Matplotlib Case Study

1. Importing Matplotlib

Import the required libraries for plotting and numerical operations.

In [76]:

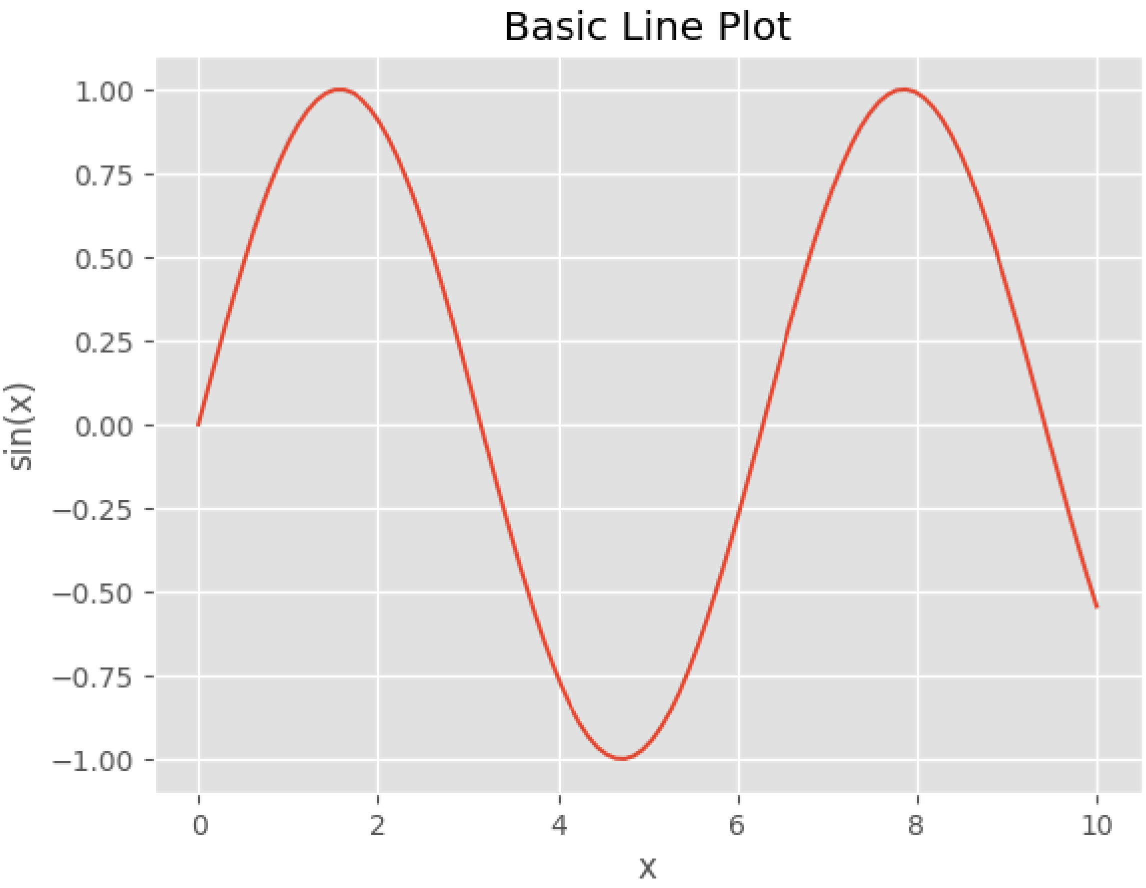
```python
# Import matplotlib for plotting and numpy for numerical operations
import matplotlib.pyplot as plt
import numpy as np
```

2. Basic Line Plot

Create a simple line plot of sin(x) over a range of x values.

In [77]:

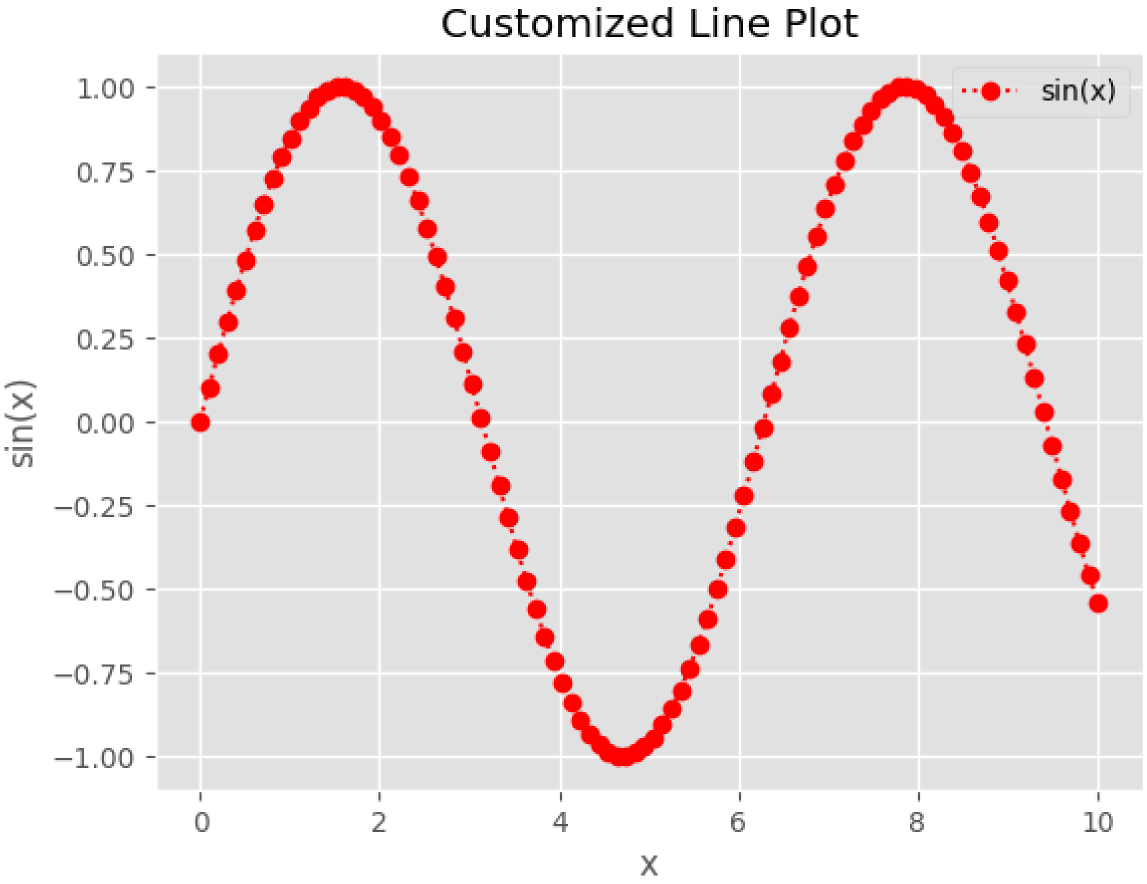
```python
# Generate x values and compute y as sin(x)
x = np.linspace(0, 10, 100)
y = np.sin(x)
# Plot the line graph for sin(x)
plt.plot(x, y)
plt.title('Basic Line Plot')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.show()
```



3. Customizing Plots (Colors, Linestyles, Markers)

Demonstrate how to customize line color, style, and markers.

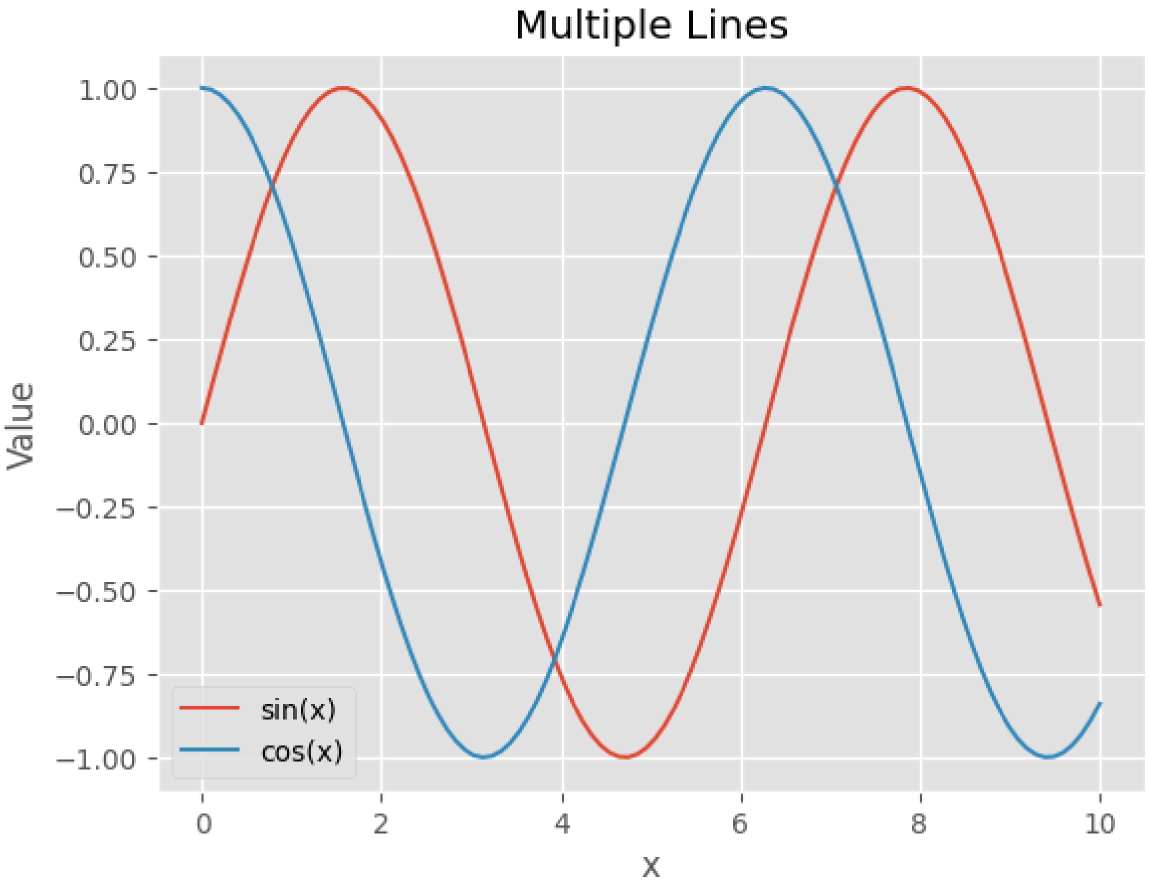
```
In [78]: # Plot with custom color, linestyle, and marker
plt.plot(x, y, color='red', linestyle=':', marker='o', label='sin(x)')
plt.title('Customized Line Plot')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.legend()
plt.show()
```



