange): [ 1  2  3  4  5  6  7  8  9 10]
Reshaped Array:
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
Transposed Array:
[[ 1  6]
 [ 2  7]
 [ 3  8]
 [ 4  9]
 [ 5 10]]
Mean: 5.5
Max: 10
Min: 1
Sum: 55
```

Practical -11

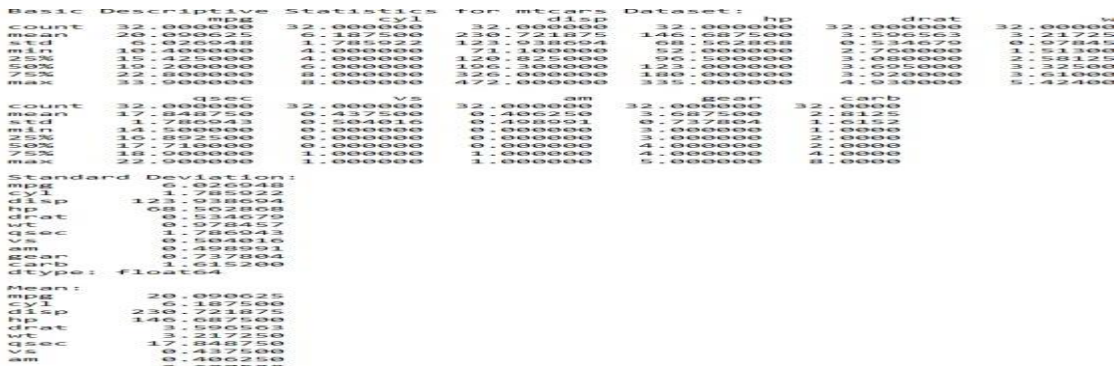
Aim: Write an Python Program script to find all basic descriptive statistics using summary,str,quartile functions on mtcars datasets.

Program:

```
import pandas as pd

# Load the mtcars dataset

mtcars_data=pd.read_csv("https://vincentarelbundock.github.io/Rdatasets/csv/datasets/mtcars.csv")

# Display basic descriptive statistics

def display_basic_stats(data):

    print("Basic Descriptive Statistics for mtcars Dataset:")

  # Describe() function provides summary statistics

    summary_stats = data.describe()

    print(summary_stats)


    # Standard Deviation

    std = data.std()

    print("\nStandard Deviation:")

    print(std)


    # Mean

    mean = data.mean()

    print("\nMean:")

    print(mean)


    # Median (50th percentile)

    median = data.median()
```

```
    print("\nMedian (50th Percentile):")
    print(median)

    # Quartiles (25th, 50th, and 75th percentiles)
    quartiles = data.quantile([0.25, 0.5, 0.75])
    print("\nQuartiles (25th, 50th, and 75th Percentiles):")
    print(quartiles)

    # Variance
    var = data.var()
    print("\nVariance:")
    print(var)

    # Count of non-null values for each column
    count = data.count()
    print("\nCount of Non-null Values:")
    print(count)

# Display all basic statistics for the mtcars dataset
display_basic_stats(mtcars_data)
```

Output:



Practical -12

Aim: Program to find the correlation matrix.

Program:

```
import pandas as pd

# collect data

data = {

        'x': [45, 37, 42, 35, 39],

        'y': [38, 31, 26, 28, 33],

        'z': [10, 15, 17, 21, 12]

}

# form dataframe

dataframe = pd.DataFrame(data, columns=['x', 'y', 'z'])

print("Dataframe is : ")

print(dataframe)

# form correlation matrix

matrix = dataframe.corr()

print("Correlation matrix is : ")

print(matrix)
```

Output:

