Experiment 5.1

Aim:

To Create a series of plots to analyze a given dataset

Description :

The dataset shows monthly sales from January to June, revealing sales trends over time. The line plot highlights growth and fluctuations across months. The bar chart compares sales volumes clearly for each month. Together, these visuals provide a comprehensive view of sales performance.

Procedure / Algorithm :

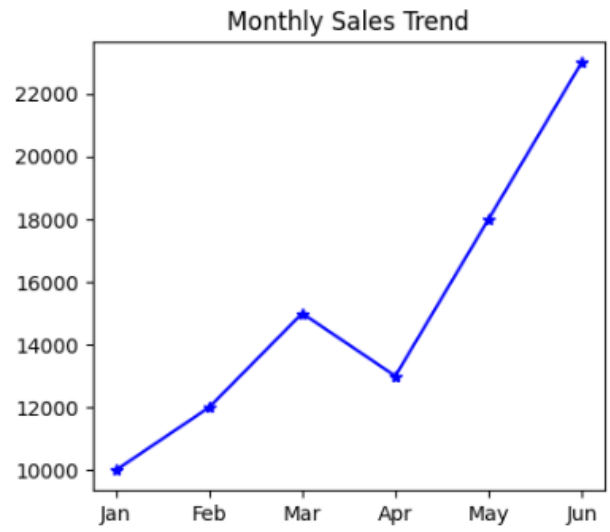
1. Start the program.
2. Prepare data arrays for months and sales.
3. Use Matplotlib to create a figure with two subplots arranged horizontally.
4. Plot sales data as a line plot on the first subplot for trend visualization.
5. Plot sales data as a bar chart on the second subplot for direct value comparison.
6. Add titles, labels, and grid for better readability.
7. End the program

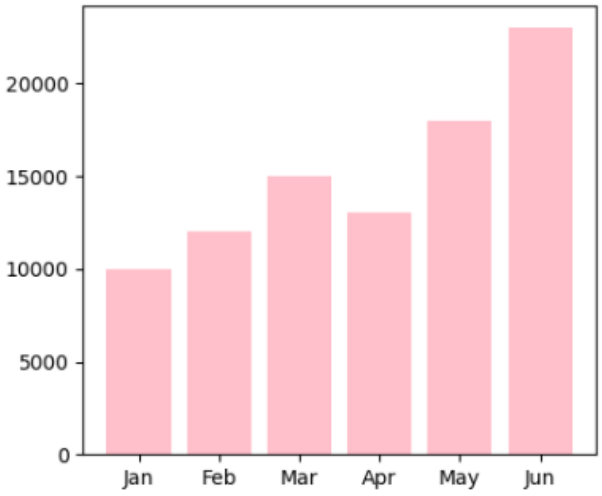
Source Code:

```
import matplotlib.pyplot as plt

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun']

sales = [10000, 12000, 15000, 13000, 18000, 23000]

plt.figure(figsize=(10, 4))

plt.subplot(1, 2, 1)

plt.plot(months, sales, color='blue', marker='*')

plt.title("Monthly Sales Trend")

plt.subplot(1, 2, 2)

plt.bar(months, sales, color='pink')
```

Input and Output :

```
<BarContainer object of 6 artists>
```





Result : A series of plots is generated to analyze the dataset effectively

Experiment 5.2

Aim:

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

Description :

This script demonstrates how to create a subplot layout in Matplotlib featuring four different types of plots: a line plot, scatter plot, vertical bar chart, and horizontal bar chart. The plots are arranged in a 2x2 grid for easy comparison.

