**Report on Customized Complex Plot Using Matplotlib**

**1. Objective:**

The objective of this task is to create a customized plot using Matplotlib, with a specific focus on customizing the appearance of the plot, such as title, axis labels, line style, and grid. The plot will display a sine wave based on a set of data points.

**2. Approach:**

In this task, we use Matplotlib’s Object-Oriented API to create a plot. This approach gives us greater flexibility to control plot elements and customize them according to our preferences.

**3. Code Implementation:**

python

Copy code

import matplotlib.pyplot as plt

import numpy as np

# Data

x = np.linspace(0, 10, 100)

y = np.sin(x)

# Create a figure and axis object

fig, ax = plt.subplots(figsize=(8, 6))

# Plotting the data

ax.plot(x, y, label='Sine wave', color='b', linewidth=2)

# Customizing the plot

ax.set\_title('Customized Complex Plot', fontsize=16)

ax.set\_xlabel('X-axis', fontsize=14)

ax.set\_ylabel('Y-axis', fontsize=14)

ax.legend(loc='upper right')

ax.grid(True)

# Show plot

plt.show()

**4. Explanation of Code:**

1. **Data Preparation:**
   * x = np.linspace(0, 10, 100) generates 100 evenly spaced values between 0 and 10. These will serve as the x-coordinates of the plot.
   * y = np.sin(x) computes the sine of each value in the x array, producing the corresponding y-coordinates for the sine wave.
2. **Creating the Plot:**
   * fig, ax = plt.subplots(figsize=(8, 6)) creates a new figure (fig) and axis object (ax). The figsize=(8, 6) parameter specifies the size of the figure in inches.
   * ax.plot(x, y, label='Sine wave', color='b', linewidth=2) plots the sine wave on the axis object ax. The line color is set to blue ('b'), and the line width is set to 2 pixels. The label for the line is 'Sine wave', which will appear in the legend.
3. **Customizing the Plot:**
   * ax.set\_title('Customized Complex Plot', fontsize=16) adds a title to the plot with a font size of 16.
   * ax.set\_xlabel('X-axis', fontsize=14) and ax.set\_ylabel('Y-axis', fontsize=14) add labels to the x-axis and y-axis with font sizes of 14.
   * ax.legend(loc='upper right') adds a legend to the plot at the upper-right corner to indicate the meaning of the plotted line.
   * ax.grid(True) enables the grid on the plot to improve readability and alignment of data points.
4. **Displaying the Plot:**
   * plt.show() renders and displays the plot in a separate window.

**5. Result:**

Running the code generates the following plot:

* A **sine wave** from x=0x = 0x=0 to x=10x = 10x=10 with a blue line and a thickness of 2 pixels.
* The plot is titled "Customized Complex Plot", with labeled x-axis as "X-axis" and y-axis as "Y-axis".
* The legend shows "Sine wave" at the upper-right corner.
* The plot includes a grid for better visual clarity.

**6. Visualization and Customization:**

The customization options used in the code enhance the plot’s clarity and presentation. The title, axis labels, and grid lines help to provide context and make the plot easier to interpret. The use of a legend ensures that even if multiple plots were included, each one could be identified easily.

**7. Conclusion:**

This exercise demonstrates how to create a customized plot using Matplotlib’s Object-Oriented API. By adjusting parameters such as line color, axis labels, grid, and title, we can tailor the appearance of the plot to suit specific needs. The Object-Oriented approach offers more flexibility, allowing us to manipulate individual components of the plot with ease.

4o mini

Top of Form

Bottom of Form