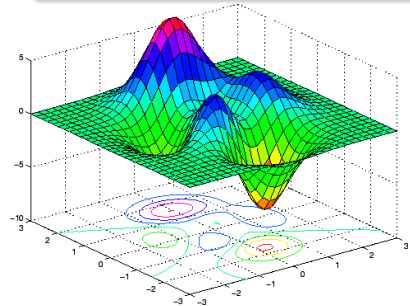
Matplotlib – Part 1 (Fundamentals)

Data Visualization Using Python



Assistant Lecturer Hadi Salah

Department of Artificial Intelligence

Python, NumPy, and Matplotlib: Workflow

1. **Python**:
   - General-purpose language.
   - Controls the program flow (if, for, functions).
2. **NumPy**:
   - Efficient numerical arrays.
   - Mathematical operations on vectors and matrices.
3. **Matplotlib**:
   - Turns numbers into figures and plots.
   - Visual explanation for students and researchers.

Checking Matplotlib Installation

Example: simple check that Matplotlib is working.

```
import matplotlib.pyplot as plt

print("Matplotlib Lecture Objectives Loaded")
# (No plot in this example, just checking import)
```

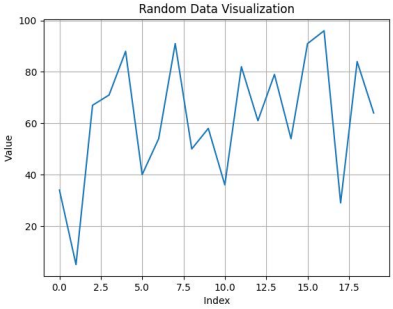
What is Data Visualization?

- Graphical representation of numerical data.
- Helps to understand trends, patterns, and outliers.
- Converts raw numbers into meaningful visuals.
- Essential in science, engineering, medicine, and AI.

Example: Visualizing Random Data

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

data = np.random.randint(0, 100, 20)
plt.plot(data)
plt.title("Random Data Visualization")
plt.xlabel("Index")
plt.ylabel("Value")
plt.grid(True)
plt.show()
```



Introduction to Matplotlib

- Matplotlib is a powerful plotting library in Python.
- Used for static, animated, and interactive visualizations.
- Integrates well with NumPy and Pandas.
- Acts as the base for other libraries such as Seaborn.

Checking the Matplotlib Version

```
import matplotlib
import matplotlib.pyplot as plt

print("Matplotlib version:", matplotlib.__version__)
```

Installing Matplotlib

Using `pip`:

- `pip install matplotlib`

Using Anaconda:

- `conda install matplotlib`

Google Colab:

- Matplotlib is usually pre-installed.

Importing Matplotlib (Step by Step)

1. Import NumPy and Matplotlib:

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

2. Create some data:

```
x = np.arange(0, 5)
y = x ** 2
```

3. Plot and show:

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

First Simple Plot: $y = x^2$

Basic workflow:

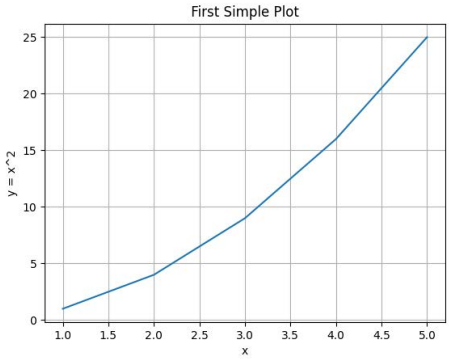
1. Define the input data x.
2. Compute the output y.
3. Call `plt.plot(x, y)`.
4. Call `plt.show()` to display.

Code: First Simple Plot

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

x = np.array([1, 2, 3, 4, 5])
y = x ** 2

plt.plot(x, y)
plt.xlabel("x")
plt.ylabel("y = x^2")
plt.title("First Simple Plot")
plt.grid(True)
plt.show()
```



Explaining the Code to Students

- `x = np.array([...])`: creates data points for the x-axis.
- `y = x ** 2`: applies the formula $y = x^2$ element-wise.
- `plt.plot(x, y)`: draws a line joining the points.
- `plt.xlabel / ylabel`: adds axis labels.
- `plt.title`: adds title to the figure.
- `plt.grid(True)`: adds background grid for clarity.

Components of a Plot

Main components:

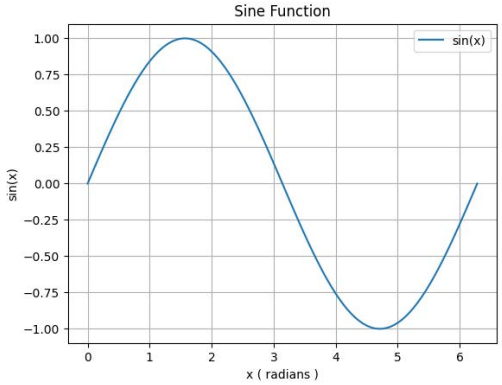
- **Title**: Describes what the graph represents.
- **X-axis label**: Name and unit of horizontal axis.
- **Y-axis label**: Name and unit of vertical axis.
- **Grid**: Helps to read exact values.
- **Legend**: Explains the plotted lines or markers.

