🖥️ UNIT IV – Data Visualization using Matplotlib

🌈 1️⃣ Introduction to Data Visualization

💡 Definition:

> **Data Visualization** is the graphical representation of data to make information easier to understand and analyze.

It helps to:

- See **patterns**, **trends**, and **relationships**
- Make **comparisons**
- Understand **large datasets quickly**

💬 Why Visualization is Important

Reason	Description
📊 Simplifies Data	Converts large data into visuals
👀 Better Understanding	Helps us notice patterns easily
🧠 Decision Making	Aids data-driven decisions
🎯 Presentation	Makes reports more attractive and clear

🧰 2️⃣ Introduction to Matplotlib

💡 Definition:

> **Matplotlib** is a Python library used to create **static, animated, and interactive visualizations**.

It is one of the most widely used plotting libraries.

📦 Installing Matplotlib

Run this command in your terminal or notebook:

```
pip install matplotlib
```

💻 Importing the Library

```
import matplotlib.pyplot as plt
```

The module `pyplot` is most commonly used because it provides simple functions for plotting.

🧩 Structure of a Matplotlib Plot

A Matplotlib figure has several parts:

Part	Description
Figure	Overall window or page
Axes	The area where data is plotted
Axis	The X and Y lines
Title	The heading of the chart
Labels	Names for X and Y axes
Legend	Identifies different data lines
Grid	Helps in reading values easily

🧱 3️⃣ Line Plot

💡 Definition:

> A **Line Plot** represents data as a series of points connected by straight lines.

📈 **Use:** To show **trends or changes over time** (e.g., growth, sales, temperature).

🧠 General Syntax

```
plt.plot(x, y, color='blue', linestyle='-', marker='o')
plt.show()
```

💻 Example 1 – Simple Line Plot

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [5, 10, 15, 20, 25]

plt.plot(x, y)
plt.show()
```

🟢 A simple line connecting all data points.

💻 Example 2 – Adding Title and Labels

```
plt.plot(x, y)
plt.title("Simple Growth Chart")
plt.xlabel("Days")
plt.ylabel("Values")
plt.show()
```

📘 Adds clarity to the axes and chart meaning.

💻 Example 3 – Customizing Color, Marker, and Line

```
plt.plot(x, y, color='red', marker='o', linestyle='--', linewidth=2)
plt.title("Customized Line Plot")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.show()
```

🟩 You can customize:

- `color` – line color
- `marker` – style of point ('o', 's', '*', '^')
- `linestyle` – line type ('–', ':', '-.', '-')
- `linewidth` – thickness of line

💻 Example 4 – Multiple Lines in One Plot

```
x = [1, 2, 3, 4, 5]
y1 = [2, 4, 6, 8, 10]
y2 = [1, 3, 5, 7, 9]

plt.plot(x, y1, label='Even', color='blue', marker='o')
plt.plot(x, y2, label='Odd', color='green', marker='s')
plt.title("Multiple Lines Example")
plt.xlabel("Numbers")
plt.ylabel("Values")
plt.legend()
plt.show()
```

🟦 `plt.legend()` helps identify different lines.

💻 Example 5 – Adding Grid

```
plt.plot(x, y, 'r--')
plt.grid(True)
plt.title("Line Plot with Grid")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.show()
```

🟨 `plt.grid(True)` adds lines for easy reading.

💻 Example 6 – Using NumPy for Smooth Curves

```
import numpy as np

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

plt.plot(x, y, color='purple')
plt.title("Sine Wave")
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.show()
```

💻 Example 7 – Annotating a Point

```
plt.plot(x, y)
plt.annotate('Peak Point', xy=(5, 10), xytext=(3, 12),
             arrowprops=dict(facecolor='black', shrink=0.05))
plt.title("Annotated Line Plot")
plt.show()
```

🟧 `annotate()` helps label specific points.

🧠 Practice Exercises

1️⃣ Plot monthly sales of a company for 6 months.
2️⃣ Plot two lines: one showing income, one showing expenses.
3️⃣ Add title, labels, legend, and grid.
4️⃣ Use different colors and markers.
5️⃣ Annotate the maximum point on the graph.

🧱 4️⃣ Bar Graphs

💡 Definition:

> A **Bar Graph** represents categorical data with rectangular bars.

📊 **Use:** Compare quantities across different categories.

🧠 Syntax

```
plt.bar(x, height, color='blue', width=0.5)
```

💻 Example 1 – Simple Vertical Bar Graph

```
students = ['John', 'Sara', 'Ali', 'Priya']
marks = [85, 90, 78, 92]

plt.bar(students, marks)
plt.title("Marks of Students")
plt.xlabel("Students")
plt.ylabel("Marks")
plt.show()
```

💻 Example 2 – Change Colors

```
plt.bar(students, marks, color=['red', 'green', 'blue', 'purple'])
plt.show()
```

💻 Example 3 – Horizontal Bars

```
plt.barh(students, marks, color='orange')
plt.title("Horizontal Bar Graph")
plt.xlabel("Marks")
plt.ylabel("Students")
plt.show()
```

💻 Example 4 – Bar Width and Edge Color

