Experiment - 5.1.

AIM : Create a series of plots to analyze a given data set.

DESCRIPTION:

Matplotlib is a powerful and widely used data visualization library in Python.
It allows users to create a wide range of static, animated, and interactive plots easily.

- pyplot is a submodule of Matplotlib that provides a simple interface similar to MATLAB.
- It helps create plots using functions like:
    - plt.plot() – Line plot
    - plt.scatter() – Scatter plot
    - plt.bar() – Bar chart
    - plt.hist() – Histogram
    - plt.title(), plt.xlabel(), plt.ylabel() – For titles and labels

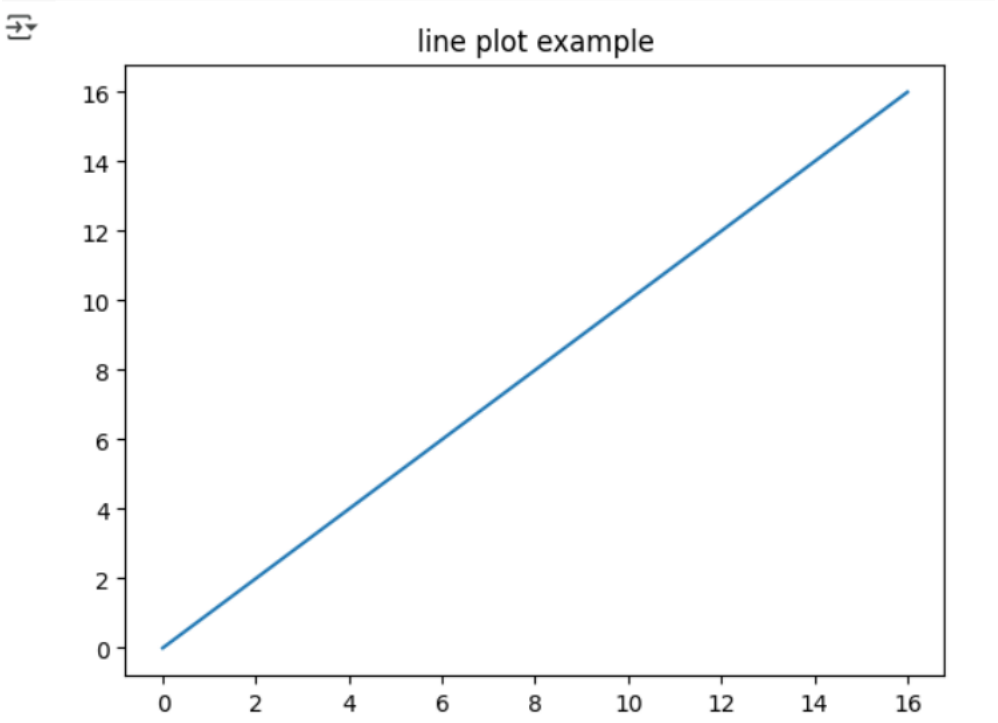
Explanation of Each Plot:

