Text and Annotation, Customization, and 3D Plotting in Matplotlib

1. Text and Annotation

Text and annotations in plots help add context, highlight data points, or label specific areas.

Syntax for Adding Text and Annotation

- **plt.text(x, y, s, ...)**: Adds text at position (x, y).
  - **x**, **y**: Coordinates for the text.
  - **s**: Text string.
  - Additional arguments: fontsize, color, backgroundcolor, etc.
- **plt.annotate(s, xy, xytext, ...)**: Adds an annotation pointing to a specific data point.
  - **s**: Text string.
  - **xy**: Coordinates of the point to annotate.
  - **xytext**: Position of the annotation text.
  - Arrow styles can be customized with the arrowprops argument.

Example: Text and Annotation

```
import matplotlib.pyplot as plt

# Data
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Plotting
plt.plot(x, y, marker='o', label='Line Plot')

# Add text
plt.text(3, 6, "Midpoint", fontsize=12, color='blue')

# Add annotation
plt.annotate("Peak Point", xy=(5, 10), xytext=(4, 12),
        arrowprops=dict(facecolor='red', shrink=0.05))

# Add labels and legend
plt.title("Text and Annotation Example")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()

# Show plot
```

```
plt.show()
```

Explanation

1. **Text**:
   - **plt.text(3, 6, "Midpoint")**: Places the text "Midpoint" at coordinates (3, 6).
2. **Annotation**:
   - **plt.annotate()**:
     - Annotates the point (5, 10) with the label "Peak Point".
     - The arrow points to (5, 10), and the label is positioned at (4, 12).
3. **Arrow Customization**:
   - **arrowprops**: Adds an arrow pointing to the annotated point.
   - facecolor='red': Sets the arrow color.

2. Customization

Matplotlib allows you to customize plots extensively to suit your needs.

Customization Features

1. **Line Styles**: linestyle='--', linewidth=2, etc.
2. **Markers**: marker='o', markerfacecolor='green'.
3. **Colors**: Use named colors, hex codes, or colormaps.
4. **Grid Lines**: plt.grid(color='gray', linestyle='--').
5. **Figure Size**: plt.figure(figsize=(8, 6)).

Example: Customization

```
# Customized Plot
plt.plot(x, y, color='orange', linestyle='--', marker='s', markersize=8,
        markerfacecolor='blue', label="Custom Line")
plt.title("Customized Plot", fontsize=14, color='purple')
plt.xlabel("X-axis", fontsize=12)
plt.ylabel("Y-axis", fontsize=12)
plt.grid(color='gray', linestyle='--', linewidth=0.5)
plt.legend()
plt.show()
```

3. Three-Dimensional Plotting

Matplotlib supports 3D plotting through the mpl_toolkits.mplot3d module.

Syntax for 3D Plotting

```
from mpl_toolkits.mplot3d import Axes3D
ax = fig.add_subplot(projection='3d')
ax.plot(x, y, z, ...)
```

Example: 3D Plot

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

# Generate 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 3D plot
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# Plot surface
surf = ax.plot_surface(X, Y, Z, cmap='viridis', edgecolor='none')

# Add color bar
fig.colorbar(surf, ax=ax, shrink=0.5, aspect=10)

# Add labels
ax.set_title("3D Surface Plot")
ax.set_xlabel("X-axis")
ax.set_ylabel("Y-axis")
ax.set_zlabel("Z-axis")

# Show plot
plt.show()
```

Explanation of the 3D Example

1. **Data Creation**:
   - **np.meshgrid()**: Creates a grid of x and y values.
   - **Z**: Computes $\sin(\sqrt{x^2 + y^2})$.
2. **3D Axis**:
   - **projection='3d'**: Enables 3D plotting.
3. **Surface Plot**:
   - **ax.plot_surface()**: Creates a 3D surface plot.

- **cmap='viridis'**: Sets the colormap.

4. **Color Bar**:
   - Adds a color bar for visualizing the Z-axis values.