4. Multiple Lines in One Plot

Plot both sin(x) and cos(x) on the same axes for comparison.

```
In [79]: # Compute y2 as cos(x) and plot both sin(x) and cos(x)
y2 = np.cos(x)
plt.plot(x, y, label='sin(x)')
plt.plot(x, y2, label='cos(x)')
plt.title('Multiple Lines')
plt.xlabel('x')
plt.ylabel('Value')
plt.legend()
plt.show()
```

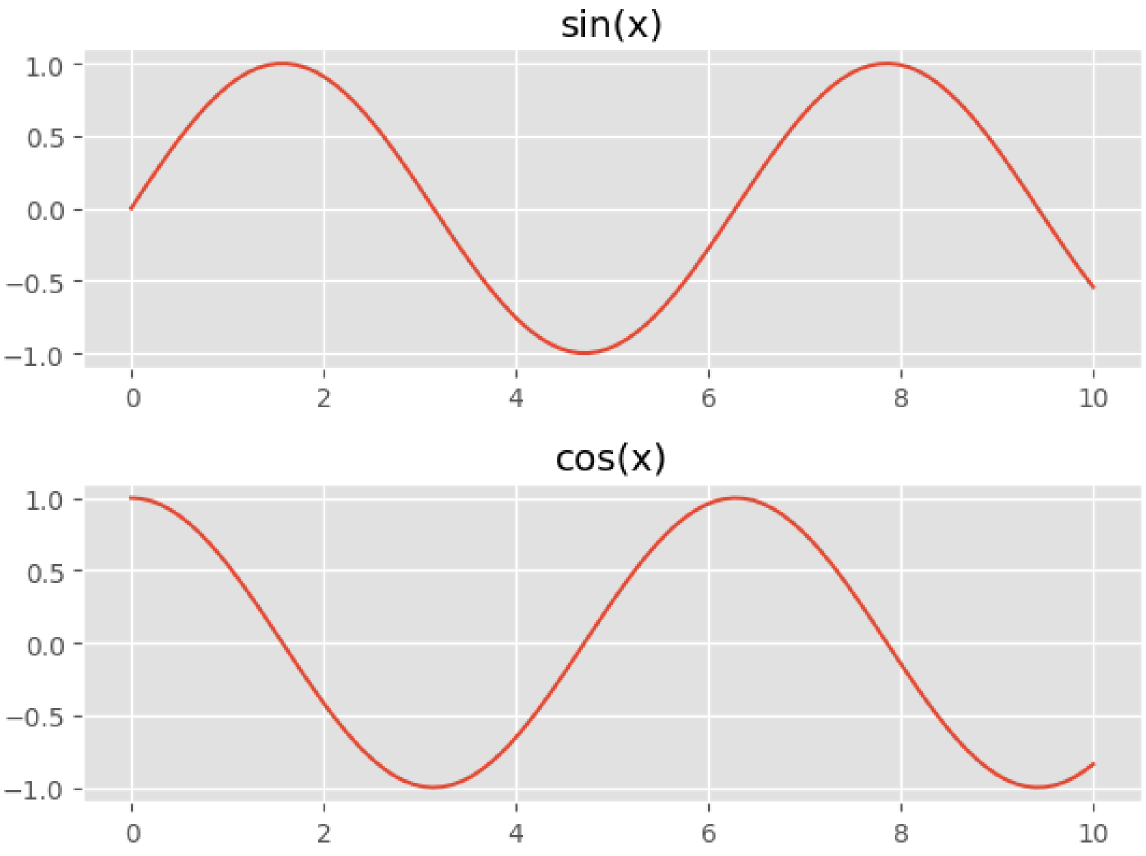


5. Subplots

Show how to create multiple subplots in a single figure.

In [80]:

```
# Create two vertically stacked subplots: one for sin(x), one for cos(x)
fig, axs = plt.subplots(2, 1)
axs[0].plot(x, y)
axs[0].set_title('sin(x)')
axs[1].plot(x, y2, label='cos(x)')
axs[1].set_title('cos(x)')
plt.tight_layout()
plt.show()
```

