📊 Basic and Advanced Data Visualizations with Matplotlib

- **Author:** Basel Amr Barakat
- **Email:** baselamr52@gmail.com

- **Date:** 2024-12-25
- **Task Name:** Basic and Advanced Data Visualizations with Matplotlib
- **Task Number:** 22
- **Part:** Matplotlib Advanced
- **Module:** Python Programming Language for AI / ML
- **Submit Number:** 1

Description:

Students will create and customize basic and advanced static plots (line, scatter, subplots, 3D plots) using Matplotlib, focusing on enhancing clarity and aesthetics.

Requirements:

1. **Requirement 1: Basic Line and Scatter Plots**
   - **Description:**
     Create a basic line plot and a scatter plot using Matplotlib. Customize the appearance by adding titles, axis labels, and adjusting the colors. Make sure to choose appropriate titles and labels to make the visualizations clear and informative.
2. **Requirement 2: Advanced Plotting with Subplots and Annotations**
   - **Description:**
     Create a figure with multiple subplots (at least two different types of plots, such as a bar plot and a histogram) in the same figure. Add annotations to one of the plots to highlight specific data points or trends. Customize the appearance of the subplots with gridlines, legends, and different colors.
3. **Requirement 3: 3D Plotting**
   - **Description:**
     Create a 3D scatter plot using Matplotlib, displaying multidimensional data. Label the axes and customize the plot's appearance, including colors and markers. Save the final plot as an image file (e.g., PNG).

📝 1. Introduction & Objective

In this notebook, we will:

- creating and customizing basic and advanced static plots using Matplotlib.
- It emphasizes clarity, aesthetics, and customization in data visualization.

🚀 2. Setting Up the Environment

```
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
# Ignore all the warnings
import warnings
warnings.filterwarnings("ignore")
```

📌 3. Requirement 1

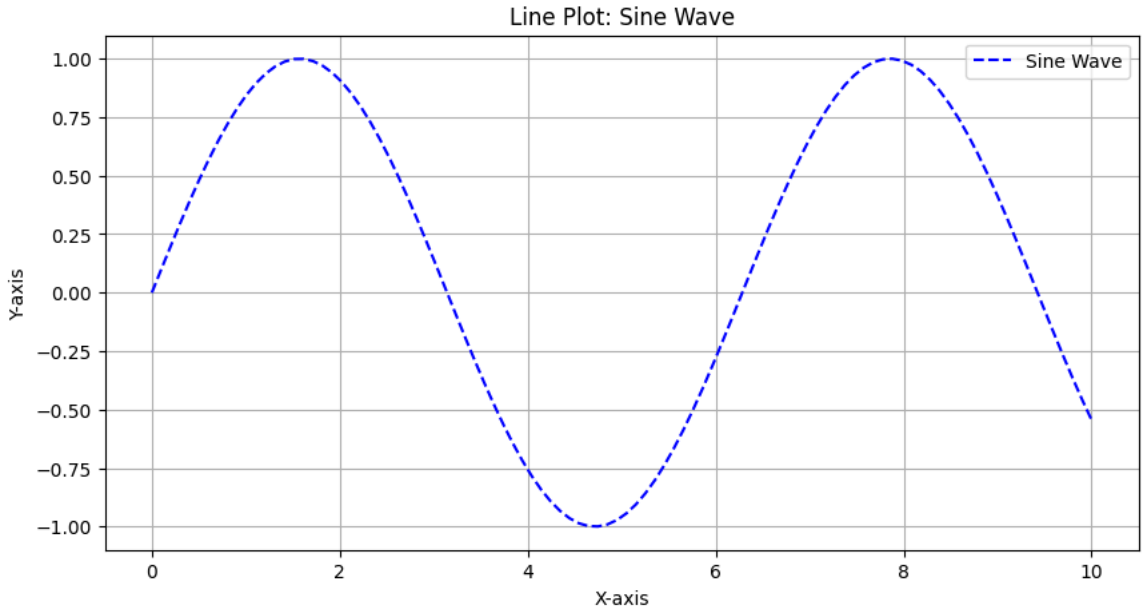
Task Name : Basic Line and Scatter Plots

Description : Create a basic line plot and a scatter plot using Matplotlib. Customize the appearance by adding titles, axis labels, and adjusting the colors.

