# Experiment 5

## Program No.: 5.1

**Aim:** Create a series of plots to analyze a given dataset.

**Software used:** Colab (or) Jupyter Notebook

**Description:** This program demonstrates the creation of multiple plots (sine, cosine, and product) using Matplotlib subplots for comparative visualization.

**SOURCE CODE:**

import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.linspace(0, 2 \* np.pi, 100)  
y = np.sin(x)  
y2 = np.cos(x)  
y3 = np.sin(x) \* np.cos(x)  
  
plt.figure(figsize=(12, 4))  
  
plt.subplot(1, 3, 1)  
plt.plot(x, y, color='blue')  
plt.title('Sine Wave')  
plt.xlabel('X-axis')  
plt.ylabel('Y-axis')  
  
plt.subplot(1, 3, 2)  
plt.plot(x, y2, color='green')  
plt.title('Cosine Wave')  
plt.xlabel('X-axis')  
plt.ylabel('Y-axis')  
  
plt.subplot(1, 3, 3)  
plt.plot(x, y3, color='red')  
plt.title('Product of Sine and Cosine')  
plt.xlabel('X-axis')  
plt.ylabel('Y-axis')  
  
plt.tight\_layout()  
plt.show()

PROCEDURE:  
1. Import Matplotlib and NumPy.  
2. Create sine, cosine, and product data.  
3. Create subplots for each function.  
4. Add titles and axis labels.  
5. Display all plots.

INPUT:  
Predefined mathematical functions.

EXPECTED / ACTUAL OUTPUT:  
Three subplots displaying sine, cosine, and their product.

## Program No.: 5.2

Aim: Generate a subplot layout with various plot types (scatter, line, bar).

Software used: Colab (or) Jupyter Notebook

Description: Demonstrates use of multiple plot types in a 2x2 grid layout using Matplotlib and Pandas.

SOURCE CODE:

import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
  
data = {'Category': ['A', 'B', 'C', 'D'],  
 'Value1': [10, 25, 15, 30],  
 'Value2': [15, 20, 25, 10]}  
df = pd.DataFrame(data)  
  
x\_scatter = np.random.rand(50)  
y\_scatter = np.random.rand(50)  
x\_line = np.arange(10)  
y\_line = np.random.randint(1, 10, size=10)  
  
fig, axes = plt.subplots(2, 2, figsize=(10, 8))  
  
axes[0, 0].scatter(x\_scatter, y\_scatter)  
axes[0, 0].set\_title('Scatter Plot')  
axes[0, 0].set\_xlabel('X-axis')  
axes[0, 0].set\_ylabel('Y-axis')  
  
axes[0, 1].plot(x\_line, y\_line, marker='o')  
axes[0, 1].set\_title('Line Plot')  
axes[0, 1].set\_xlabel('X-axis')  
axes[0, 1].set\_ylabel('Y-axis')  
  
axes[1, 0].bar(df['Category'], df['Value1'])  
axes[1, 0].set\_title('Bar Chart')  
axes[1, 0].set\_xlabel('Category')  
axes[1, 0].set\_ylabel('Value1')  
  
axes[1, 1].set\_visible(False)  
plt.tight\_layout()  
plt.show()

PROCEDURE:  
1. Import Matplotlib, NumPy, and Pandas.  
2. Create sample datasets.  
3. Create a 2x2 subplot grid.  
4. Plot scatter, line, and bar charts.  
5. Adjust layout and display plots.

INPUT:  
Randomly generated and predefined data.

EXPECTED / ACTUAL OUTPUT:  
Scatter plot, line plot, and bar chart displayed in a 2x2 grid.

## Program No.: 5.3

Aim: Visualize time-series data and customize axis labels and date formats.

Software used: Colab (or) Jupyter Notebook

Description: Demonstrates time-series visualization using Pandas and Matplotlib with customized date formatting.

SOURCE CODE:

import matplotlib.pyplot as plt  
import pandas as pd  
import numpy as np  
  
dates = pd.to\_datetime(pd.date\_range(start='2024-01-01', periods=100, freq='D'))  
sales\_data = np.random.randint(100, 500, size=100) + np.sin(np.arange(100) \* 0.2) \* 50  
df = pd.DataFrame({'Date': dates, 'Sales': sales\_data})  
df = df.set\_index('Date')  
  
plt.figure(figsize=(12, 6))  
plt.plot(df.index, df['Sales'])  
plt.xlabel('Date')  
plt.ylabel('Sales')  
plt.title('Daily Sales Over Time')  
  
df.plot(y='Sales', figsize=(12, 6), title='Daily Sales Over Time')  
plt.show()

PROCEDURE:  
1. Import Pandas, NumPy, and Matplotlib.  
2. Generate time-series data.  
3. Plot sales data over time.  
4. Customize axis labels and titles.  
5. Display time-series plot.

INPUT:  
Predefined simulated sales data with dates.

EXPECTED / ACTUAL OUTPUT:  
Time-series plot showing daily sales variation with proper date formatting.

## Program No.: 5.4

Aim: Create a 3D plot.

Software used: Colab (or) Jupyter Notebook

Description: Demonstrates creation of a 3D scatter plot using Matplotlib’s 3D plotting toolkit.

SOURCE CODE:

import matplotlib.pyplot as plt  
import numpy as np  
from mpl\_toolkits.mplot3d import Axes3D  
  
np.random.seed(42)  
x = np.random.rand(50)  
y = np.random.rand(50)  
z = np.random.rand(50)  
  
fig = plt.figure(figsize=(8, 6))  
ax = fig.add\_subplot(111, projection='3d')  
  
ax.scatter(x, y, z)  
ax.set\_xlabel('X-axis')  
ax.set\_ylabel('Y-axis')  
ax.set\_zlabel('Z-axis')  
ax.set\_title('3D Scatter Plot')  
  
plt.show()

PROCEDURE:  
1. Import Matplotlib and NumPy.  
2. Generate 3D random data.  
3. Create a 3D figure.  
4. Plot data and label axes.  
5. Display 3D scatter plot.

INPUT:  
Randomly generated x, y, z coordinates.

EXPECTED / ACTUAL OUTPUT:  
A 3D scatter plot with labeled axes.