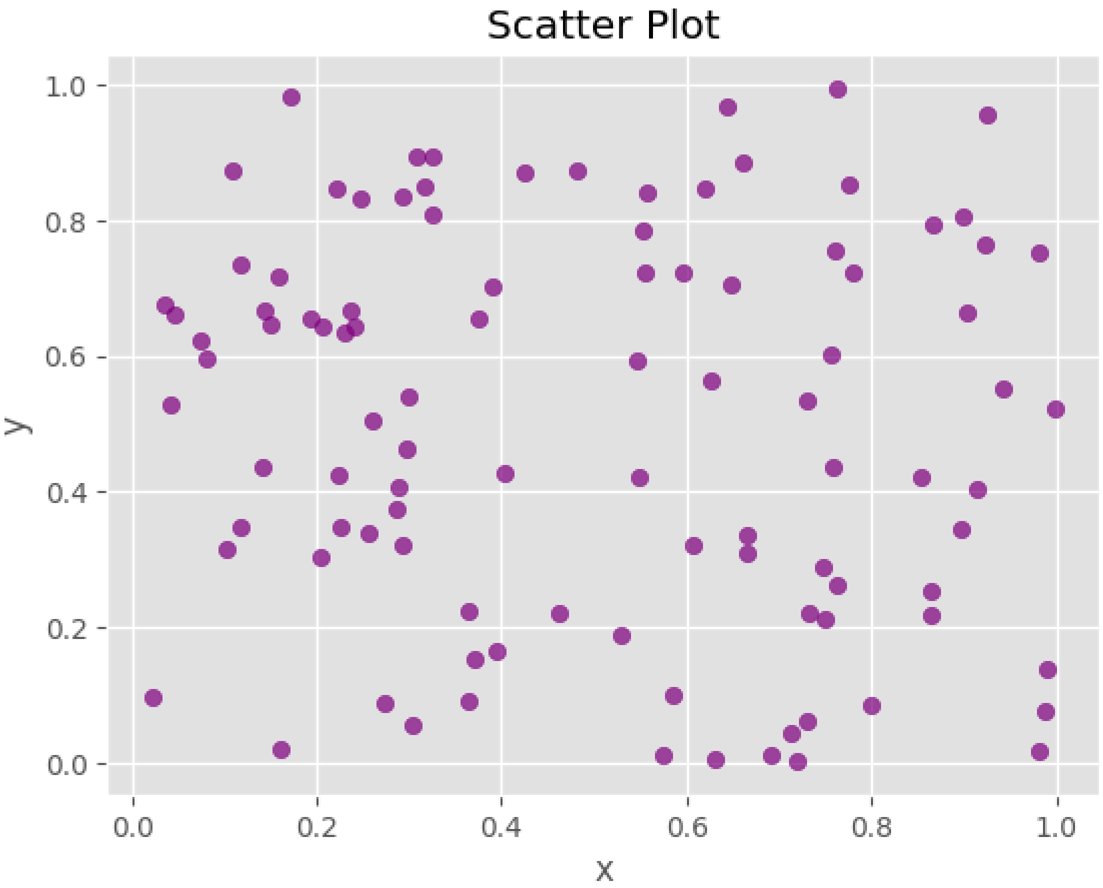


6. Scatter Plot

Visualize the relationship between two random variables using a scatter plot.

In [81]:

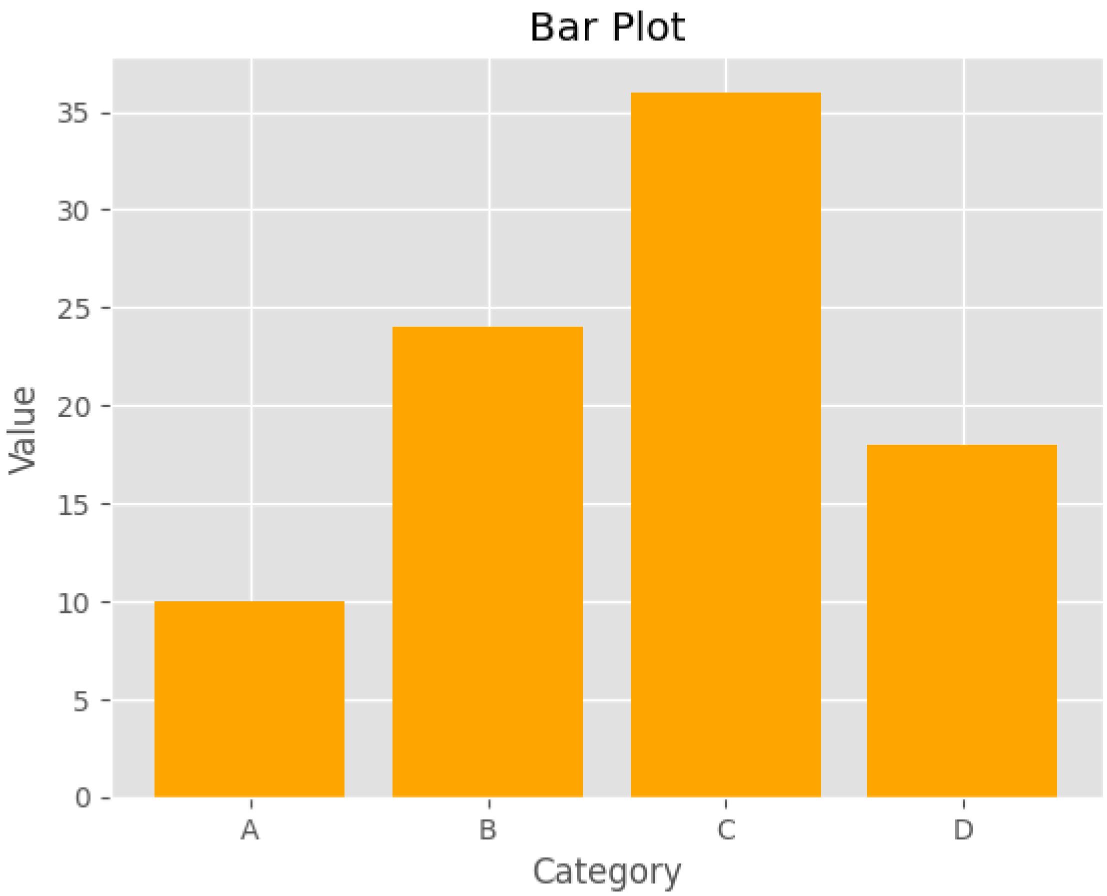
```
# Generate random data for scatter plot
x_scatter = np.random.rand(100)
y_scatter = np.random.rand(100)
plt.scatter(x_scatter, y_scatter, color='purple', alpha=0.7)
plt.title('Scatter Plot')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



7. Bar Plot

Create a bar plot to compare values across categories.