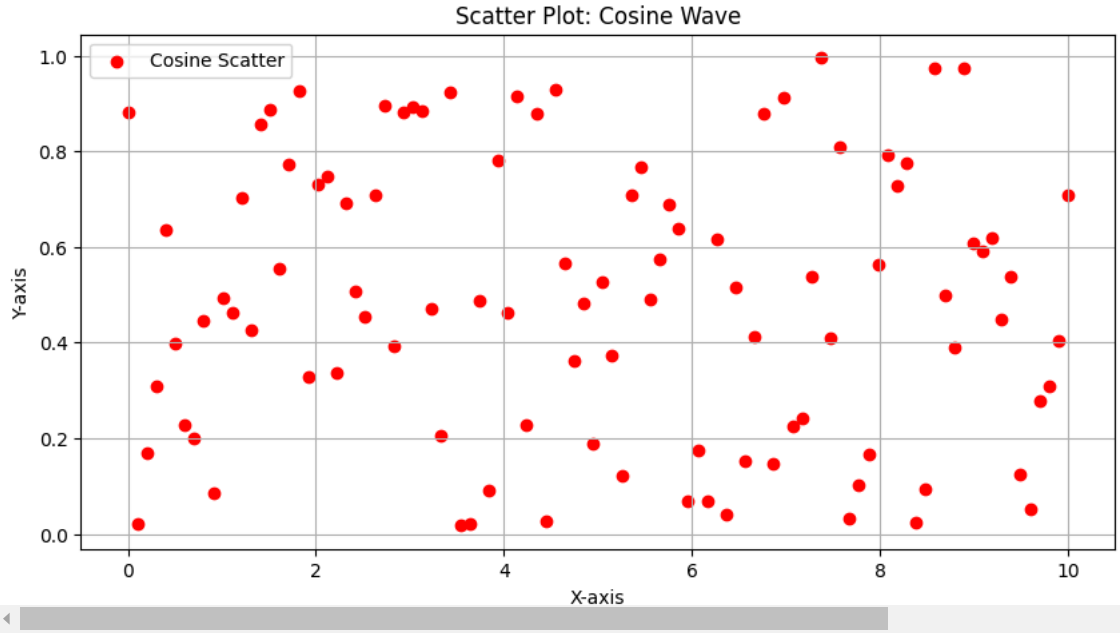
3.1 Ploting a line_plot and Scatter_plot

```
# Make The Data for Line and Scatter plot
x_line = np.linspace(0, 10, 100)
y_line = np.sin(x_line)
y_scatter = np.random.rand(100)

# Create Line Plot
print("Creating a line plot")
plt.figure(figsize=(10, 5))
plt.plot(x_line, y_line, label='Sine Wave', color='b', linestyle='--')
plt.title('Line Plot: Sine Wave')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.grid(True)
plt.savefig('01_line_plot.png',format="png",dpi=800)
plt.show()

# Create Scatter Plot
print("Creating a Scatter plot")
plt.figure(figsize=(10, 5))
plt.scatter(x_line, y_scatter, label='Cosine Scatter', color='r', marker='o')
plt.title('Scatter Plot: Cosine Wave')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.grid(True)
plt.savefig('01_scatter_plot.png',format="png",dpi=800)
plt.show()

```

```
Creating a line plot
```




```
Creating a Scatter plot
```



3.2 Custome Line plots

```
# Get the styles of the plots
styles = plt.style.available
print(f"There is {len(styles)} in the plot lobrary")
print(styles)
```