```
Dataframe is :
   x   y   z
0  45  38  10
1  37  31  15
2  42  26  17
3  35  28  21
4  39  33  12
Correlation matrix is :
          x         y         z
x  1.000000  0.518457 -0.701886
y  0.518457  1.000000 -0.860941
z -0.701886 -0.860941  1.000000
```

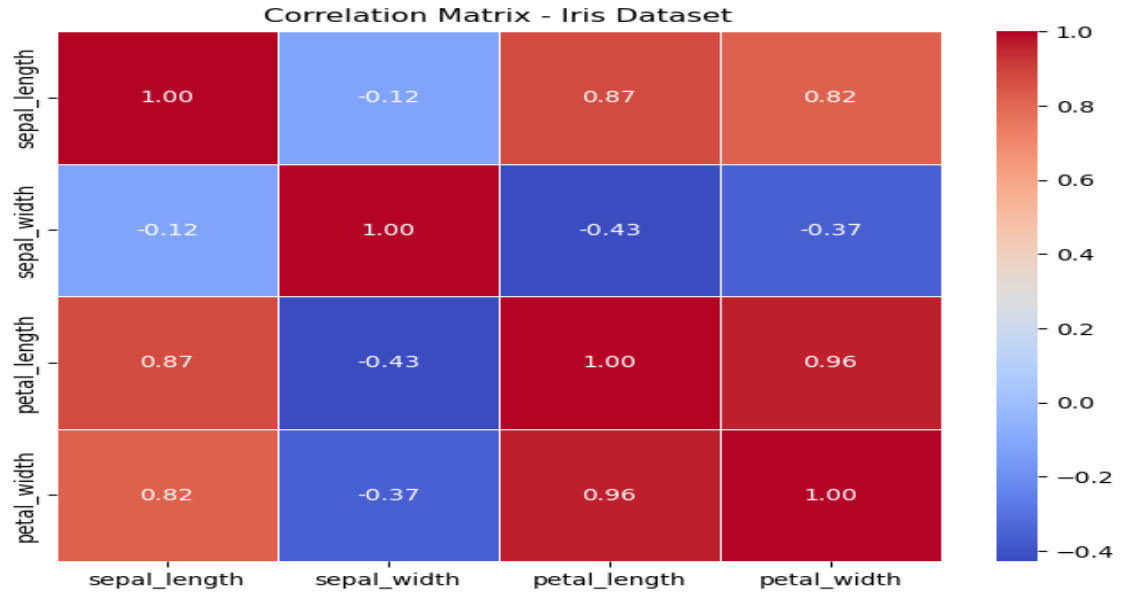
Practical -13

Aim: Plot the correlation plot on dataset and visualize giving an overview of relationships among data on iris dataset.

Program:

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Load the Iris dataset

iris_data = sns.load_dataset("iris")

# Compute the correlation matrix

correlation_matrix = iris_data.corr()

# Create a heatmap to visualize the correlation matrix

plt.figure(figsize=(8, 6))

sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=0.5)

plt.title("Correlation Matrix - Iris Dataset")

plt.show()
```

Output:



Practical -14

Aim: Write a python program to explore a simple dataset using pandas.

Program:

```
import pandas as pd

# Create a sample dataset (you can replace this with your own dataset)

data = {

    'Name': ['Amit', 'Boby', 'Chetan', 'Davesh', 'Isha'],

    'Age': [28, 24, 30, 22, 35],

    'Salary': [50000, 45000, 60000, 40000, 75000]

}

# Convert the dictionary into a Pandas DataFrame

df = pd.DataFrame(data)

# Display the first few rows of the dataset

print("First 5 rows of the dataset:")

print(df.head())

# Basic information about the dataset

print("\nDataset Information:")

print(df.info())

# Summary statistics

print("\nSummary Statistics:")

print(df.describe())

# Check for missing values

print("\nMissing Values:")

print(df.isnull().sum())

# Unique values in a column

unique_names = df['Name'].unique()

print("\nUnique Names:")

print(unique_names)
```

```
# Value counts for a categorical column

age_counts = df['Age'].value_counts()

print("\nAge Value Counts:")

print(age_counts)
```

Output:

```
First 5 rows of the dataset:
     Name  Age  Salary
0    Amit   28   50000
1    Boby   24   45000
2  Chetan   30   60000
3  Davesh   22   40000
4    Isha   35   75000

Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    5 non-null      object
 1   Age     5 non-null      int64
 2   Salary  5 non-null      int64
dtypes: int64(2), object(1)
memory usage: 248.0+ bytes
None

Summary Statistics:
             Age        Salary
count   5.000000      5.000000
mean   27.800000  54000.000000
std     5.118594  13874.436926
min    22.000000  40000.000000
25%    24.000000  45000.000000
50%    28.000000  50000.000000
75%    30.000000  60000.000000
max    35.000000  75000.000000

Missing Values:
Name      0
Age       0
Salary    0
dtype: int64

Unique Names:
['Amit' 'Boby' 'Chetan' 'Davesh' 'Isha']

Age Value Counts:
28    1
24    1
30    1
22    1
35    1
Name: Age, dtype: int64
```

Practical -15

Aim: Install ,import scikit learn and explore Iris dataset with pandas for ML modelling.