Code: Sine Function with Components

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

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.plot(x, y, label="sin(x)")
plt.title("Sine Function")
plt.xlabel("x (radians)")
plt.ylabel("sin(x)")
plt.grid(True)
plt.legend()
plt.show()
```



Mini Exercise (In-Class)

Ask students to:

- Change the title text to any other phrase.
- Replace `sin(x)` with `cos(x)`.
- Change color of the line using `color="red"`.

This helps them to feel more comfortable editing small parts of code.

Line Plot

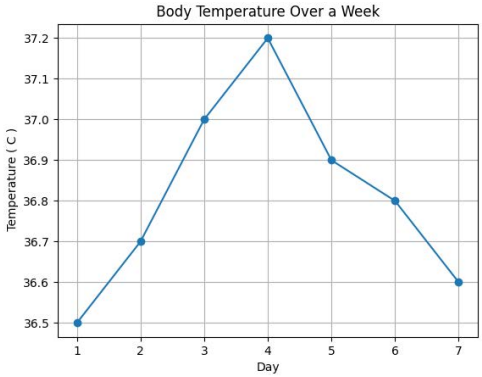
- Used for continuous data and trends.
- Common for time series (temperature, heart rate, etc.).
- X-axis usually represents time or ordered sequence.

Example: Body Temperature Over a Week

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

days = np.array([1, 2, 3, 4, 5, 6, 7])
temp = np.array([36.5, 36.7, 37.0, 37.2, 36.9, 36.8, 36.6])

plt.plot(days, temp, marker="o")
plt.xlabel("Day")
plt.ylabel("Temperature ( C )")
plt.title("Body Temperature Over a Week")
plt.grid(True)
plt.show()
```



Tips for Line Plots

- Use markers (`marker="o"`) when there are few data points.
- Use grid when teaching or explaining to students.
- Always label axes—never leave them without units.

Scatter Plot

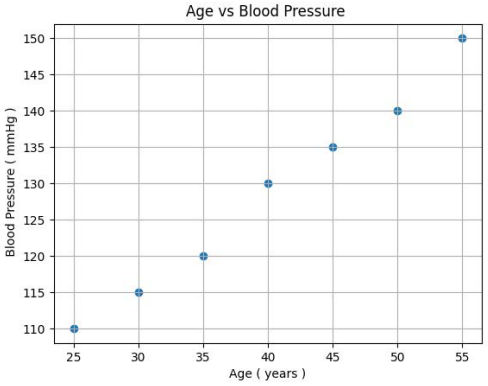
- Shows relationship between two variables.
- Each point represents one observation (x, y).
- Useful for correlation analysis and detecting outliers.

Example: Age vs Blood Pressure

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

age = np.array([25, 30, 35, 40, 45, 50, 55])
bp = np.array([110, 115, 120, 130, 135, 140, 150])

plt.scatter(age, bp)
plt.xlabel("Age (years)")
plt.ylabel("Blood Pressure (mmHg)")
plt.title("Age vs Blood Pressure")
plt.grid(True)
plt.show()
```



When to Use Scatter vs Line

- **Line plot**: data is ordered (e.g., time, increasing x).
- **Scatter plot**: data is just pairs (x, y) without order.
-

Bar Charts

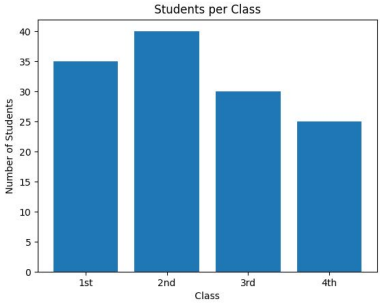
- Represent categorical data with rectangular bars.
- Height (or length) of each bar corresponds to its value.
- Can be vertical or horizontal.

Vertical Bar Chart: Students per Class

```
import matplotlib.pyplot as plt

classes = ["1st", "2nd", "3rd", "4th"]
students = [35, 40, 30, 25]

plt.bar(classes, students)
plt.xlabel("Class")
plt.ylabel("Number of Students")
plt.title("Students per Class")
plt.show()
```

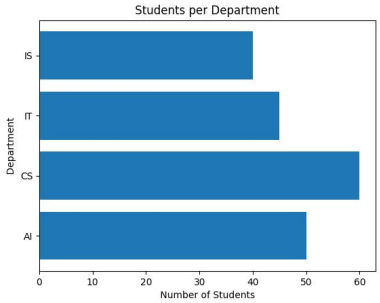


Horizontal Bar Chart: Students per Department

