Customizing Matplotlib Graphs

Customizing Matplotlib graphs is a fundamental skill for creating effective visualizations. This lab exercise will guide you through various customization options, including titles, labels, legends, colors, and more.

Lab 1: Exercise: Customizing Matplotlib Graphs

Objective

Learn how to customize Matplotlib graphs to improve their readability and aesthetics.

Requirements

- Python 3.x
- Matplotlib library
- Numpy library

Steps

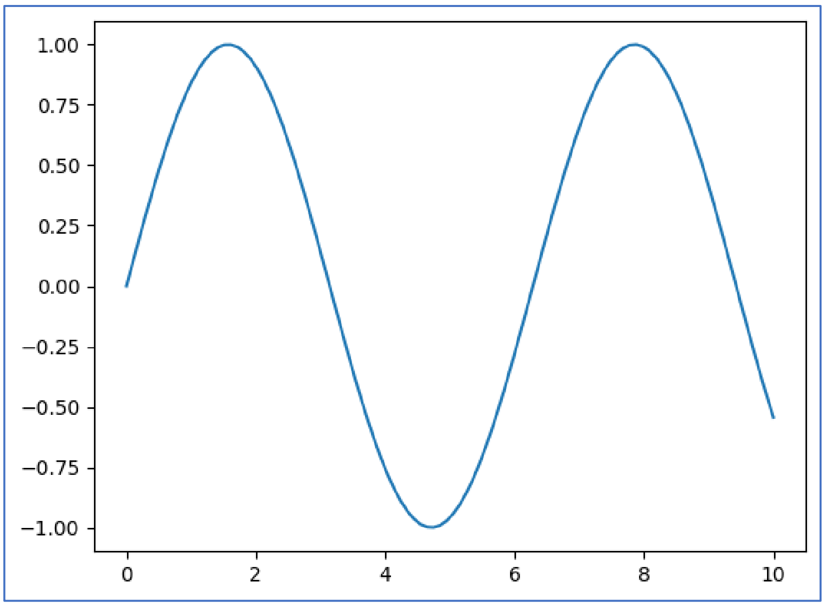
Install Required Libraries (if not already installed):

```
pip install matplotlib numpy
```

1. Basic Plot Customization:

Start with a simple line plot and progressively customize it.

```
import numpy as np

import matplotlib.pyplot as plt


# Create some data

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

y = np.sin(x)


# Create a basic plot

fig, ax = plt.subplots()

ax.plot(x, y)


# Display the plot

plt.show()
```