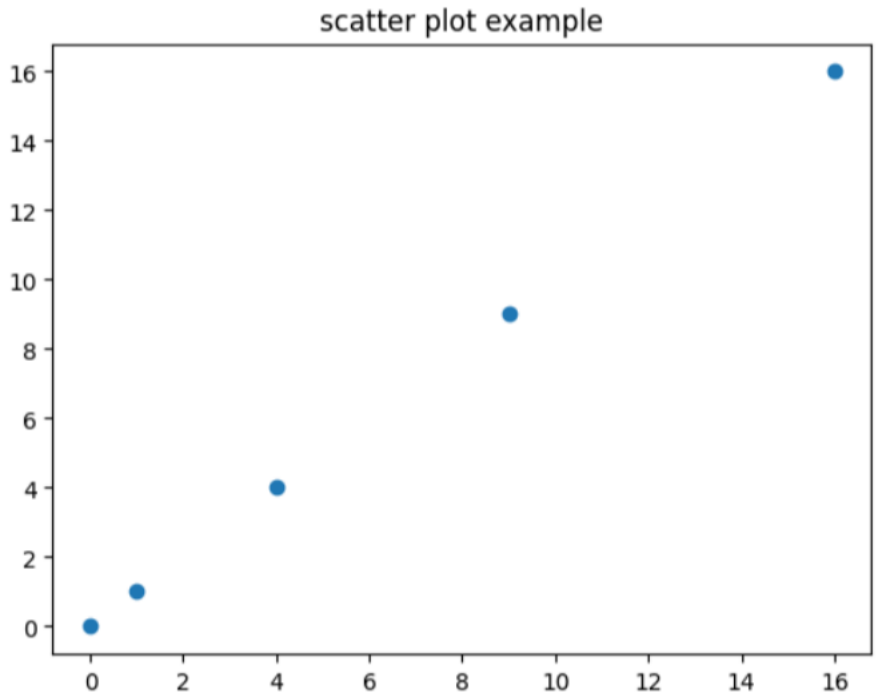
1. **Line Plot**
    - Shows a continuous straight line because y = x.
    - Demonstrates a linear relationship between x and y.
    - Useful for showing trends or relationships between numeric variables.
2. **Scatter Plot**
    - Displays individual data points without connecting lines.
    - Used to observe how two variables relate to each other.
    - Here, the points fall on a straight line, confirming the direct linear relationship.
3. **Bar Chart**
    - Represents the same data using rectangular bars.
    - Each bar's height corresponds to the value of y for a given x.
    - Highlights differences or magnitudes between the data values visually.

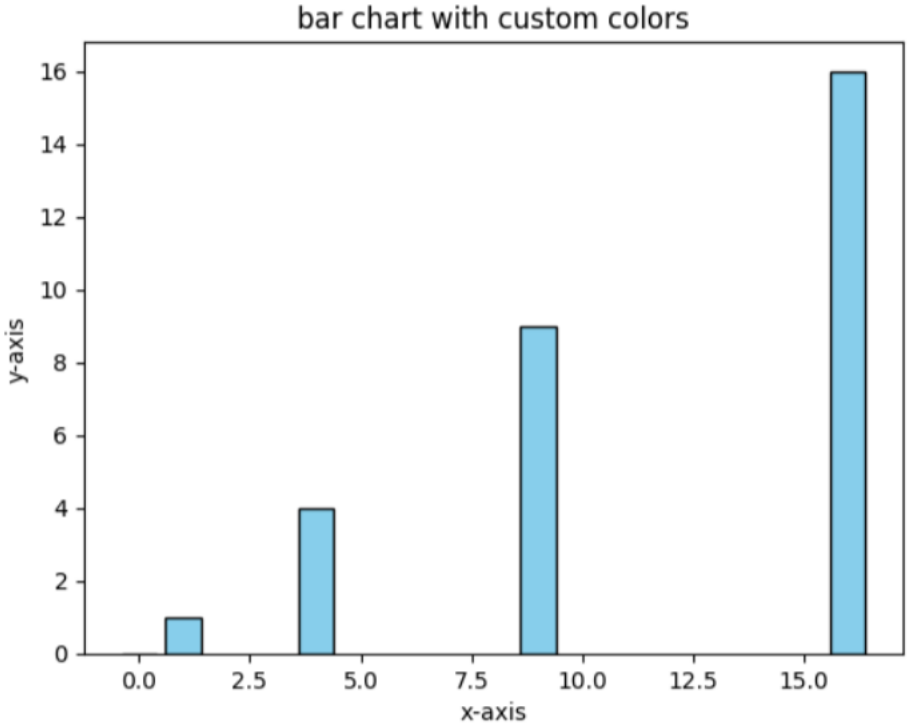
PROGRAM:

```
import matplotlib.pyplot as plt
import numpy as np
x=[0,1,4,9,16]
y=[0,1,4,9,16]
plt.plot(x,y)
plt.title("line plot example")
plt.show()
plt.scatter(x,y)
plt.title("scatter plot example")
plt.show()
plt.bar(x,y,color='skyblue',edgecolor='black')
plt.title("bar chart with custom colors")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.show()
```

OUTPUT:







RESULT: Hence to create a series of plots to analyze a given data set.has been executed and its output is verified.

