# Chapter 5: Introduction to Python Libraries

## Learning Objectives

By the end of this chapter, the student will be able to:

- Understand the role of libraries in Python
- Use **NumPy** to efficiently manipulate numerical arrays
- Perform **vectorized operations**
- Create and customize plots using **Matplotlib**

## 5.1 Introduction to package NumPy

### 1. Introduction

NumPy (**Numerical Python**) is an essential library for scientific computing.

It allows you to:

- Work with multidimensional arrays (**ndarray**)
- Perform fast computations
- Replace loops with vectorized operations

### 2. Installation (if needed)

pip install numpy

### 3. Importing NumPy

```
import numpy as np
```

### 4. Creating arrays

**a) Simple array**

```
import numpy as np
a = np.array([1, 2, 3, 4])
print(a)
```

**b) 2D array (matrix)**

```
import numpy as np
b = np.array([[1, 2], [3, 4]])
print(b)
```

## 5. Array properties

```
import numpy as np
a = np.array([1, 2, 3, 4])
print(a.shape)   # dimensions
print(a.dtype)   # data type    int64
print(a.size)    # number of elements   4
```

## 6. Creation functions (NumPy)

```
import numpy as np
```

**a) np.zeros((2, 3))**

Creates a matrix filled with **zeros**

```
a=np.zeros((2, 3))
print(a)
```

Result:

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

✔2 rows, 3 columns

✔All elements = 0

**b) np.ones((3, 3))**

Creates a matrix filled with **ones**

```
a=np.ones((3, 3))
print(a)
```

Result:

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

✔3 × 3

✔All elements = 1

**c) np.eye(3)**

Creates an **identity matrix**

```
a=np.eye(3)
print(a)
```

Result:

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

✔Diagonal elements = 1

✔Others = 0

**d) np.arange(0, 10, 2)**

Creates a sequence of numbers (like range)

```
a=np.arange(0, 10, 2)
print(a)
```

Result:

```
[0 2 4 6 8]
```

✔Start = 0

✔End = 10 (not included)

✔Step = 2

**e) np.linspace(0, 1, 5)**

Creates **evenly spaced values**

```
a=np.linspace(0, 1, 5)
print(a)
```

Result:

```
[0.   0.25 0.5  0.75 1.  ]
```

✔Start = 0

✔End = 1

✔Number of values = 5

**Simple Summary**

| Function | Purpose |
|---|---|
| zeros() | array of zeros |
| ones() | array of ones |
| eye() | identity matrix |
| arange() | sequence with step |
| linspace() | evenly spaced values |

**Important Tip**

**arange vs linspace**

- arange → you choose the **step**
- linspace → you choose the **number of values**

## 7. Accessing elements

```
a = np.array([10, 20, 30])

print(a[0])
print(a[-1])
```

## 8. Vectorized operations

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print(a + b)
print(a * b)
print(a ** 2)
```

## 9. Mathematical functions

```
np.sqrt(a)
np.sum(a)
```

np.mean(a)
np.max(a)

### 10. Comparison with Python lists

| Python List | NumPy |
|---|---|
| Requires loops | No loops needed |
| Slower | Faster |
| Less practical | More powerful |

## 5.2 Introduction to Matplotlib

### 1. Introduction

Matplotlib is used to:

- Visualize data
- Create professional graphs

### 2. Installation

```
pip install matplotlib
```

### 3. Importing Matplotlib

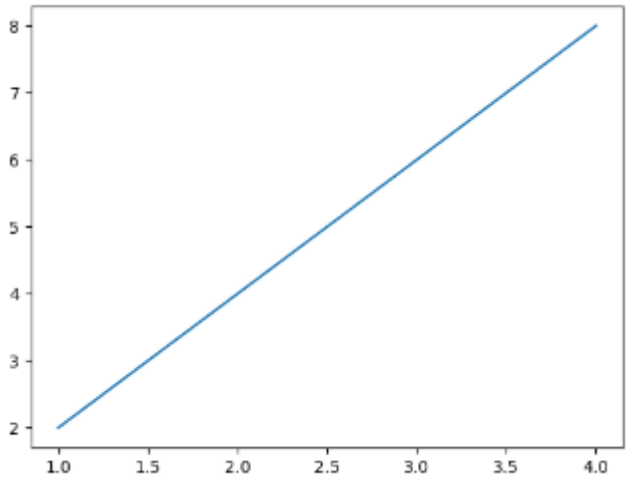
```
import matplotlib.pyplot as plt
```

### 4. Line plot

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [2, 4, 6, 8]

plt.plot(x, y)    # To plot a curve (with the x-axis horizontal and the y-axis vertical)
plt.show()        # It allows displaying the graph.
```