Procedure / Algorithm:

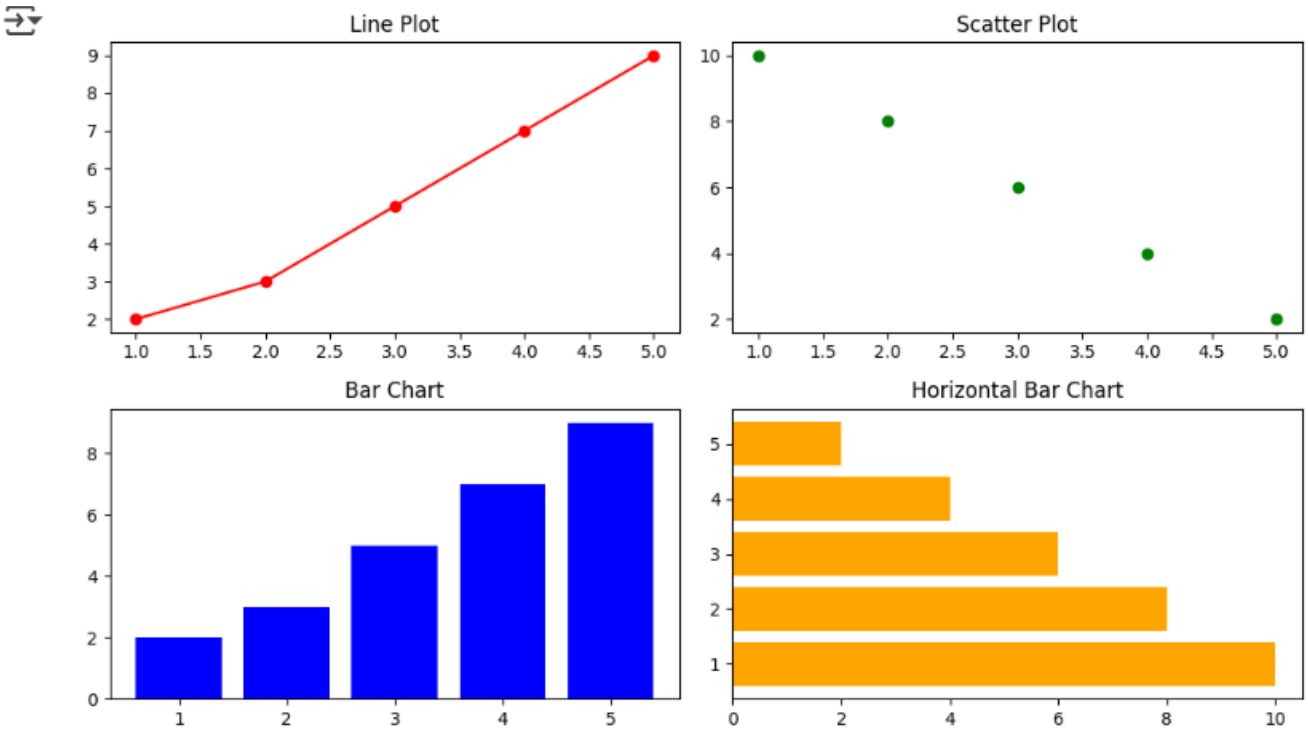
1. Start the program
2. Import the matplotlib.pyplot module as plt to enable plotting.
3. Create lists x, y1, and y2 to hold the data points.
4. Initialize the figure with a specified size using plt.figure().
5. Add Subplots: Line plot using plt.plot(,Scatter plot using plt.scatter().
6. Vertical bar chart using plt.bar(). Horizontal bar chart using plt.barh().
7. Give each subplot a descriptive title.
8. Use plt.tight_layout() to prevent overlapping of subplots.
9. Use plt.show() to render the figure with all subplots.
10. End the program

Source Code:

```
import matplotlib.pyplot as plt

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

y1 = [2, 3, 5, 7, 9]

y2 = [10, 8, 6, 4, 2]

plt.figure(figsize=(10, 6))

plt.subplot(2, 2, 1)

plt.plot(x, y1, 'r-o')

plt.title("Line Plot")
```

```
plt.subplot(2, 2, 2)
plt.scatter(x, y2, color='green')
plt.title("Scatter Plot")
plt.subplot(2, 2, 3)
plt.bar(x, y1, color='blue')
plt.title("Bar Chart")
plt.subplot(2, 2, 4)
plt.barh(x, y2, color='orange')
plt.title("Horizontal Bar Chart")
plt.tight_layout()
plt.show()
```

Input and Output :



Result : A subplot layout with various plot types scatter, line, bar.is generated correctly

Experiment 5.3

Aim:

Visualize time-series data and customize axis labels and date formats

Description

This program generates a time-series line plot using a sequence of dates and corresponding values. It customizes the axis labels and formats the date ticks for better readability.

