Matplotlib Visualization Guide

1. Introduction to Matplotlib

Matplotlib is a widely used library for creating static, animated, and interactive visualizations in .

Installation:

pip install matplotlib

Importing Matplotlib:

import matplotlib.pyplot as plt

2. Types of Charts in Matplotlib

1. Line Plot

Used to display data points connected by a line, typically to show trends.

Syntax:

plt.plot(x, y, linestyle, marker, color)
plt.xlabel("X-axis label")
plt.ylabel("Y-axis label")
plt.title("Title")
plt.show()

The plt.plot(x, y, linestyle, marker, color) function in **Matplotlib** is used to plot a 2D line graph. Here's a breakdown of each parameter:

Syntax:

plt.plot(x, y, linestyle, marker, color)

- x: The data values for the x-axis (list, array, or range).
- y: The data values for the y-axis (list, array, or range).
- linestyle (ls): Defines the style of the line. Example values:
    - '-' : Solid line (default)

    - '--' : Dashed line
    - ':' : Dotted line
    - '-.' : Dash-dot line
- marker: Defines the shape of markers at data points. Example values:
    - 'o' : Circle
    - 's' : Square
    - '^' : Triangle
    - 'd' : Diamond
- color (c): Specifies the color of the line and markers. Example values:
    - 'r' : Red
    - 'g' : Green
    - 'b' : Blue
    - 'k' : Black
    - Can also use hex codes (e.g., '#ff5733').

Example:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]

plt.plot(x, y, linestyle='-', marker='o', color='b')
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.title("Line Plot Example")
plt.show()
```

2. Bar Chart

Used to compare categorical data.

Syntax:

```
plt.bar(x, height, color)
```

x: The categorical or numerical values representing the positions of the bars on the x-axis.
height: The height of each bar, representing the values to be plotted.
color (optional): Specifies the color of the bars.

Example:

```
categories = ["A", "B", "C", "D"]
values = [5, 7, 3, 8]

plt.bar(categories, values, color='skyblue')
plt.xlabel("Categories")
plt.ylabel("Values")
plt.title("Bar Chart Example")
plt.show()
```

Horizontal Bar Chart

Similar to a bar chart but in horizontal form.

Syntax:

```
plt.barh(y, width, color)
```

Example:

```
plt.barh(categories, values, color='green')
plt.xlabel("Values")
plt.ylabel("Categories")
plt.title("Horizontal Bar Chart Example")
plt.show()
```

Scatter Plot

Used to show relationships between two numerical variables.

Syntax:

```
plt.scatter(x, y, color, marker)
```

x: The data values for the x-axis (list, array, or range).

y: The data values for the y-axis (list, array, or range).

color (optional): Specifies the color of the points.

marker (optional): Defines the shape of the data points.

Example values:

'o' : Circle (default)

's' : Square

'^' : Triangle

'd' : Diamond

'x' : X shape

Example:

```
x = [1, 2, 3, 4, 5]
y = [10, 15, 8, 20, 18]

plt.scatter(x, y, color='red', marker='x')
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.title("Scatter Plot Example")
plt.show()
```

Histogram

Used to visualize the distribution of a dataset.

Syntax:

```
plt.hist(data, bins, color, edgecolor)
```

data: The dataset (list or array) for which the histogram is created.
bins (optional): The number of bins (intervals) for grouping data. Default is 10.
color (optional): The fill color of the bars in the histogram.
edgecolor (optional): The color of the edges of the bars.

- Example values: 'black', 'white', 'gray'

Example:

```
import numpy as np

data = np.random.randn(1000)

plt.hist(data, bins=30, color='purple', edgecolor='black')
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.title("Histogram Example")
plt.show()
```

Pie Chart

Used to show the proportion of different categories.

Syntax:

```
plt.pie(sizes, labels, colors, autopct, startangle)
```

sizes: A list or array of numerical values representing the portions of the pie chart.
labels (optional): A list of category names corresponding to each slice.
colors (optional): A list of colors for each slice.

- Named colors: 'red', 'green', 'blue', etc.
- Hex codes: '#ff5733'
- RGB tuples: (0.5, 0.2, 0.8)

autopct (optional): A format string to display the percentage value on each slice.

- '%1.1f%%' → Displays one decimal place (e.g., 25.3%)
- '%d%%' → Displays as a whole number (e.g., 25%)

startangle (optional): The angle (in degrees) to start the first slice.

- 90 → Starts from the top
- 180 → Starts from the left

Example:

```
labels = ['A', 'B', 'C', 'D']
sizes = [40, 30, 20, 10]
colors = ['blue', 'red', 'green', 'yellow']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart Example")
plt.show()
```

Area Chart

Similar to a line chart but filled with color.

Syntax:

```
plt.fill_between(x, y, color, alpha)
```

alpha defines the **opacity level** of the filled area.

It takes values between 0 and 1:

- 0 → Fully transparent (invisible)
- 1 → Fully opaque (solid color)
- 0.5 → 50% transparency

Example:

```
x = [1, 2, 3, 4, 5]
y = [10, 20, 30, 40, 50]

plt.fill_between(x, y, color='lightblue', alpha=0.5)
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.title("Area Chart Example")
plt.show()
```

3. Customization in Matplotlib

Changing Figure Size

```
plt.figure(figsize=(width, height))
```

Example:

```
plt.figure(figsize=(8, 5))
plt.plot([1, 2, 3], [10, 20, 30])
plt.show()
```

Adding Grid

```
plt.grid(True)
```

Example:

```
plt.plot([1, 2, 3], [10, 20, 30])
plt.grid(True)
plt.show()
```

Adding Legends

```
plt.legend(["Label 1", "Label 2"])
```

Example:

```
plt.plot([1, 2, 3], [10, 20, 30], label="Line 1")
plt.legend()
plt.show()
```