## 5. Scatter plot

```
plt.scatter(x, y)  # Plot a scatter graph (par point)
plt.show()
```

## 6. Histogram

```
data = [1, 2, 2, 3, 3, 3, 4]

plt.hist(data)  # Plot (trace)a histogram
plt.show()
```

## 7. Customization

```
plt.plot(x, y)
plt.title("My Graph")  # Add a title to the graph
plt.xlabel("X Axis")  # Label the horizontal axis (X)
plt.ylabel("Y Axis")  # Label the vertical axis (Y)
plt.legend(["Curve"]) # Provide a legend to explain the curves
plt.grid()#Makes the graph more readable
plt.show()
```

## 8. Multiple curves

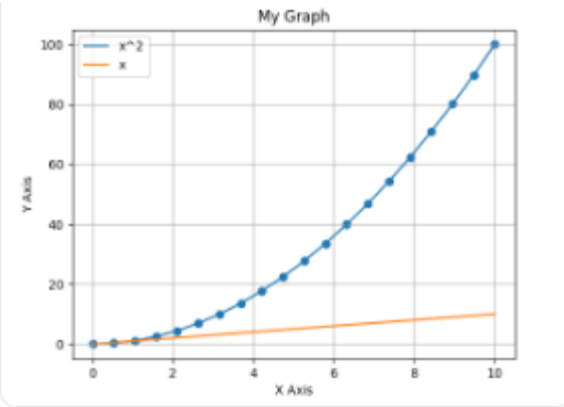
```
x = [1, 2, 3, 4]
y1 = [1, 4, 9, 16]
y2 = [1, 2, 3, 4]
plt.plot(x, y1, label="x^2")# Plot a first curve.
```

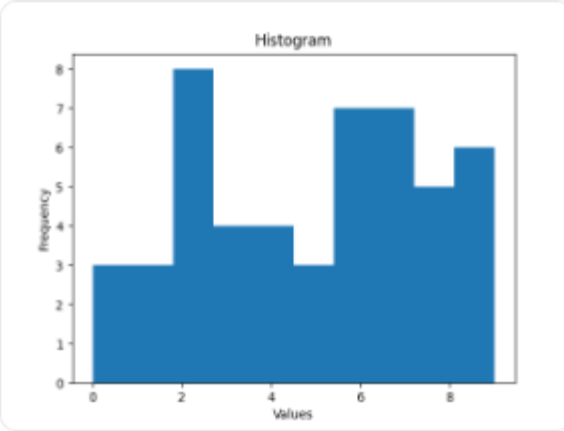
```
                                                    #    label = "x²" → name in the
legend
plt.plot(x, y2, label="x")  #  Plot a second curve .
                                                  #    Allows comparison of multiple
functions
plt.legend()#Displays the legend based on the labels.
plt.show()
```

## 9. Complete example (NumPy + Matplotlib)

```
import numpy as np
import matplotlib.pyplot as plt
# Données
x = np.linspace(0, 10, 20)
y1 = x**2
y2 = x

data = np.random.randint(0, 10, 50)
# Courbes
plt.plot(x, y1, label="x^2")
plt.plot(x, y2, label="x")
# Points
plt.scatter(x, y1)
# Histogramme (dans une autre figure)
plt.figure()
plt.hist(data)
plt.title("Histogram")
plt.xlabel("Values")
plt.ylabel("Frequency")
# Retour à la première figure
plt.figure(1)
# Personnalisation
plt.title("My Graph")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.legend()
plt.grid()
# Affichage
plt.show()
```




## Summary

| Library | Purpose |
|---|---|
| NumPy | Numerical computing |
| Matplotlib | Data visualization |

## Teaching Tips

✔Always import as np and plt

✔Use NumPy to avoid loops

✔Always end with plt.show()