Experiment - 5.2

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

DESCRIPTION:

A subplot layout in Matplotlib allows you to display multiple plots in a single figure window, making it easier to compare different types of data visualizations side by side.
It helps in analyzing the same dataset or different datasets from multiple perspectives.

What is a Subplot Layout?

- A subplot layout divides the plotting area into a grid of rows and columns, where each cell of the grid can hold a separate plot.
- The function used is:
- fig, axs = plt.subplots(rows, columns)
    - fig represents the overall figure (canvas).
    - axs is an array of Axes objects, each representing one subplot.

Different Plot Types Used

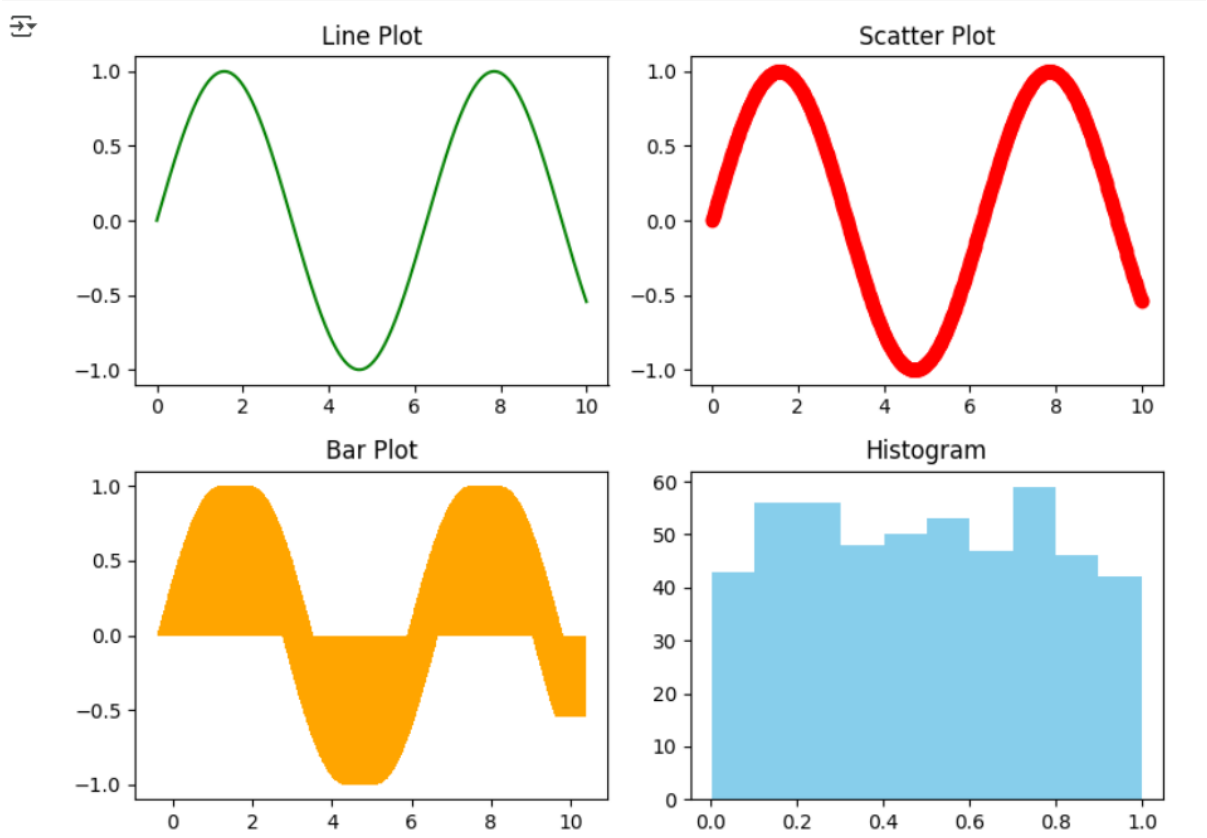
1. **Line Plot**
    - Displays data points connected by straight lines.
    - Useful for showing trends and continuous changes over a range.
2. **Scatter Plot**
    - Displays individual data points using dots.
    - Helps visualize relationships or patterns between two variables.
3. **Bar Chart**
    - Represents data using rectangular bars where the height indicates the value.
    - Useful for comparing quantities among categories.