Procedure / Algorithm

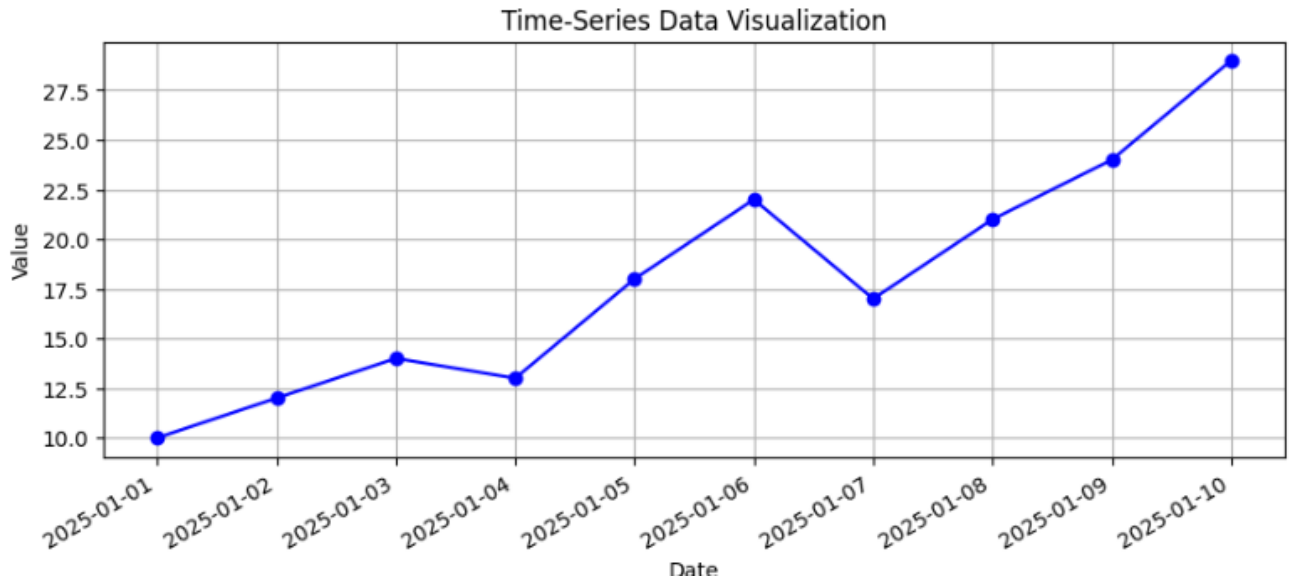
1. Start the program
2. Import matplotlib.pyplot for plotting and pandas for date handling.
3. Use pd.date_range() to create 10 consecutive dates starting from January 1, 2025.
4. Create a list of numerical values corresponding to each date.
5. Set the figure size using plt.figure(figsize=(10, 4)).
6. Use plt.plot() to create a line plot with markers and a blue line.
7. : Set the plot title and axis labels using plt.title(), plt.xlabel(), and plt.ylabel().
8. Add a grid to the plot using plt.grid(True).
9. Automatically rotate and format date labels using plt.gcf().autofmt_xdate().
10. :Show the plot using plt.show().
11. End the program

Source Code:

```
import matplotlib.pyplot as plt

import pandas as pd

dates = pd.date_range('2025-01-01', periods=10)

values = [10, 12, 14, 13, 18, 22, 17, 21, 24, 29]

plt.figure(figsize=(10, 4))

plt.plot(dates, values, marker='o', linestyle='-', color='blue')
```

```
plt.title("Time-Series Data Visualization")
plt.xlabel("Date")
plt.ylabel("Value")
plt.grid(True)
plt.gcf().autofmt_xdate()  # Auto-format date labels
plt.show()
```

Input and Output:



Result : Time-series data is visualized with customized axis labels and formatted date ticks

Experiment 5.4

Aim :

To Create a 3D plot.