Program:

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
# Step 1: Load the Iris dataset
iris = load_iris()
# Step 2: Create a DataFrame from the dataset for exploration
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_df['target'] = iris.target
# Step 3: Explore the dataset (optional)
# Display the first few rows of the dataset
print("First few rows of the Iris dataset:")
print(iris_df.head())
# Check the number of rows and columns
print("\nNumber of rows and columns:")
print(iris_df.shape)
# Check for missing values (Iris dataset typically doesn't have missing values)
print("\nMissing values:")
print(iris_df.isnull().sum())
# Step 4: Prepare data for machine learning
# Split the data into features (X) and target labels (y)
X = iris_df.drop('target', axis=1)
y = iris_df['target']
```

```
# Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 5: Train a machine learning model (SVM classifier in this case)

svm_classifier = SVC(kernel='linear', C=1)  # You can adjust the model hyperparameters

# Fit the model on the training data

svm_classifier.fit(X_train, y_train)

# Step 6: Evaluate the model

# Make predictions on the test set

y_pred = svm_classifier.predict(X_test)

# Calculate accuracy

accuracy = accuracy_score(y_test, y_pred)

print("\nAccuracy:", accuracy)

# Display classification report

classification_rep = classification_report(y_test, y_pred, target_names=iris.target_names)

print("\nClassification Report:\n", classification_rep)

# Display confusion matrix

conf_matrix = confusion_matrix(y_test, y_pred)

print("\nConfusion Matrix:\n", conf_matrix)
```

Output:

```
First few rows of the Iris dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0

Number of rows and columns:
(150, 5)

Missing values:
```

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
target               0
dtype: int64

Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30


Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Practical -16

Aim: Python Program to find the outliers using plot.

```
import numpy as np
from scipy import stats

# Sample data (replace with your dataset)
data = [15, 18, 20, 22, 24, 30, 45, 50, 65, 800, 120, 130, 140]

# Calculate the Z-scores for each data point
z_scores = np.abs(stats.zscore(data))

# Set a Z-score threshold for identifying outliers (e.g., threshold of 2.0)
threshold = 2.0

# Find and print outliers
outliers = [data[i] for i in range(len(z_scores)) if z_scores[i] > threshold]
print("Outliers:", outliers)
```

Output:

```
Outliers: [800]
```

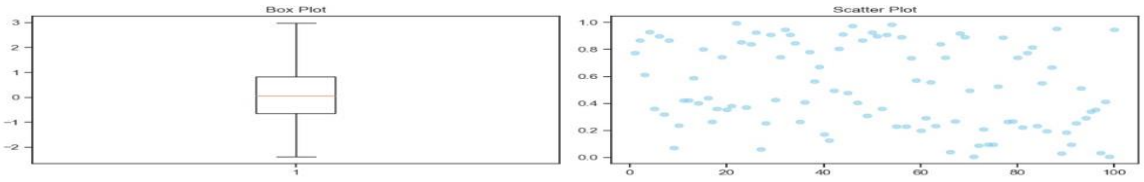
Practical -17

Aim: Find the data distribution using box and scatterplot.

Program:

```
import matplotlib.pyplot as plt

import numpy as np

# Sample data for the box plot

box_data = np.random.normal(0, 1, 100)  # Generating 100 random data points with a normal distribution

# Sample data for the scatter plot

scatter_x = np.arange(1, 101)  # Generate x values (1 to 100)

scatter_y = np.random.rand(100)  # Generate random y values

# Create a figure with subplots

plt.figure(figsize=(12, 4))

# Box Plot

plt.subplot(1, 2, 1)

plt.boxplot(box_data)

plt.title('Box Plot')

# Scatter Plot

plt.subplot(1, 2, 2)

plt.scatter(scatter_x, scatter_y, color='skyblue', alpha=0.6)

plt.title('Scatter Plot')

plt.tight_layout()