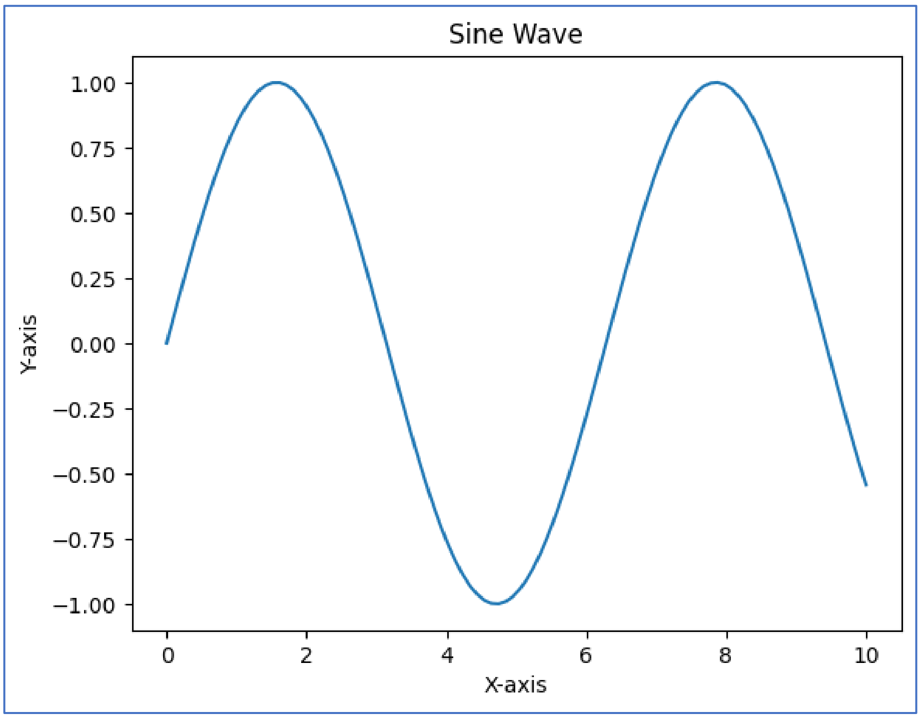


2. Lab Exercise Tasks:

a) Task 1: Adding Titles and Labels

Add a title, and x and y labels to the plot.

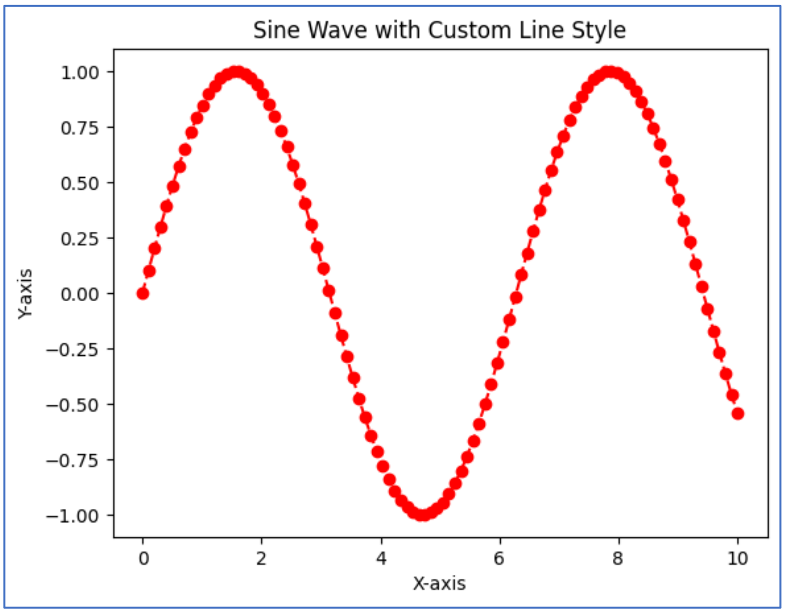
```
fig, ax = plt.subplots()

ax.plot(x, y)


# Add a title and labels

ax.set_title('Sine Wave')

ax.set_xlabel('X-axis')

ax.set_ylabel('Y-axis')


# Display the plot

plt.show()
```



b) Task 2: Customizing Line Styles and Colors

Change the line style, color, and add markers.

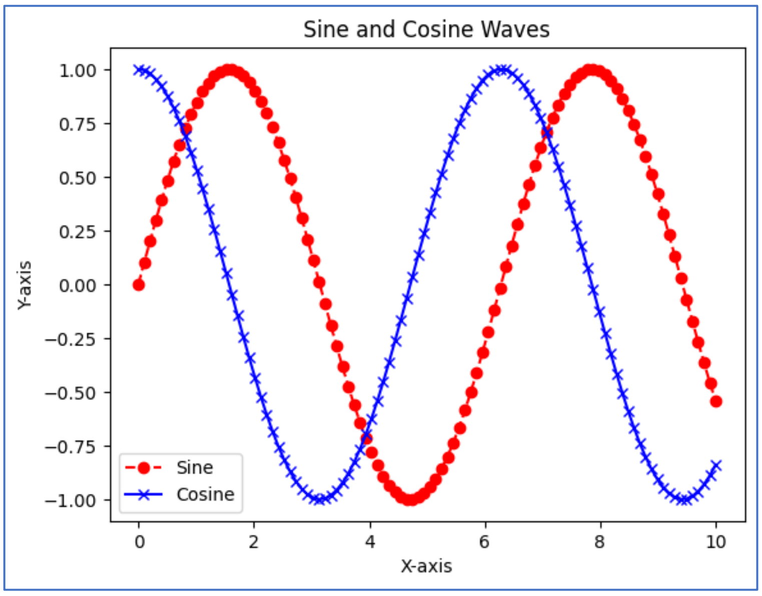
```
fig, ax = plt.subplots()
ax.plot(x, y, linestyle='--', color='r', marker='o')

# Add a title and labels
ax.set_title('Sine Wave with Custom Line Style')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')

# Display the plot
plt.show()
```