```
In [82]: # Define categories and their values for the bar plot
categories = ['A', 'B', 'C', 'D']
values = [10, 24, 36, 18]
plt.bar(categories, values, color='orange')
plt.title('Bar Plot')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()
```

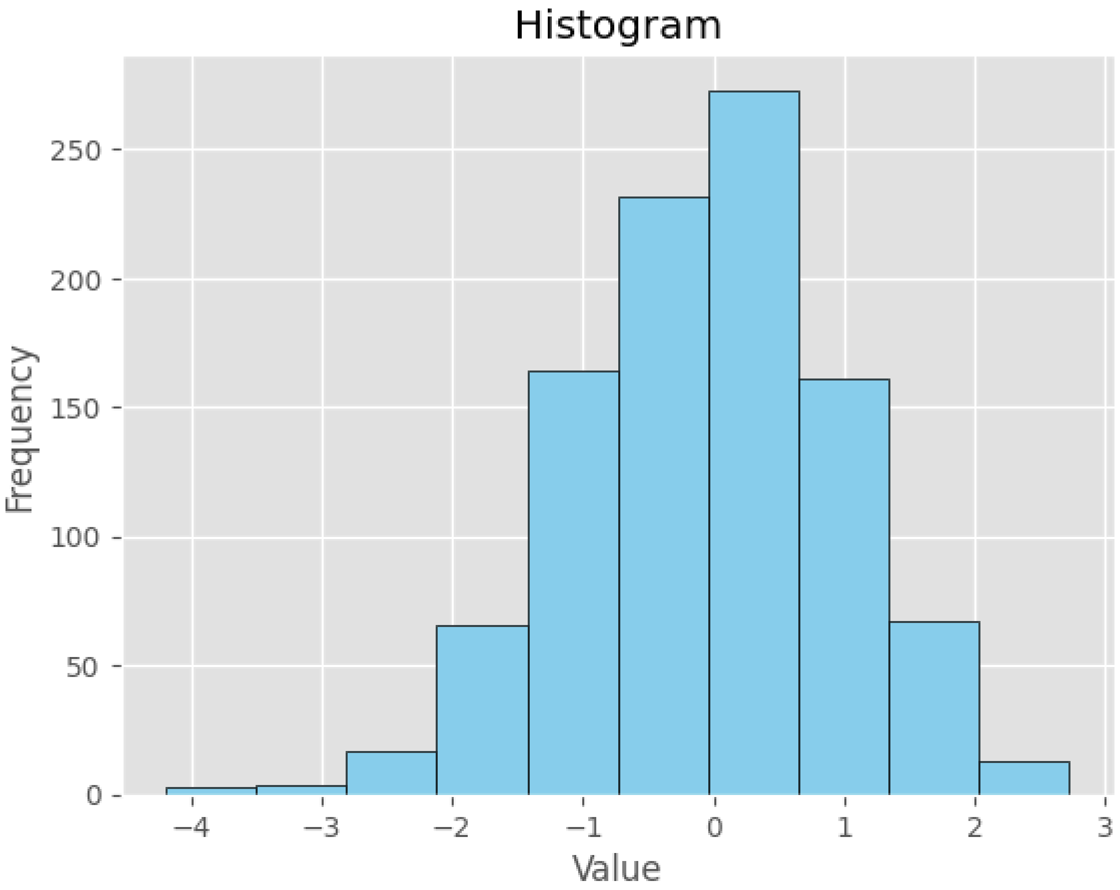


8. Histogram

Show the distribution of a dataset using a histogram.

In [83]:

```
# Generate random data and plot its histogram
data = np.random.randn(1000)
plt.hist(data, bins=10, color='skyblue', edgecolor='black')
plt.title('Histogram')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