plt.show()
```

Output:



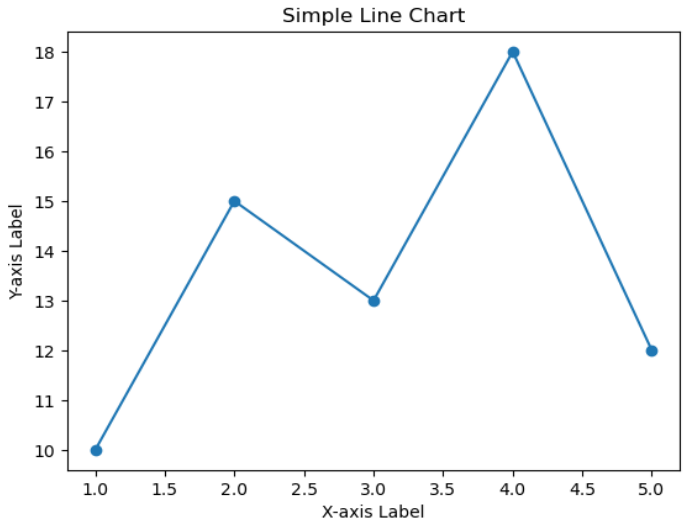
Practical -18

Aim: Write a python Program for Line Chart.

Program:

```
import matplotlib.pyplot as plt

# Sample data

x = [1, 2, 3, 4, 5]

y = [10, 15, 13, 18, 12]

# Create a line chart

plt.plot(x, y, marker='o', linestyle='-')

# Add labels and a title

plt.xlabel('X-axis Label')

plt.ylabel('Y-axis Label')

plt.title('Simple Line Chart')

# Show the chart

plt.show()
```

Output:



Practical -19

Aim: Write a python Program for Pie Chart.

Program:

```
# Pie Chart

import matplotlib.pyplot as plt

import numpy as np

y = np.array([35, 25, 25, 15])

plt.pie(y)

plt.show()
```

Output:

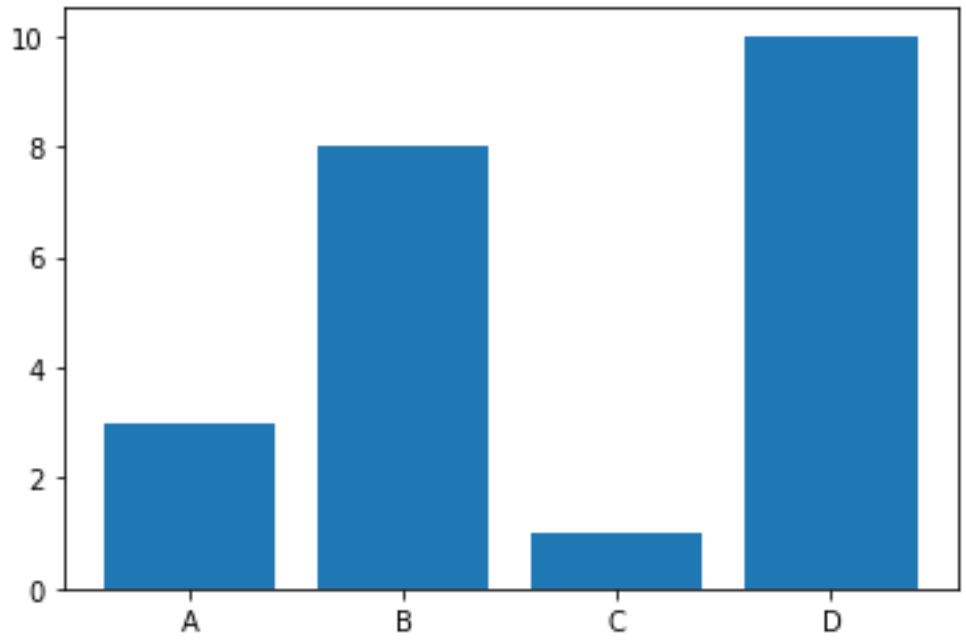
Practical -20

Aim: Write a python Program for Bar Graph.

Program:

```
# Bar Graph

import matplotlib.pyplot as plt

import numpy as np


x = np.array(["A", "B", "C", "D"])

y = np.array([3, 8, 1, 10])


plt.bar(x,y)

plt.show()
```

Output:



Practical –21

Aim: Python program for customizing plots.

Program:

```
import matplotlib.pyplot as plt

import numpy as np

# Sample data

x = np.random.rand(50)  # 50 random x-values

y = np.random.rand(50)  # 50 random y-values

colors = np.random.rand(50)  # Random colors for each point

sizes = np.random.randint(10, 100, 50)  # Random sizes for each point

# Create the scatter plot

plt.scatter(x, y, c=colors, s=sizes, alpha=0.7, cmap='viridis')

# Customize the plot

plt.title('Customized Scatter Plot')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.colorbar(label='Color Intensity')

plt.grid(True)


# Show the plot

plt.show()
```

Output: