What is Matplotlib?

Matplotlib is a powerful and widely-used Python library for creating static, animated, and interactive visualizations. It is especially useful for plotting 2D charts like line graphs, bar charts, pie charts, scatter plots, and more.

Why Use Matplotlib?

- Visualize trends, patterns, and relationships in data
- Customize every part of a chart (colors, labels, grid, ticks)
- Combine with NumPy, Pandas, Seaborn for data analysis
- Save charts in multiple formats (PNG, PDF, SVG, etc.)

Installation

pip install matplotlib

import matplotlib.pyplot as plt

plt.title("Chart with Grid")

- Purpose: Adds a title to the top of your plot.
- Why it's useful: Gives context to the chart — tells the viewer what they're looking at.

plt.xlabel("X Axis")

- Purpose: Adds a label to the horizontal axis (x-axis).
- Why it's useful: Explains what the values along the x-axis represent (e.g., time, categories).

plt.ylabel("Y Axis")

- **Purpose: Adds a label to the vertical axis (y-axis).**
- **Why it's useful: Explains what the values on the y-axis mean (e.g., sales, quantity).**

plt.grid(True)

- Purpose: Displays grid lines across the chart.
- Why it's useful: Helps the viewer align values visually with axis ticks for better readability.

Legend

A **legend** is a label guide that tells the reader **what each line, bar, or marker represents**.
It's automatically created using the `label` argument and displayed with `plt.legend()`.

1. Line Chart

➤ **Use Case: Showing trends over time**

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [10, 15, 12, 20, 18]
plt.plot(x, y,  color='blue', label="Sales") # use marker='o' adds circle markers on each (x, y) point.
plt.title("Line Chart")
plt.xlabel("Days")
plt.ylabel("Revenue")
plt.grid(True)
plt.legend()
plt.show()
```

---

2. Bar Chart

➤ **Use Case: Comparing categories**

```
categories = ['A', 'B', 'C', 'D']
values = [25, 40, 15, 30]
plt.bar(categories, values, color='skyblue')
plt.title("Bar Chart")
plt.xlabel("Category")
plt.ylabel("Value")
plt.grid(axis='y')
plt.show()
```

---

3. Histogram

➤ **Use Case: Distribution of numerical data**

```
import numpy as np
data = np.random.randint(10, 100, 50)
plt.hist(data, bins=10, color='orange', edgecolor='black')
plt.title("Histogram")
```

```
plt.xlabel("Value Range")
plt.ylabel("Frequency")
plt.grid(True)
plt.show()
```

---

4. Scatter Plot

➤ **Use Case: Relationships between two variables**

```
x = [5, 7, 8, 7, 2, 17, 2, 9, 4, 11]
y = [99, 86, 87, 88, 100, 86, 103, 87, 94, 78]
plt.scatter(x, y, color='purple')
plt.title("Scatter Plot")
plt.xlabel("X values")
plt.ylabel("Y values")
plt.grid(True)
plt.show()
```

---

5. Pie Chart

➤ **Use Case: Show parts of a whole**

```
labels = ['Python', 'Java', 'C++', 'Ruby']
sizes = [45, 30, 15, 10]
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
plt.title("Pie Chart")
plt.axis('equal')  # Equal aspect ratio makes the pie circular
plt.show()
```

---

6. Horizontal Bar Chart

```
python
CopyEdit
categories = ['X', 'Y', 'Z']
values = [50, 30, 60]
```

```
plt.barh(categories, values, color='teal')
plt.title("Horizontal Bar Chart")
plt.xlabel("Values")
plt.ylabel("Categories")
plt.grid(axis='x')
plt.show()
```

---

7. Multiple Lines in One Plot

```
x = [1, 2, 3, 4, 5]
y1 = [5, 10, 15, 10, 5]
y2 = [2, 6, 12, 18, 10]
plt.plot(x, y1, label='Product A', marker='o')
plt.plot(x, y2, label='Product B', marker='s')
plt.title("Multiple Line Plot")
plt.xlabel("Time")
plt.ylabel("Value")
plt.legend()
plt.grid(True)
plt.show()
```

Quick Summary of Chart Uses

Chart Type	**Best For**
Line	Time-based trends
Bar	Category comparison
Histogram	Frequency distribution
Scatter	Correlation between variables
Pie	Parts of a whole (percentages)
Horizontal Bar	Long category names
Multi-Line	Compare multiple trends together

Dataset: https://www.kaggle.com/datasets/ratnarohith/uncleaned-bike-sales-data

Firstly Clean the datasets before plotting…….

Line Plot:

```
import matplotlib.pyplot as plt

plt.plot(df['Date'].head(10), df['Revenue'].head(10), marker='o')

plt.title("Revenue of First 10 Days")

plt.xlabel("Date")

plt.ylabel("Revenue")

plt.grid(True)

plt.show()
```

Use Case: This line chart helps you visualize how revenue changed over the first 10 days of data.

Bar Chart:

```
plt.bar(df['Country'].head(5), df['Revenue'].head(5), color='skyblue')

plt.title("Revenue by First 5 Countries")

plt.xlabel("Country")

plt.ylabel("Revenue")

plt.xticks(rotation=30)  # rotate labels for better readability

plt.show()
```

Use Case: Use a bar chart to compare revenues from the top 5 countries in the dataset (first 5 rows).

Histogram:

```
plt.hist(df['Customer_Age'], bins=10, color='orange')

plt.title("Customer Age Distribution")

plt.xlabel("Age")

plt.ylabel("Frequency")

plt.grid(True)

plt.show()
```

Use Case: A histogram shows how customer ages are distributed—ideal for understanding customer demographics.

Scatter Plot:

```
plt.scatter(df['Revenue'].head(20), df['Profit'].head(20), color='green')

plt.title("Revenue vs Profit")

plt.xlabel("Revenue")

plt.ylabel("Profit")

plt.grid(True)

plt.show()
```

Introduction to Seaborn

Seaborn is a powerful Python library built on top of **Matplotlib**. It provides a high-level interface for drawing attractive and informative statistical graphics.

Install:

```
pip install seaborn
```

Import:

```
import seaborn as sns
```

---

2. Uses of Seaborn

- Creating beautiful **statistical plots** easily
- Exploring **relationships between variables**
- Plotting distributions of **numerical data**
- **Categorical comparisons** (bar, box, violin)
- **Built-in themes** and color palettes

---

3. ***Difference Between Seaborn and Matplotlib***

Feature	**Matplotlib**	**Seaborn**
Level	Low-level (manual setup)	High-level (simpler code)
Styling	Manual	Automatic (themes, palettes)
Statistical Plots	Not built-in	Built-in (bar, box, violin, etc.)
Categorical Handling	Needs grouping	Direct column usage

Built-in Datasets	No	Yes (sns.load_dataset())
Visual Appeal	Basic	Pre-styled and professional

Example 1: Bar Plot

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.set_theme(style="whitegrid")  # Optional styling
sns.barplot(x=df['Country'].head(5), y=df['Revenue'].head(5), palette="Blues")
plt.title("Revenue by First 5 Countries")
plt.xlabel("Country")
plt.ylabel("Revenue")
plt.xticks(rotation=30)
plt.show()
```

Use: Compare revenue for each country visually.

Example 2:

```
sns.lineplot(x=df['Date'].head(10), y=df['Revenue'].head(10), marker='o')
plt.title("Revenue Over First 10 Days")
plt.xlabel("Date")
plt.ylabel("Revenue")
plt.xticks(rotation=45)
plt.show()
```

Use: Analyze revenue trend over time.

Example 3: Histogram

```
sns.histplot(df['Customer_Age'], bins=10, kde=True, color="orange")
plt.title("Customer Age Distribution")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

Use: Understand the age group distribution of customers.

Example 4: Scatter Plot

```
sns.scatterplot(x=df['Revenue'].head(20), y=df['Profit'].head(20), color="green", s=100)
plt.title("Revenue vs Profit")
plt.xlabel("Revenue")
plt.ylabel("Profit")
plt.grid(True)
plt.show()
```

Use: Check how revenue and profit relate (e.g., correlation).

Example 5: Box Plot

```
sns.boxplot(x=df['Product_Category'].head(50), y=df['Profit'].head(50), palette="Set2")
plt.title("Profit Distribution by Product Category")
plt.xlabel("Product Category")
plt.ylabel("Profit")
plt.xticks(rotation=45)
plt.show()
```

Use: Spot outliers and spread in profit for each product category.

---

6. ***Count Plot – Frequency of Categories***

➤ **Use: How many times each category appears**

```
sns.countplot(x='Country', data=df, palette="pastel")
plt.title("Order Count by Country")
plt.xticks(rotation=45)
plt.show()
```

Practice Questions:

Matplotlib – Line, Bar, Histogram, Scatter, Pie

1. Line Plot

- Plot the sales of a product for 12 months.
- Add title, x and y labels, and grid lines.
- Highlight the highest month with a red marker.

2. Bar Chart

- Create a bar chart of 5 product categories and their total sales.
- Add value labels on top of each bar.
- Use a different color for each bar.

3. Horizontal Bar Chart

- Plot customer count by region using a horizontal bar chart.
- Sort regions from highest to lowest.

4. Histogram

- Plot a histogram of customer ages.
- Use 10 bins and set color to orange.
- Add KDE (if using Seaborn) for smooth curve.

5. Scatter Plot

- Create a scatter plot for Revenue vs Profit.
- Use different color and size for each point based on Order Quantity.

6. Pie Chart

- Show percentage share of sales for 4 product categories.
- Label each slice with both name and percentage.

7. Multiple Line Plot

- Plot revenue trends for 3 different stores over time in one graph.
- Use different colors and markers.
- Add legend, grid, and title.

B. Seaborn – Advanced & Statistical Plots

8. Bar Plot

- Use sns.barplot() to show average revenue by country.
- Use a light pastel palette.

9. Count Plot

- Show how many purchases happened on each weekday using countplot.

10. Box Plot

- Compare profit distribution across product categories.
- Highlight outliers.

Basic Menu-Based Chart Plotting

Take dataset from Kaggle

Write a Python program to display different types of charts using matplotlib.pyplot.

- Show a menu with options:
    1. Bar Chart
    2. Line Chart
    3. Pie Chart
    4. Scatter Plot
    5. Exit
- Ask the user to enter a number (1–4) to choose which chart to display.
- Use sample product sales data and show the chart accordingly.
- Loop the menu until the user chooses to exit.