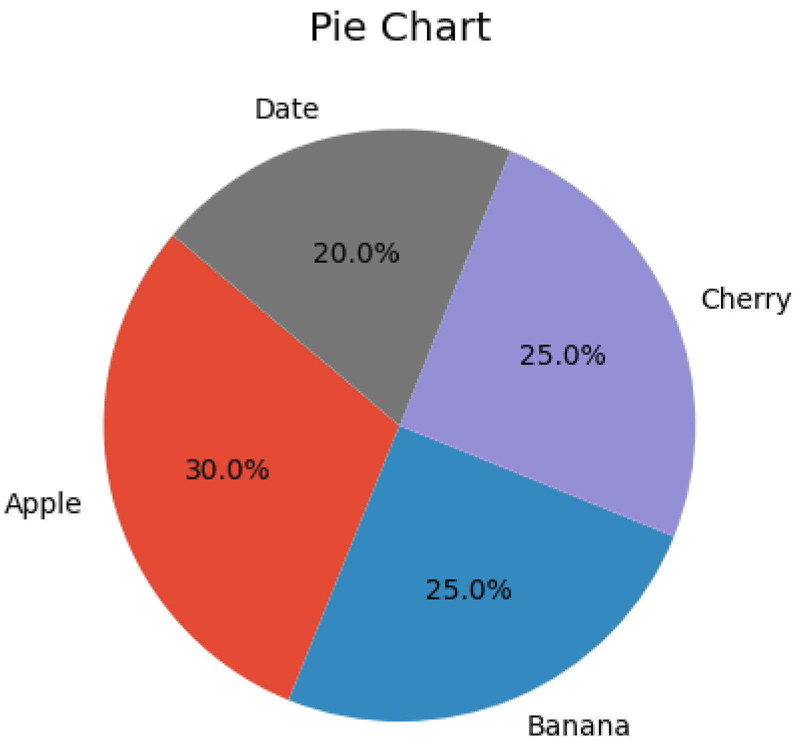


9. Pie Chart

Display proportions of categories as slices of a pie.

In [84]:

```
# Define labels and sizes for the pie chart
labels = ['Apple', 'Banana', 'Cherry', 'Date']
sizes = [30, 25, 25, 20]
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Pie Chart')
plt.show()
```



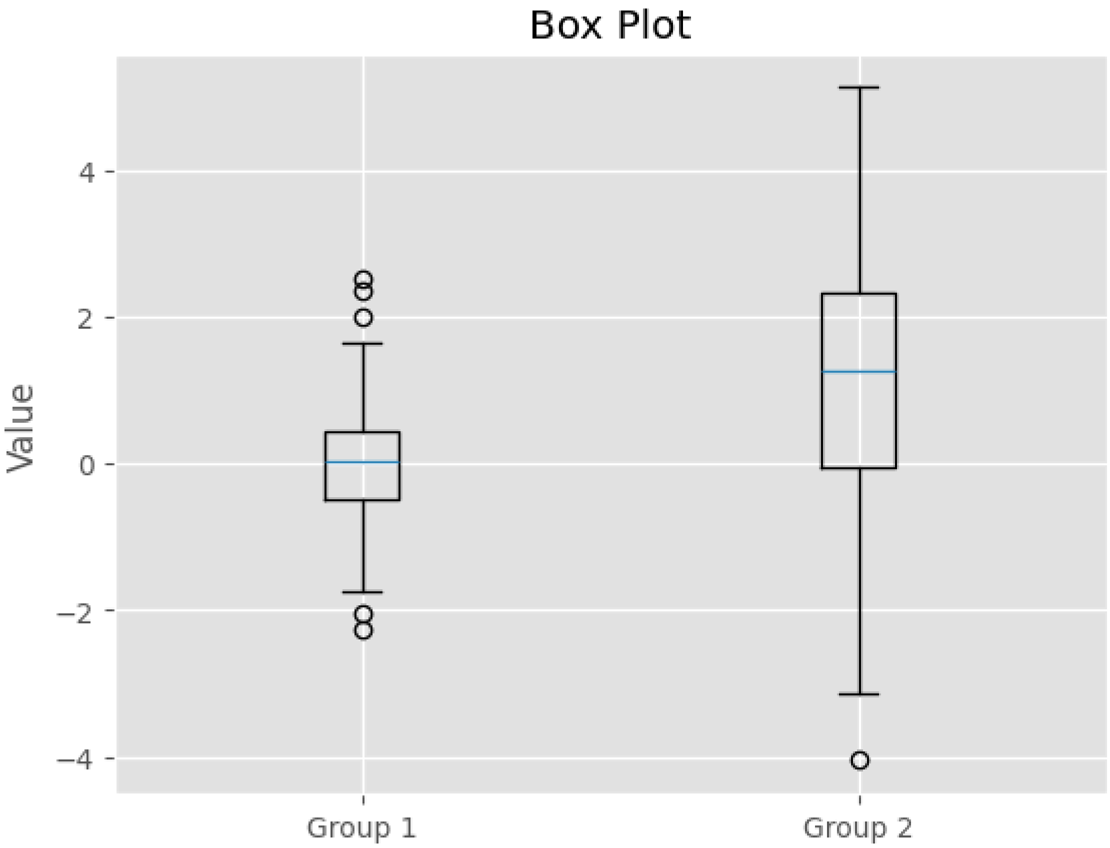
10. Box Plot

Visualize the spread and outliers of two groups using a box plot.

In [85]:

```
# Generate two groups of random data and plot their box plots
data1 = np.random.normal(0, 1, 100)
data2 = np.random.normal(1, 2, 100)
plt.boxplot([data1, data2], labels=['Group 1', 'Group 2'])
plt.title('Box Plot')
plt.ylabel('Value')
plt.show()
```

```
C:\Users\Bhanu Sri V\AppData\Local\Temp\ipykernel_22340\431249029.py:4: Matplotli
bDeprecationWarning: The 'labels' parameter of boxplot() has been renamed 'tick_l
abels' since Matplotlib 3.9; support for the old name will be dropped in 3.11.
  plt.boxplot([data1, data2], labels=['Group 1', 'Group 2'])
```

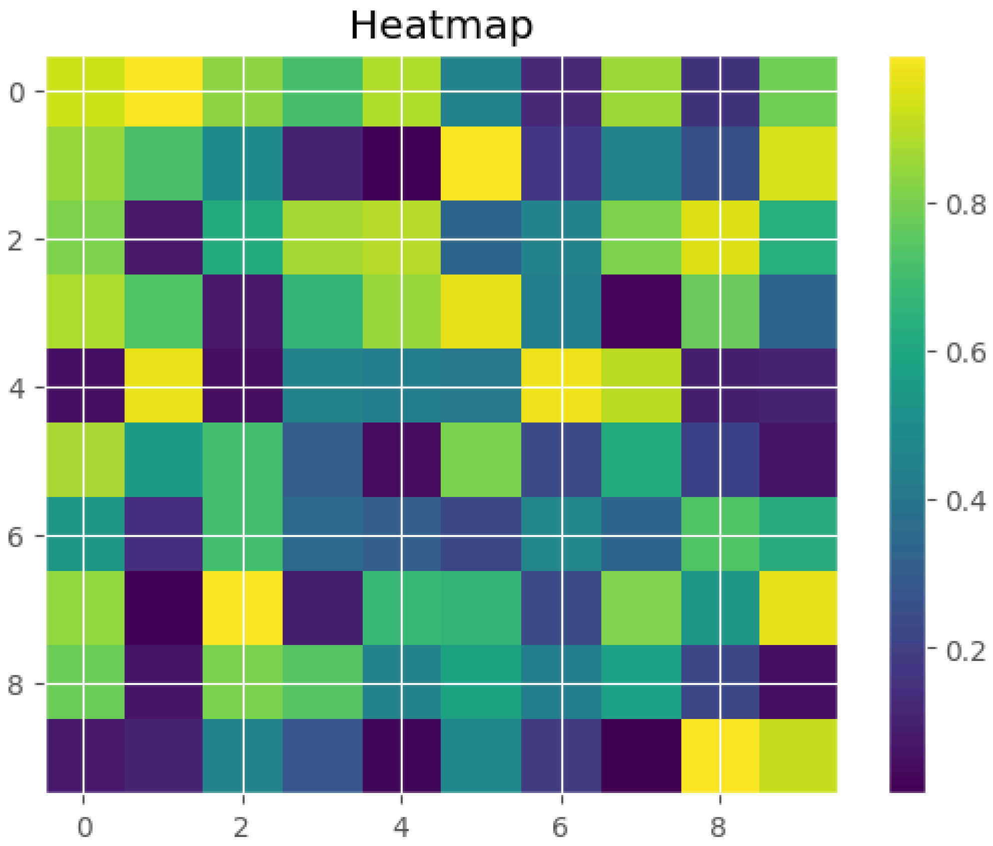


11. Heatmap

Show a 2D matrix of values as a color-coded image (heatmap).

In [86]:

```
# Generate a 10x10 matrix of random values and display as a heatmap
mat = np.random.rand(10, 10)
plt.imshow(mat, cmap='viridis', aspect='auto')
plt.colorbar()
plt.title('Heatmap')
plt.show()
```