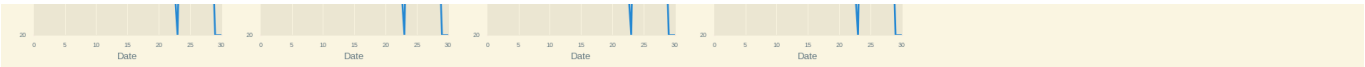
```
There is 28 in the plot lobrary
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'bmh', 'classic', 'dark_background', 'fast', 'fivethir
```

```
# Prepare Dates
dates = np.arange(1, 31)
# Prepare tempreture values
temperature = np.random.randint(20, 35, 30)

# Get all available styles in Matplotlib
styles = plt.style.available

# Create a grid for subplots
n_styles = len(styles)
```

```
cols = 6
rows = -(-n_styles // cols)  # Calculate number of rows needed
print(n_styles,cols,rows)
# Create a figure for subplots
fig, axes = plt.subplots(rows, cols, figsize=(24, 7 * rows))
axes = axes.flatten()  # Flatten axes for easy iteration

# Loop through styles and plot on subplots
for i, style in enumerate(styles):
    plt.style.use(style)
    ax = axes[i]
    ax.plot(dates, temperature, label='Tempreture of January')
    ax.set_xlabel('Date')
    ax.set_ylabel('Temperature')
    ax.set_title(f'Style: {style}', fontsize=10)
    ax.legend()
    ax.grid(True)
    ax.tick_params(axis='both', which='major', labelsize=8)

# Hide any unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.savefig('01_custome_line_plot.png',format="png",dpi=800)
plt.show()
```

28 6 5


Style: Solarize_Light2
Style: _classic_test_patch
Style: _mpl-gallery
Style: _mpl-gallery-nogrid
Style: bmh
Style: classic
Style: dark_background
Style: fast
Style: fivethirtyeight
Style: ggplot
Style: grayscale
Style: seaborn-v0_8
Style: seaborn-v0_8-bright
Style: seaborn-v0_8-colorblind
Style: seaborn-v0_8-dark
Style: seaborn-v0_8-dark-palette
Style: seaborn-v0_8-darkgrid
Style: seaborn-v0_8-deep
Style: seaborn-v0_8-muted
Style: seaborn-v0_8-notebook
Style: seaborn-v0_8-paper
Style: seaborn-v0_8-pastel
Style: seaborn-v0_8-poster
Style: seaborn-v0_8-talk
Style: seaborn-v0_8-ticks
Style: seaborn-v0_8-white
Style: seaborn-v0_8-whitegrid
Style: tableau-colorblind10
Tempreture of January
Temperature
Date




```
with PdfPages('Custome_Styles.pdf') as pdf:
    for style in plt.style.available:
        print(f"Plotting style: {style}")
        plt.style.use(style)
        fig, ax = plt.subplots(figsize=(8, 4))
        ax.plot(dates, temperature, label='Temperature', lw=2)
        ax.set_title(f'Style: {style}', fontsize=14)
        ax.set_xlabel('Dates')
        ax.set_ylabel('Temperature')
        ax.legend()
        ax.grid(True)
        pdf.savefig(fig)
        plt.close(fig)
print("The Plotting styles have been successfully  saved in Costome_Styles.pdf")
```

```
Plotting style: Solarize_Light2
Plotting style: _classic_test_patch
Plotting style: _mpl-gallery
Plotting style: _mpl-gallery-nogrid
Plotting style: bmh
Plotting style: classic
Plotting style: dark_background
Plotting style: fast
Plotting style: fivethirtyeight
Plotting style: ggplot
Plotting style: grayscale
Plotting style: seaborn-v0_8
Plotting style: seaborn-v0_8-bright
Plotting style: seaborn-v0_8-colorblind
Plotting style: seaborn-v0_8-dark
Plotting style: seaborn-v0_8-dark-palette
Plotting style: seaborn-v0_8-darkgrid
Plotting style: seaborn-v0_8-deep
Plotting style: seaborn-v0_8-muted
Plotting style: seaborn-v0_8-notebook
Plotting style: seaborn-v0_8-paper
Plotting style: seaborn-v0_8-pastel
Plotting style: seaborn-v0_8-poster
Plotting style: seaborn-v0_8-talk
Plotting style: seaborn-v0_8-ticks
Plotting style: seaborn-v0_8-white
Plotting style: seaborn-v0_8-whitegrid
Plotting style: tableau-colorblind10
The Plotting styles have been successfully  saved in Costome_Styles.pdf
```

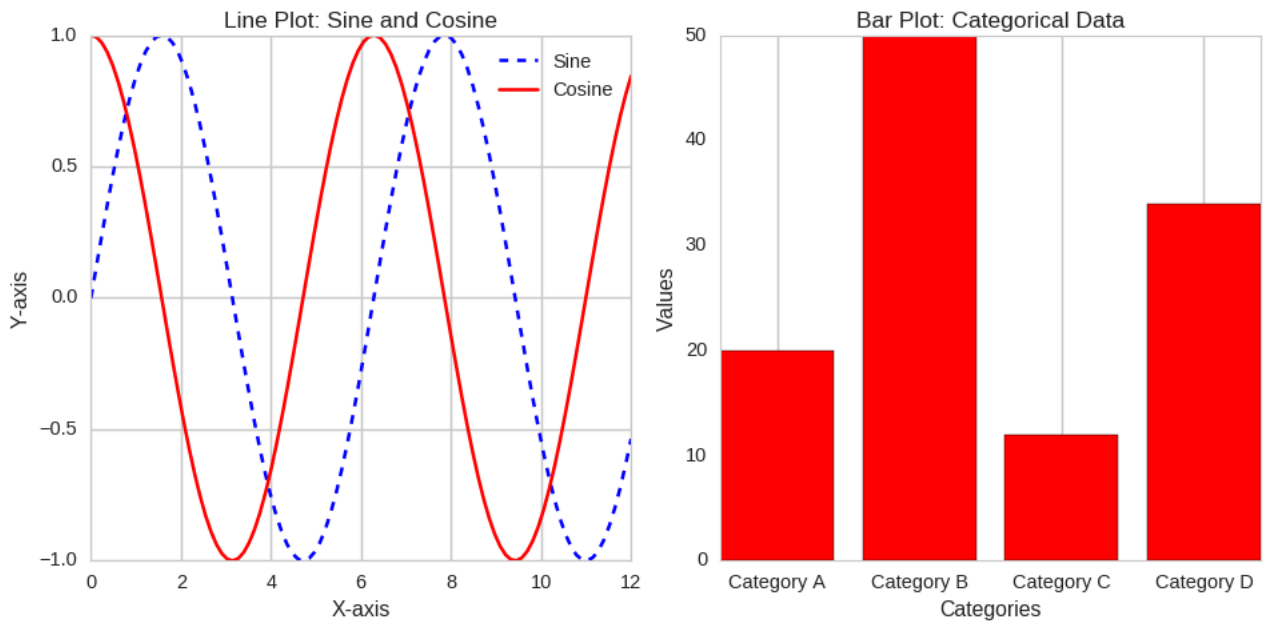
📌 4. Requirement 2

Task Name : Advanced Plotting with Subplots and Annotations

Description : Create a figure with multiple subplots (at least two different types of plots, such as a bar plot and a histogram) in the same figure..

4.1 Creating two subplots with the same sizes

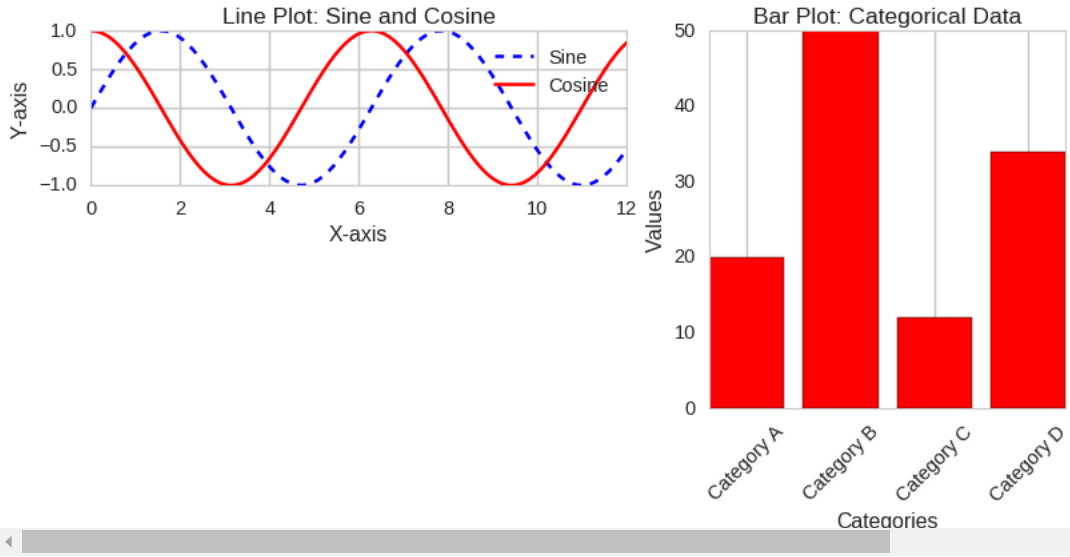
```
# Create the data
x = np.linspace(0, 12, 100)
y1 = np.sin(x)
y2 = np.cos(x)

# Create categorical data for bar plot
categories = ['Category A', 'Category B', 'Category C', 'Category D']
values = [20, 50, 12, 34]

# Create the figure with subplots
plt.figure(figsize=(12, 6))

# Subplot 1: Line plot
plt.subplot(1, 2, 1)
plt.plot(x, y1, label='Sine', color='b', linestyle='--')
plt.plot(x, y2, label='Cosine', color='r', linestyle='-')
plt.title('Line Plot: Sine and Cosine')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.grid(True)

# Subplot 2 : Bar plot
plt.subplot(1, 2, 2)
plt.bar(categories, values, color='r')
plt.title('Bar Plot: Categorical Data')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.grid(axis='y')
plt.savefig('02_LineandBarPlot_SameSizes_.png',format="png",dpi=800)
# Adjust spacing between subplots
plt.tight_layout()
```



4.2 Creating two suplots with different sizes

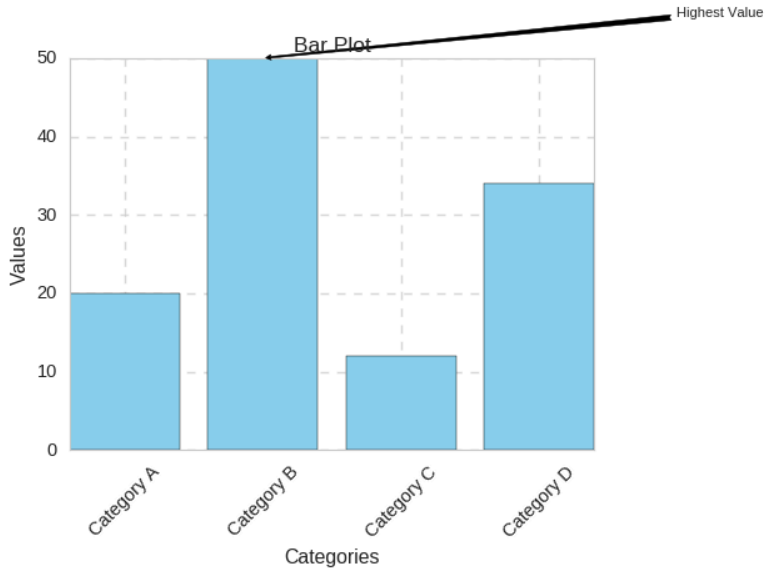
```
# Creating two suplots with different sizes
fig = plt.figure(constrained_layout=True, figsize=(10, 5))
```