```
import matplotlib.pyplot as plt

departments = ["AI", "CS", "IT", "IS"]
students = [50, 60, 45, 40]

plt.barh(departments, students)
plt.xlabel("Number of Students")
plt.ylabel("Department")
plt.title("Students per Department")
plt.show()
```



Bar Plots: Teaching Tips

- Always sort categories logically (e.g., 1st, 2nd, 3rd, 4th).
- For many categories, horizontal bars are easier to read.
-

Histogram

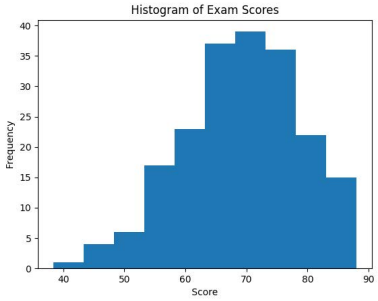
- Displays distribution of continuous data.
- Data is divided into equal-sized *bins*.
- Shows how many values fall into each bin.

Example: Histogram of Exam Scores

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

scores = np.random.normal(70, 10, 200)

plt.hist(scores, bins=10)
plt.xlabel("Score")
plt.ylabel("Frequency")
plt.title("Histogram of Exam Scores")
plt.show()
```



Choosing the Number of Bins

- Too few bins: plot hides important details.
- Too many bins: plot becomes noisy and confusing.
-

Grid and Saving Plots

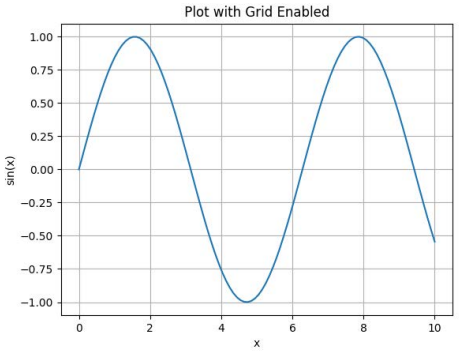
- Grid lines make reading values easier.
- Turn on with `plt.grid(True)`.
- Plots can be saved as PNG, JPG, PDF, etc.
- Use `plt.savefig("filename.png")` before `plt.show()`.

Example: Grid

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

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

plt.plot(x, y)
plt.grid(True)
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.title("Plot with Grid Enabled")
plt.show()
```



Example: Saving a Plot

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

x = np.linspace(0, 5, 50)
y = x ** 2

plt.plot(x, y)
plt.title("y = x^2")
plt.xlabel("x")
plt.ylabel("y")

plt.savefig("my_plot.png")  # Save figure to file
plt.show()
```

Common Errors

- Using `import matplotlib` but forgetting `import matplotlib.pyplot as plt`.
- Arrays `x` and `y` have different lengths.
- Forgetting `plt.show()` (nothing appears).
- Misspelling function names, e.g., `plt.plt()` instead of `plt.plot()`.

Example of a Length Error

```
import matplotlib.pyplot as plt

x = [1, 2, 3]
y = [1, 4]  # shorter!

plt.plot(x, y)  # will raise ValueError: x and y must have same
    first dimension
plt.show()
```

Simple Medical Application

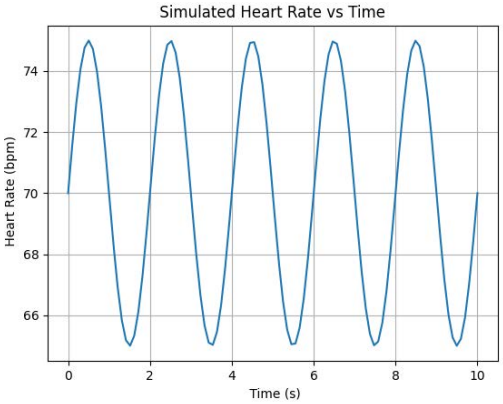
- Simulate heart rate measurement over time.
- Visualize a normal heart rate pattern.

Example: Simulated Heart Rate vs Time

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

t = np.linspace(0, 10, 100)
hr = 70 + 5 * np.sin(2 * np.pi * t / 2)

plt.plot(t, hr)
plt.xlabel("Time (s)")
plt.ylabel("Heart Rate (bpm)")
plt.title("Simulated Heart Rate vs Time")plt.grid
(True)
plt.show()
```



Homework (Practice)

1. Plot $y = x^3$ for x from -5 to 5 (line plot).
2. Draw a bar chart of 5 subjects and their marks.
3. Create a histogram of 100 random values from a normal distribution.
4.