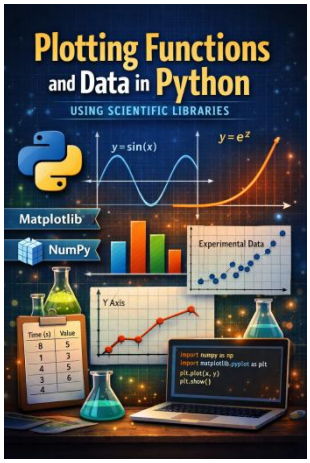
Chapter: 6

Plotting Functions and Data in Python Using Scientific Libraries



2. Learning Objectives

At the end of this course, students should be able to:

- understand the importance of graphical representation in science,
- use the Matplotlib library,
- plot mathematical functions,
- plot data stored in arrays,
- customize graphs (title, labels, grid, legend),
- plot multiple curves on the same graph,
- visualize experimental data.

3. Introduction

Graphical representation is essential in scientific analysis. It allows us to:

- visualize trends,
- compare results,
- interpret experimental data,
- validate mathematical models.

In Python, the most commonly used library for plotting is:

Matplotlib

It is often used together with:

NumPy

4. Required Libraries

Before plotting, we must import the necessary libraries.

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

5. Basic Structure of a Plot

The simplest plot in Python follows these steps:

```
import matplotlib.pyplot as plt
plt.plot(x, y)
plt.show()
```

Where:

- x → values on the x-axis
- y → values on the y-axis

6. Plotting Data from Arrays

6.1. Example 1: Simple Plot

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([1, 2, 3, 4, 5])
y = np.array([1, 4, 9, 16, 25])
plt.plot(x, y)
plt.show()
```

This plots the function: $y=x^2$

6.2. Adding Labels and Title

```
plt.plot(x, y)
plt.title("Graph of y = x^2")
plt.xlabel("x values")
plt.ylabel("y values")
plt.show()
```

6.3. Adding Grid

plt.grid()

7. Plotting Mathematical Functions

Instead of manually writing arrays, we can generate them using **NumPy**.

7.1. Example: Plot $y = x^2$

```
y=x^2
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-5, 5, 100)
y = x**2
plt.plot(x, y)
plt.title("y = x^2")
plt.xlabel("x")
plt.ylabel("y")
plt.grid()
plt.show()
```

7.2. Example: Plot y = sin(x)

```
y=sin(x)
x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)
plt.plot(x, y)
plt.title("y = sin(x)")
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.grid()
plt.show()
```

7.3. Example: Plot y = exp(x)

```
y=e^x
x = np.linspace(0, 5, 100)
y = np.exp(x)
plt.plot(x, y)
plt.title("y = exp(x)")
plt.grid()
plt.show()
```

8. Plotting Multiple Functions on the Same Graph

Example: sin(x) and cos(x)

```
x = np.linspace(0, 2*np.pi, 100)
y1 = np.sin(x)
y2 = np.cos(x)
plt.plot(x, y1, label="sin(x)")
plt.plot(x, y2, label="cos(x)")
plt.title("sin(x) and cos(x)")
plt.legend()
plt.grid()
plt.show()
```

9. Customizing the Plot

9.1. Line Style and Markers

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

9.2. Colors

```
plt.plot(x, y, color='red')
```

9.3. Leend

```
plt.legend()
```

10. Plotting Data from Experimental Tables

This is very important for **chemistry and physics students**.

Example: Experimental Data

```
x = np.array([0, 1, 2, 3, 4])
y = np.array([0, 2.1, 4.2, 6.1, 8.3])
plt.plot(x, y, marker='o')
plt.title("Experimental Data")
plt.xlabel("Time (s)")
plt.ylabel("Concentration")
plt.grid()
plt.show()
```

11. Scatter Plot (Important for Experiments)

```
plt.scatter(x, y)
plt.title("Scatter Plot")
```

```
plt.grid()
plt.show()
```

12. Bar Plot

```
categories = ["A", "B", "C"]
values = [10, 20, 15]
plt.bar(categories, values)
plt.title("Bar Chart")
plt.show()
```

13. Saving a Graph

```
plt.savefig("graph.png")
```

14. Complete Example

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-2, 2, 100)
y = x**3
plt.plot(x, y, label="y = x^3")
plt.title("Cubic Function")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.grid()
plt.show()
```

15. Solved Exercises

Exercise 1

Plot the function: $y=x^3$

Solution

```
x = np.linspace(-5, 5, 100)
y = x**3
plt.plot(x, y)
plt.grid()
plt.show()
```

Exercise 2

Plot the function: y=x

Solution

```
x = np.linspace(0, 10, 100)
y = np.sqrt(x)
plt.plot(x, y)
plt.show()
```

Exercise 3

Plot experimental data using a scatter plot.

Solution

```
x = [1, 2, 3, 4]
y = [2.2, 2.8, 3.6, 4.5]
plt.scatter(x, y)
plt.show()
```

Conclusion

In this course, students learned how to:

- plot mathematical functions,
- plot data from arrays,
- use the **Matplotlib** library,
- visualize experimental results,
- customize graphs.

Graphical visualization is a fundamental tool in:

- chemistry,
- physics,
- engineering,
- data analysis.