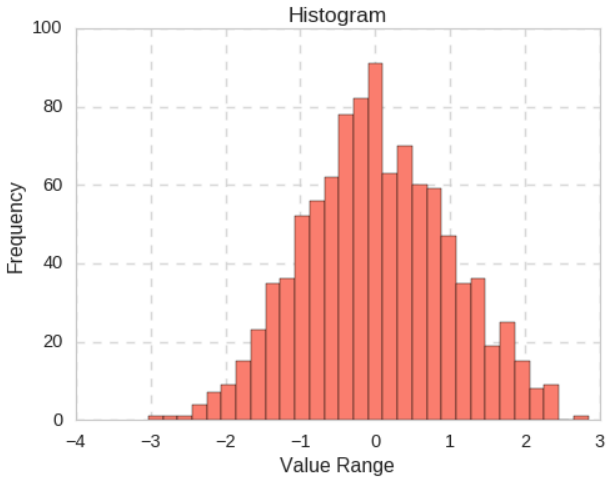
```
gs = fig.add_gridspec(2, 5)

# Subplot 1: Line plot
ax1 = fig.add_subplot(gs[0, :3])
ax1.plot(x, y1, label='Sine', color='b', linestyle='--')
ax1.plot(x, y2, label='Cosine', color='r', linestyle='-')
ax1.set_xlabel('X-axis')
ax1.set_ylabel('Y-axis')
ax1.legend()
ax1.grid(True)
ax1.set_title('Line Plot: Sine and Cosine')

# Subplot 2 : Bar plot
ax2 = fig.add_subplot(gs[:, 3:])
ax2.bar(categories, values, color='r')
ax2.set_title('Bar Plot: Categorical Data')
ax2.set_xlabel('Categories')
ax2.set_ylabel('Values')
ax2.set_xticklabels(labels=categories,rotation=45)
ax2.grid(axis='y')
plt.savefig('02_LineandBarPlot_DifferentSizes_.png',format="png",dpi=800)
```



4.3 Creating Bar plots with annotaion

```
data = np.random.randn(1000)

fig, axs = plt.subplots(1, 2, figsize=(14, 6))

# Bar Plot
axs[0].bar(categories, values, color='skyblue')
axs[0].set_title('Bar Plot')
axs[0].set_xlabel('Categories')
axs[0].set_ylabel('Values')
axs[0].grid(True, linestyle='--', alpha=0.7)
axs[0].annotate('Highest Value', xy=('Category B', 50), xytext=('C', 55),
                arrowprops=dict(facecolor='black', arrowstyle='fancy'),
                horizontalalignment='left',
                verticalalignment='bottom')
axs[0].set_xticklabels(labels=categories,rotation=45)

# Histogram
axs[1].hist(data, bins=30, color='salmon', edgecolor='black')
axs[1].set_title('Histogram')
axs[1].set_xlabel('Value Range')
axs[1].set_ylabel('Frequency')
axs[1].grid(True, linestyle='--', alpha=0.7)

plt.tight_layout()
plt.savefig('02_LineandBarPlot_Annotation_.png',format="png",dpi=800)
plt.show()
```





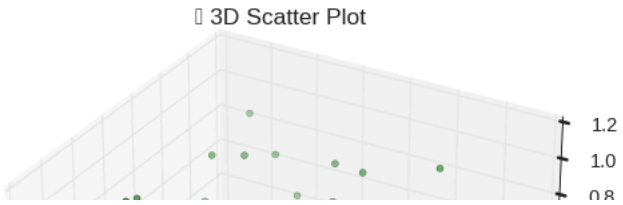
📌 5. Requirement 3

Task Name : 3D Plotting

Description : Create a 3D scatter plot using Matplotlib, displaying multidimensional data. Label the axes and customize the plot's appearance, including colors and markers.