PROGRAM:

```
import numpy as np

import matplotlib.pyplot as plt

x = np.linspace(0, 10, 1000)

y = np.sin(x)

fig, axs = plt.subplots(2, 2, figsize=(8, 6))
```

```
axs[0, 0].plot(x,y, color='green')
axs[0,0].set_title("Line Plot")
axs[0, 1].scatter(x,y, color='red')
axs[0, 1].set_title("Scatter Plot")
axs[1, 0].bar(x,y, color='orange')
axs[1, 0].set_title("Bar Plot")
data = np.random.rand(500)
axs[1, 1].hist(data, bins=10, color='skyblue')
axs[1, 1].set_title("Histogram")
plt.tight_layout()
plt.show()
```

OUTPUT:



RESULT: Hence to generate a subplot layout with various plot types (scatter,line,bar) has been executed and its output is verified.

Experiment - 5.3

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

DESCRIPTION:

What is Time-Series Data?

Time-series data is a sequence of data points collected or recorded at regular time intervals — such as daily temperatures, monthly sales, or stock prices over time.
It helps in analyzing trends, patterns, and seasonal variations over a period.

Examples:

- Daily stock prices
- Hourly temperature reading

Visualizing Time-Series Data with Matplotlib

Matplotlib provides powerful tools to plot and format time-series data efficiently.
You can plot data against time (dates or timestamps) using the plot() function, and then customize the x-axis to display properly formatted date labels

Formatting Date Labels

To make date values on the x-axis more readable, Matplotlib provides DateFormatter and AutoDateLocator from the matplotlib.dates module.

Advantages

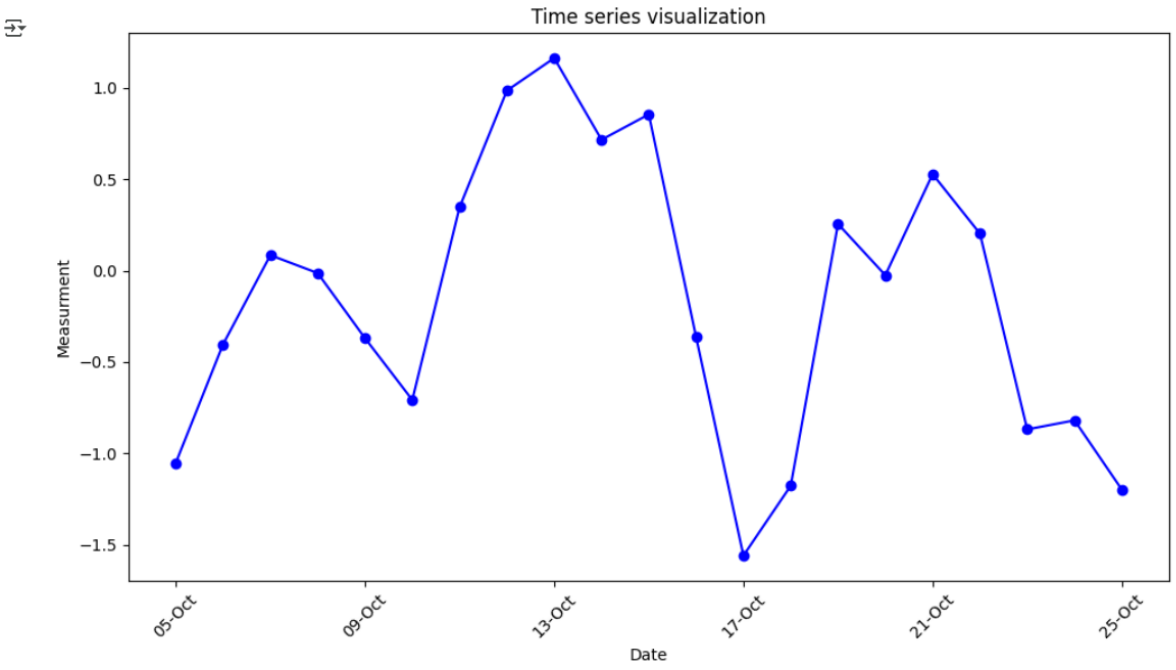
- Makes temporal data easy to understand.
- Helps identify trends, cycles, and anomalies over time.
- Proper axis labeling and date formatting enhance readability and presentation quality.