Description :

This program generates a 3D surface plot of the function Z = \sin(\sqrt{X^2 + Y^2}) over a meshgrid of X and Y values. It uses the viridis colormap and labels all three axes for clarity.

Procedure / Algorithm :

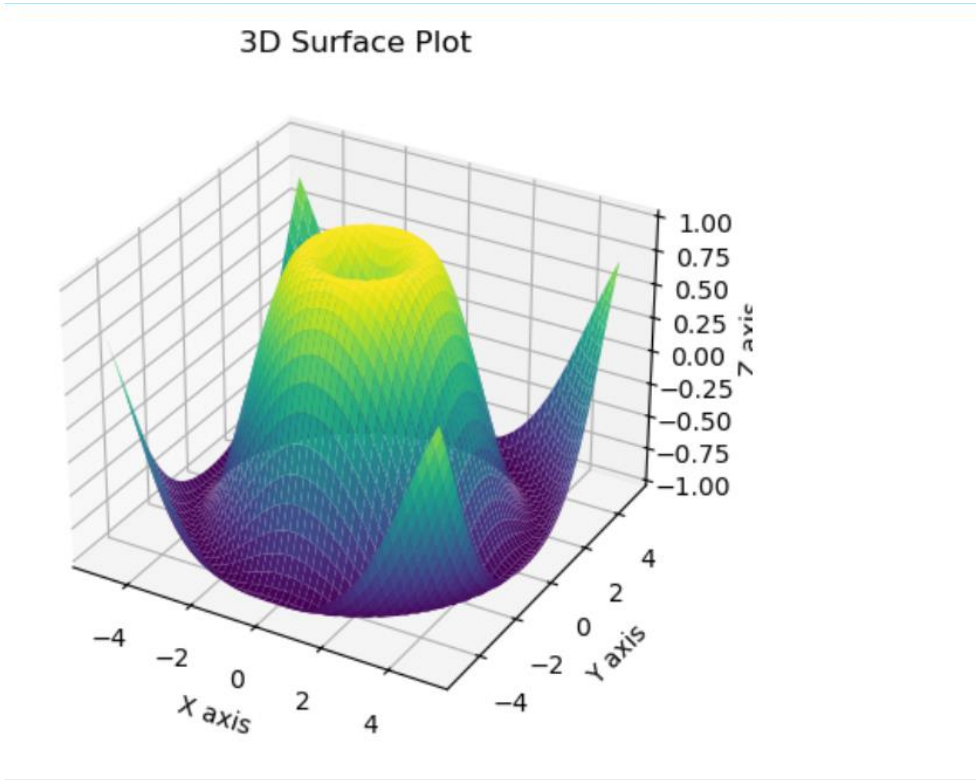
1. Start the program
2. Import matplotlib.pyplot, mpl_toolkits.mplot3d for 3D plotting, and numpy for numerical operations.
3. Initialize a figure using plt.figure().
4. Use fig.add_subplot() with projection='3d' to enable 3D plotting.
5. Generate data:
6. Create linearly spaced arrays x and y using np.linspace().
7. Generate meshgrid X, Y using np.meshgrid(x, y).
8. Compute Z as \sin(\sqrt{X^2 + Y^2}).
9. Use ax.plot_surface() with the viridis colormap to render the 3D surface.
10. Customize plot:
11. Set plot title using ax.set_title().
12. Label axes using ax.set_xlabel(), ax.set_ylabel(), and ax.set_zlabel().
13. Display plot: Use plt.show() to render the figure.
14. End the program

Source Code:

```
import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

import numpy as np
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
x = np.linspace(-5, 5, 50)
y = np.linspace(-5, 5, 50)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))
ax.plot_surface(X, Y, Z, cmap='viridis')
ax.set_title("3D Surface Plot")
ax.set_xlabel("X axis")
ax.set_ylabel("Y axis")
ax.set_zlabel("Z axis")
plt.show()
```

Input and Outpur :



Result : A 3D plot is generated to visualize data across three dimensions.