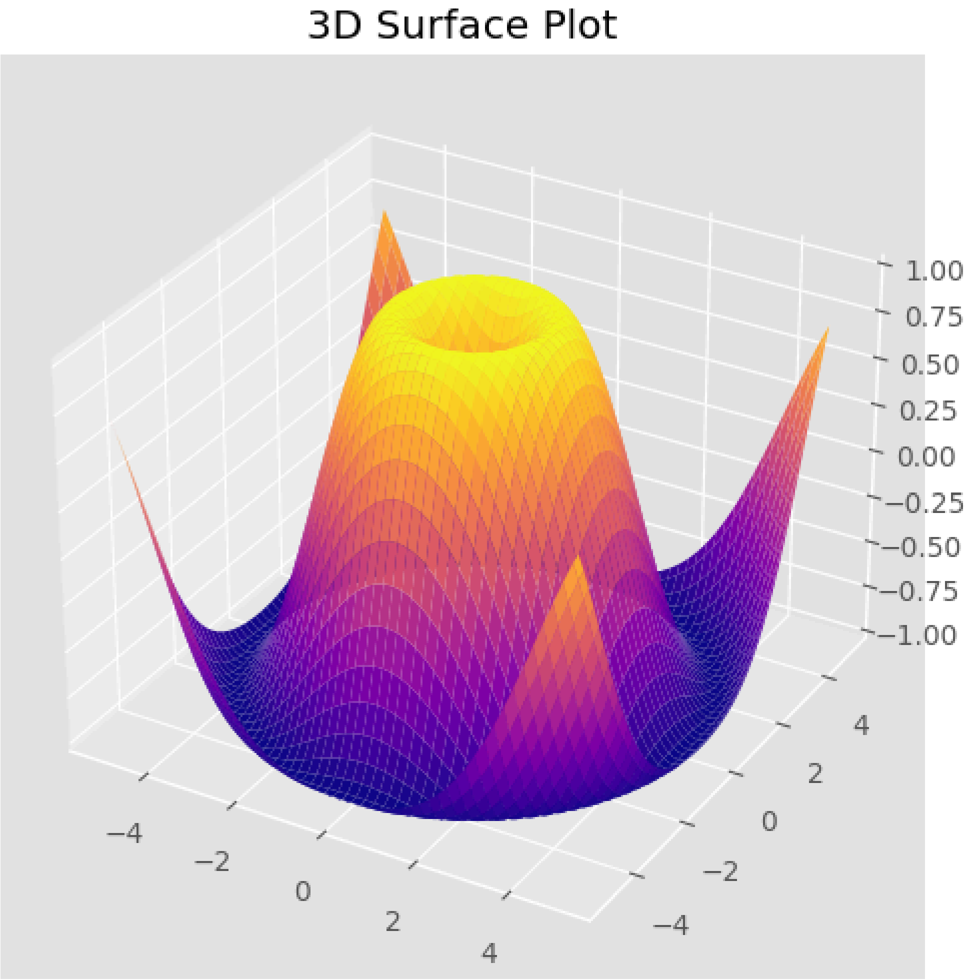


12. Advanced: 3D Plotting

Create a 3D surface plot to visualize functions of two variables.

In [87]:

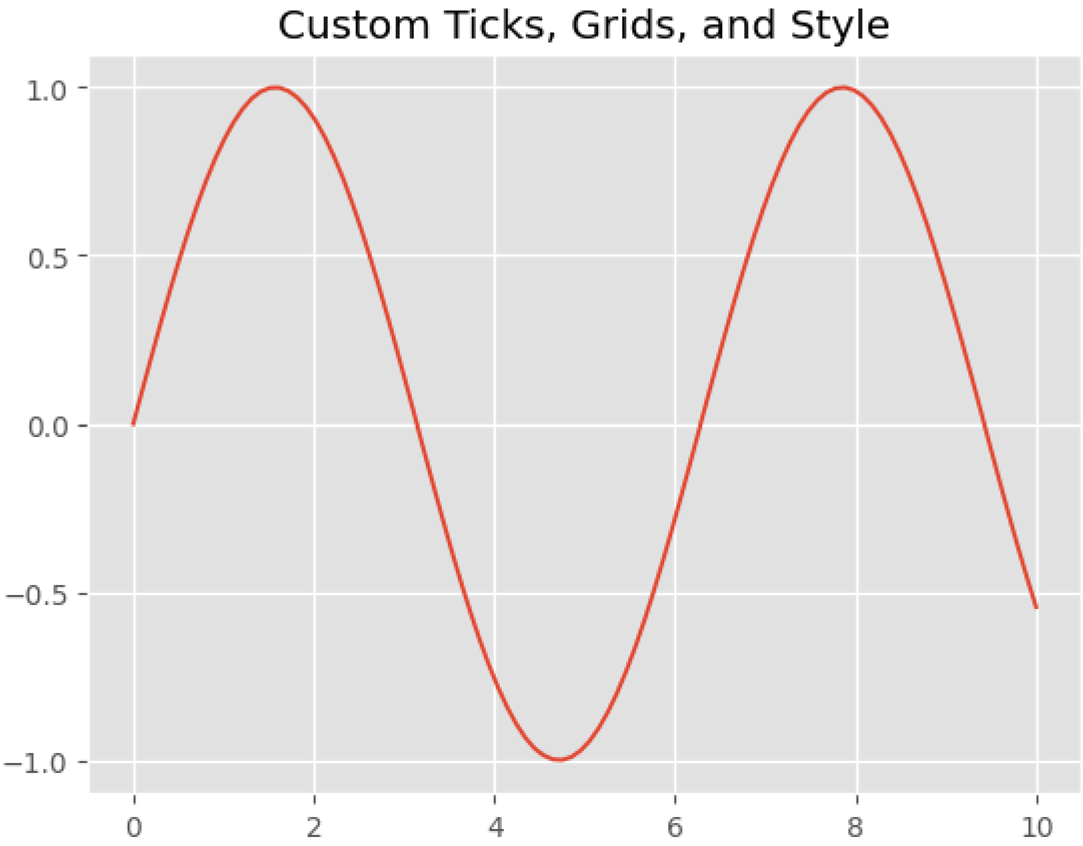
```
# 3D surface plot using meshgrid and plot_surface
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(111, projection='3d')
x3d = np.linspace(-5, 5, 100)
y3d = np.linspace(-5, 5, 100)
x3d, y3d = np.meshgrid(x3d, y3d)
z3d = np.sin(np.sqrt(x3d**2 + y3d**2))
ax.plot_surface(x3d, y3d, z3d, cmap='plasma')
ax.set_title('3D Surface Plot')
plt.show()
```



13. Customizing Ticks, Grids, and Styles

Demonstrate how to change plot style, ticks, and add a grid for better readability.

In [88]:
```
# Use a predefined style, set custom ticks, and enable grid
plt.style.use('ggplot')
plt.plot(x, y)
plt.xticks(np.arange(0, 11, 2))
plt.yticks(np.arange(-1, 1.1, 0.5))
plt.grid(True)
plt.title('Custom Ticks, Grids, and Style')
plt.show()
```


Custom Ticks, Grids, and Style
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10