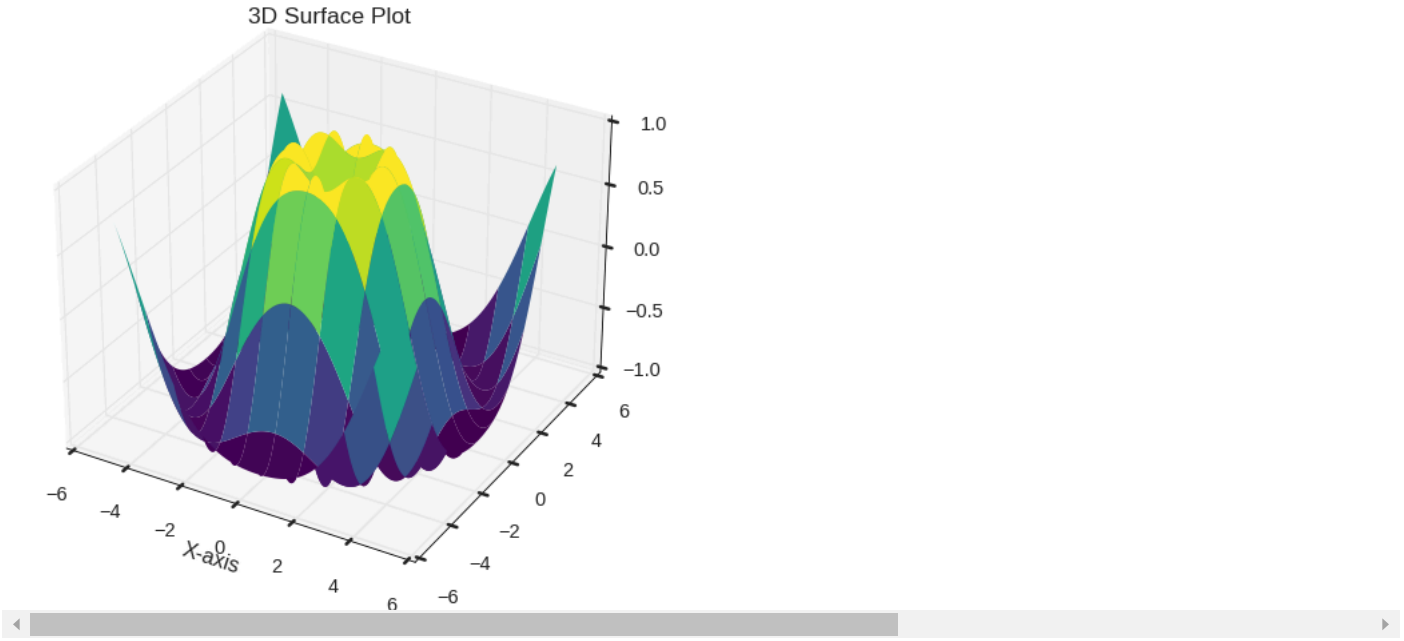
5.1 Scatter plot

```
# 3D Data
x = np.random.rand(100)
y = np.random.rand(100)
z = np.random.rand(100)

fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(x, y, z, c='g', marker='o')

ax.set_title('🎲 3D Scatter Plot')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')

plt.savefig('03_3d_scatter_plot.png',format='png',dpi=800)
plt.show()

print('3D Scatter Plot saved as 3d_scatter_plot.png')
```



5.2 3D surface plot

```
# Sample data
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
z = np.sin(np.sqrt(X**2 + Y**2))

# Create the 3D surface plot
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, z, cmap='viridis', edgecolor='none')

ax.set_title('3D Surface Plot')
ax.set_xlabel('X-axis')
plt.savefig('03_3d_SurfacePlot.png',format="png",dpi=800)
plt.show()
```



5.3 Customizing 3D plots

```
# Customize 3D plots
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, z, cmap='plasma', edgecolor='k',alpha = 0.7)
ax.set_title('Custom 3D Surface Plot')
ax.set_xlabel('X-axis')
```