```
plt.bar(students, marks, color='lightgreen', edgecolor='black', width=0.6
plt.title("Customized Bars")
plt.show()
```

💻 Example 5 – Adding Gridlines

```
plt.bar(students, marks, color='cyan')
plt.grid(True, axis='y')
plt.title("Bar Graph with Gridlines")
plt.show()
```

💻 Example 6 – Comparing Two Groups

```
import numpy as np
x = np.arange(len(students))
boys = [80, 85, 78, 90]
girls = [88, 92, 85, 95]

plt.bar(x - 0.2, boys, width=0.4, label='Boys')
plt.bar(x + 0.2, girls, width=0.4, label='Girls')
plt.xticks(x, students)
plt.xlabel("Students")
plt.ylabel("Marks")
plt.legend()
plt.title("Comparison of Marks")
plt.show()
```

💻 Example 7 – Adding Value Labels

```
bars = plt.bar(students, marks, color='lightblue')

for bar in bars:
    plt.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 1,
             str(bar.get_height()), ha='center')

plt.title("Bar Graph with Data Labels")
plt.show()
```

💻 Example 8 – Multiple Category Bars

```
departments = ['CSE', 'ECE', 'MECH']
boys = [50, 40, 30]
girls = [45, 35, 25]

x = np.arange(len(departments))
plt.bar(x - 0.2, boys, 0.4, label='Boys', color='blue')
plt.bar(x + 0.2, girls, 0.4, label='Girls', color='pink')
plt.xticks(x, departments)
plt.title("Department Strength")
plt.legend()
plt.show()
```

🧠 Practice Exercises

1️⃣ Create a bar graph showing the sales of 5 products.
2️⃣ Plot a horizontal bar chart showing population of 5 cities.
3️⃣ Compare two groups (2022 and 2023 sales).
4️⃣ Add data labels above each bar.
5️⃣ Experiment with colors, edge colors, and width.

🧱 5️⃣ Pie Charts

💡 Definition:

> A **Pie Chart** displays data in a circular graph divided into slices.

Each slice represents a **percentage** of the total.

🧠 Syntax

```
plt.pie(values, labels=names, autopct='%1.1f%%')
```

💻 Example 1 – Simple Pie Chart

```
activities = ['Sleep', 'Study', 'Play', 'Others']
hours = [8, 5, 4, 7]

plt.pie(hours, labels=activities)
plt.title("Daily Routine")
plt.show()
```

💻 Example 2 – Add Percentage

```
plt.pie(hours, labels=activities, autopct='%1.1f%%')
plt.show()
```

💻 Example 3 – Explode (Highlight Slice)

```
explode = [0.1, 0, 0, 0]
plt.pie(hours, labels=activities, autopct='%1.1f%%', explode=explode)
plt.show()
```

💻 Example 4 – Add Colors and Shadow

```
colors = ['gold', 'lightblue', 'lightcoral', 'lime']
plt.pie(hours, labels=activities, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=90)
plt.show()
```

💻 Example 5 – Donut Chart

```
plt.pie(hours, labels=activities, autopct='%1.1f%%', wedgeprops={'width':
plt.title("Donut Style Pie Chart")
plt.show()
```

💻 Example 6 – Pie Chart from Dictionary

```
data = {'Python': 40, 'Java': 25, 'C++': 20, 'Others': 15}
plt.pie(data.values(), labels=data.keys(), autopct='%1.1f%%')
plt.title("Programming Languages Popularity")
plt.show()
```

🧠 Practice Exercises

1️⃣ Create a pie chart for monthly expenses (Rent, Food, Transport, Savings).
2️⃣ Add percentages, colors, and explode largest slice.
3️⃣ Create a donut chart for time spent in a day.
4️⃣ Make a pie chart from dictionary data.
5️⃣ Try different start angles.

🎨 6️⃣ Customizing Plots

Feature	Function	Description
Title	`plt.title("Title")`	Adds a heading
Labels	`plt.xlabel("X")` , `plt.ylabel("Y")`	Describes axes
Legend	`plt.legend()`	Adds label box
Grid	`plt.grid(True)`	Adds background grid
Colors	`color='red'`	Changes color
Line Width	`linewidth=2`	Line thickness
Markers	`marker='o'`	Adds points
Annotations	`plt.annotate()`	Adds text with arrow

💻 Example – Fully Customized Plot

```
x = [1, 2, 3, 4, 5]
y = [5, 7, 9, 10, 12]

plt.plot(x, y, color='green', marker='^', linestyle='--', linewidth=2, la
plt.title("Growth Trend")
plt.xlabel("Years")
```

```
plt.ylabel("Growth Rate (%)")
plt.legend()
plt.grid(True)
plt.show()
```

🧠 Practice

Try customizing:

- Different marker shapes (`o` , `s` , `*` , `x`)
- Colors (`red` , `blue` , `orange`)
- Line styles (`--` , `-` , `:`)
- Add legends and grids