c) Task 3: Adding a Legend

Add multiple lines to the plot and include a legend.

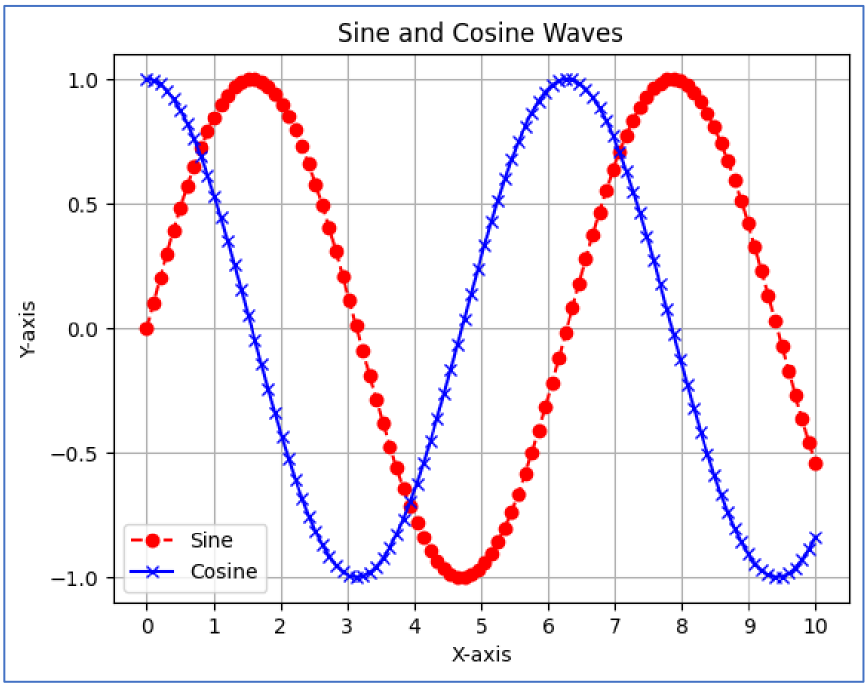
```
y2 = np.cos(x)

fig, ax = plt.subplots()

ax.plot(x, y, linestyle='--', color='r', marker='o', label='Sine')

ax.plot(x, y2, linestyle='-', color='b', marker='x', label='Cosine')

# Add a title, labels, and legend

ax.set_title('Sine and Cosine Waves')

ax.set_xlabel('X-axis')

ax.set_ylabel('Y-axis')

ax.legend()

# Display the plot

plt.show()
```



d) Task 4: Customizing Ticks and Grid

Customize the ticks and add grid lines to the plot.

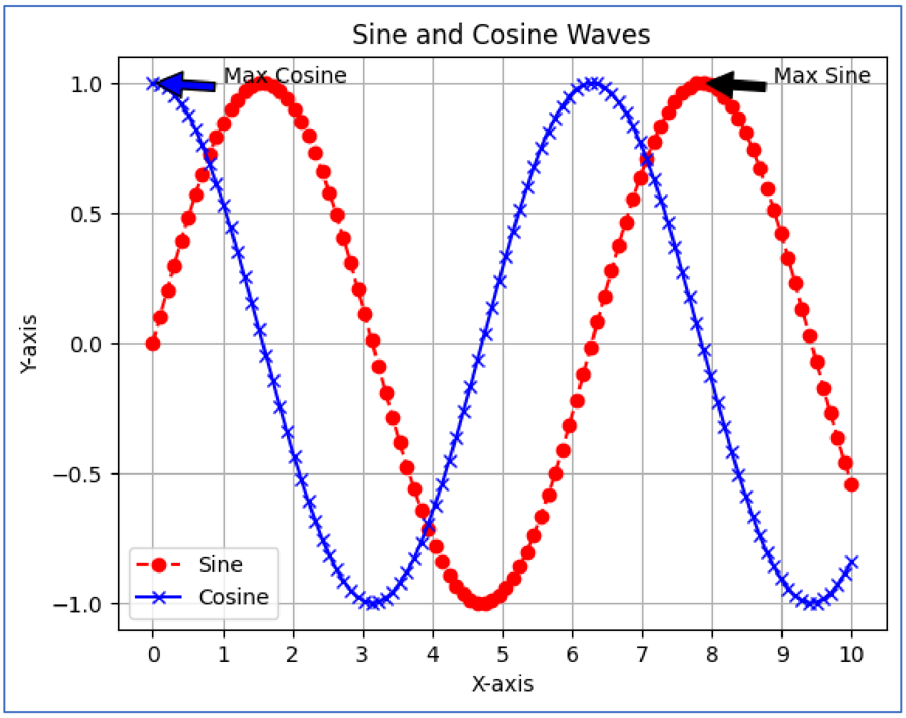
```
fig, ax = plt.subplots()
ax.plot(x, y, linestyle='--', color='r', marker='o', label='Sine')
ax.plot(x, y2, linestyle='-', color='b', marker='x', label='Cosine')

# Add a title, labels, and legend
ax.set_title('Sine and Cosine Waves')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.legend()

# Customize ticks
ax.set_xticks(np.arange(0, 11, 1))
ax.set_yticks(np.arange(-1, 1.5, 0.5))

# Add grid
ax.grid(True)

# Display the plot
plt.show()
```



e) Task 5: Adding Annotations

Add annotations to highlight specific points on the plot.

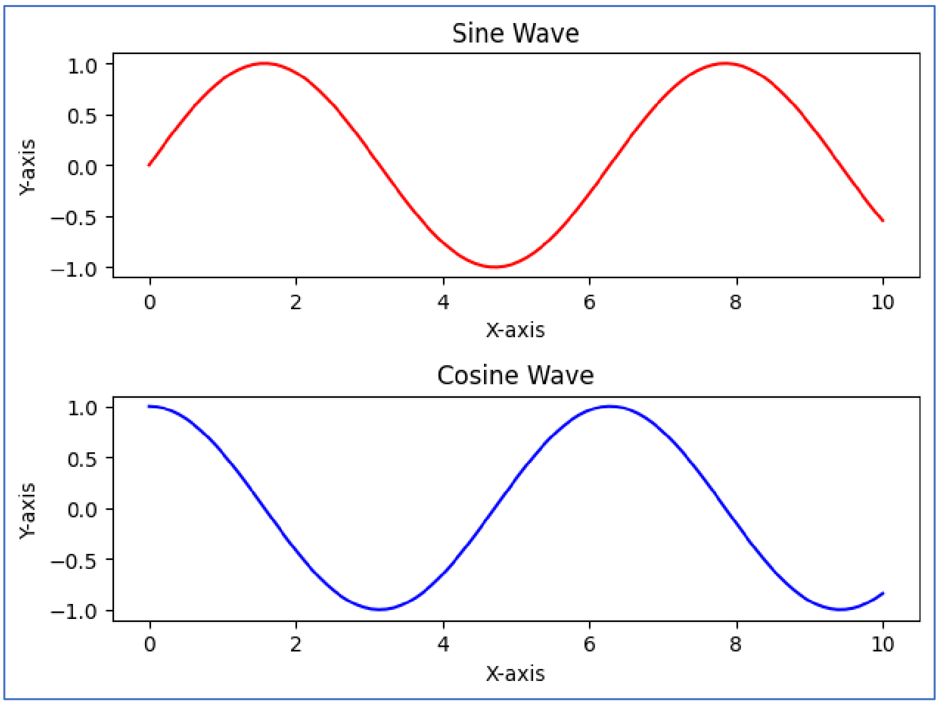
```
fig, ax = plt.subplots()
ax.plot(x, y, linestyle='--', color='r', marker='o', label='Sine')
ax.plot(x, y2, linestyle='-', color='b', marker='x', label='Cosine')

# Add a title, labels, and legend
ax.set_title('Sine and Cosine Waves')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.legend()

# Customize ticks
ax.set_xticks(np.arange(0, 11, 1))
ax.set_yticks(np.arange(-1, 1.5, 0.5))

# Add grid
ax.grid(True)

# Add annotations
max_y = np.max(y)
max_y_index = np.argmax(y)
ax.annotate('Max Sine', xy=(x[max_y_index], max_y), xytext=(x[max_y_index] + 1, 
max_y),
        arrowprops=dict(facecolor='black', shrink=0.05))

max_y2 = np.max(y2)
max_y2_index = np.argmax(y2)
ax.annotate('Max Cosine', xy=(x[max_y2_index], max_y2), 
xytext=(x[max_y2_index] + 1, max_y2),
        arrowprops=dict(facecolor='blue', shrink=0.05))

# Display the plot
plt.show()
```



f) Task 6: Subplots

Create multiple subplots in a single figure.

```
fig, axs = plt.subplots(2)

# First subplot
axs[0].plot(x, y, 'r-')
axs[0].set_title('Sine Wave')
axs[0].set_xlabel('X-axis')
axs[0].set_ylabel('Y-axis')

# Second subplot
axs[1].plot(x, y2, 'b-')
axs[1].set_title('Cosine Wave')
axs[1].set_xlabel('X-axis')
axs[1].set_ylabel('Y-axis')

# Adjust layout
plt.tight_layout()

# Display the plot
plt.show()
```



By completing these tasks, you should have a good understanding of how to customize Matplotlib graphs to make them more informative and visually appealing.

Subplot using Boxplot

Generate Sample Data:

Generate some sample data to be used in the box plots.

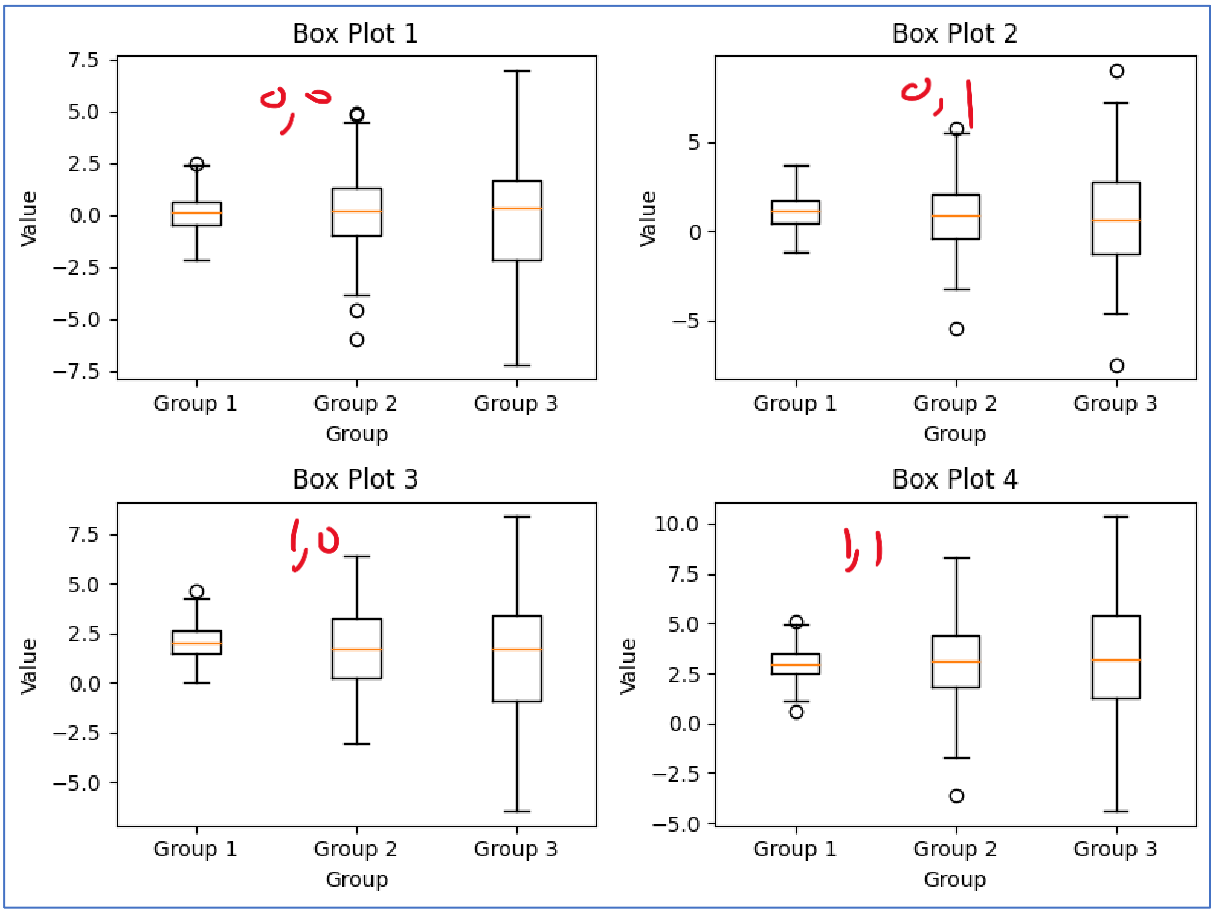
```
import numpy as np
import matplotlib.pyplot as plt

# Generate sample data
np.random.seed(10)
data1 = [np.random.normal(0, std, 100) for std in range(1, 4)]
data2 = [np.random.normal(1, std, 100) for std in range(1, 4)]
data3 = [np.random.normal(2, std, 100) for std in range(1, 4)]
data4 = [np.random.normal(3, std, 100) for std in range(1, 4)]
```