PROGRAM:

```
#5.3 Visualize time-series data and customize axis labels and date formats.

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import matplotlib.dates as mdates

#simulate time-series data

date_rng=pd.date_range(start='2025-10-05',end='2025-10-25',freq='D')

data=np.random.randn(len(date_rng))
```

```
fig,axs=plt.subplots(figsize=(10,6))
axs.plot(date_rng,data,marker='o',linestyle='-',color='blue')
#label axis and use custom date format
axs.set_xlabel("Date")
axs.set_ylabel("Measurment")
axs.set_title("Time series visualization")
axs.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

OUTPUT:



RESULT: Hence to visualize time-series data and customize axis labels and date formats has been executed and its output is verified.

Experiment - 5.4

AIM : Create a 3D plot

DESCRIPTION:

What is a 3D Plot?

A 3D plot is a graphical representation of data in three dimensions — the X, Y, and Z axes. It helps visualize relationships between three variables and understand data in a more spatial and realistic way compared to 2D plots.

Example uses:

- Visualizing mathematical surfaces
- Representing geographical elevation

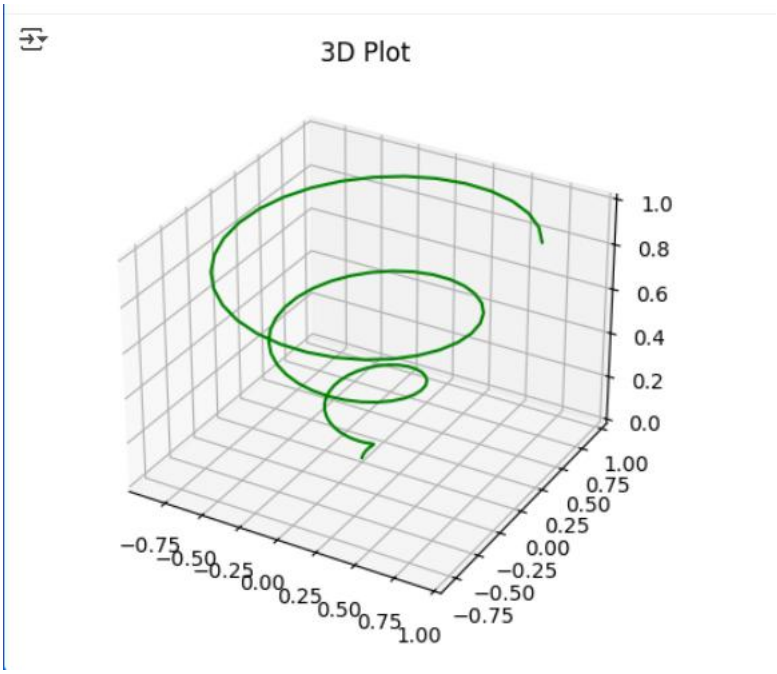
Creating a 3D Plot using Matplotlib

Matplotlib provides a toolkit called mplot3d for creating 3D visualizations. To enable 3D plotting, you must first import the 3D projection toolkit.

PROGRAM:

```
#5.4 Create a 3D plot.

import matplotlib.pyplot as plt

import numpy as np

from mpl_toolkits.mplot3d import Axes3D

fig=plt.figure()

ax=fig.add_subplot(111,projection='3d')

z=np.linspace(0,1,100)

x=z*np.sin(20*z)

y=z*np.cos(20*z)

ax.plot(x,y,z,color='green')

ax.set_title('3D Plot')

plt.show()
```

OUTPUT:



RESULT: Hence to Create a 3D plot has been executed and its output is verified.