Create a 2x2 Grid of Subplots with Box Plots:

Create subplots and add box plots to each subplot.

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

# First subplot
axs[0, 0].boxplot(data1)
axs[0, 0].set_title('Box Plot 1')
axs[0, 0].set_xlabel('Group')
axs[0, 0].set_ylabel('Value')
axs[0, 0].set_xticklabels(['Group 1', 'Group 2', 'Group 3'])

# Second subplot
axs[0, 1].boxplot(data2)
axs[0, 1].set_title('Box Plot 2')
axs[0, 1].set_xlabel('Group')
axs[0, 1].set_ylabel('Value')
axs[0, 1].set_xticklabels(['Group 1', 'Group 2', 'Group 3'])

# Third subplot
axs[1, 0].boxplot(data3)
axs[1, 0].set_title('Box Plot 3')
axs[1, 0].set_xlabel('Group')
axs[1, 0].set_ylabel('Value')
axs[1, 0].set_xticklabels(['Group 1', 'Group 2', 'Group 3'])

# Fourth subplot
axs[1, 1].boxplot(data4)
axs[1, 1].set_title('Box Plot 4')
axs[1, 1].set_xlabel('Group')
axs[1, 1].set_ylabel('Value')
```

```
axs[1, 1].set_xticklabels(['Group 1', 'Group 2', 'Group 3'])
# Adjust layout
plt.tight_layout()

# Display the plots
plt.show()
```


Box Plot 1
Value
Group 1
Group 2
Group 3
Group
Box Plot 2
Box Plot 3
Box Plot 4