Chapter: 5: Arrays in Python

Scientific Programming Course

1. Introduction

In scientific programming, we often need to store and manipulate many values of the same kind, such as:

- temperatures,
- pressures,
- concentrations,
- velocities,
- experimental data,
- numerical results.

Instead of creating many separate variables like this:

x1 = 2.1
x2 = 3.4
x3 = 5.8
x4 = 1.9

it is much more efficient to use a **single structure** that can hold multiple values:

x = [2.1, 3.4, 5.8, 1.9]

This structure is called an **array** (or sometimes a **list**, depending on the context).

Arrays are essential in:

- mathematics,
- physics,
- chemistry,
- engineering,
- data analysis,
- numerical simulation.

2. Definition of an Array

An **array** is a data structure used to store multiple values under a single variable name.

These values are usually:

- ordered,
- accessible by position (index),
- often of the same type in scientific applications.

Examples:

- **1D array** → vector
- **2D array** → matrix
- **3D array** → data cube

3. Arrays in Python: Two Main Approaches

In Python, arrays can be handled in two main ways:

3.1. Python Lists

Lists are the standard built-in sequence structure in Python.

Example:

```
A = [10, 20, 30, 40]
```

3.2. NumPy Arrays

For scientific computing, Python mainly uses the library NumPy.

Example:

```
import numpy as np
A = np.array([10, 20, 30, 40])
```

Important note:

- **Lists** are simple and useful for beginners.
- **NumPy arrays** are faster, more powerful, and more suitable for scientific work.

Therefore, in this course we will study:

1. **Python lists**
2. **NumPy arrays**

PART I — PYTHON LISTS

4. Creating an Array (List)

4.1. Direct Declaration

```
A = [1, 2, 3, 4, 5]
print(A)
```

Output:

[1, 2, 3, 4, 5]

4.2. Array of Strings

names = ["Ali", "Sara", "Nadia"]
print(names)

4.3. Mixed List (Possible in Python, but not recommended for scientific programming)

B = [1, 2.5, "Hello", True]
print(B)

In scientific computing, we usually prefer **homogeneous arrays**, meaning arrays containing values of the same type.

5. Accessing Elements of an Array

In Python, indexing starts at **0**.

Example

A = [10, 20, 30, 40]
print(A[0])
print(A[1])
print(A[2])

Output:

10
20
30

Comparison with Fortran

In **Fortran**, we often write:

A(1), A(2), A(3)

In **Python**, we write:

A[0], A[1], A[2]

Very important:

- **Fortran often starts at 1**
- **Python starts at 0**

6. Modifying an Element

```
A = [10, 20, 30, 40]
A[2] = 99
print(A)
```

Output:

[10, 20, 99, 40]

7. Length of an Array

In Python, the length of a list is obtained using:

len(A)

Example

```
A = [10, 20, 30, 40]
print(len(A))
```

Output:

4

This gives the **number of elements** in the array.

8. Traversing an Array

8.1. Using a for loop with indices

```
A = [10, 20, 30, 40]

for i in range(len(A)):
    print("A[", i, "] =", A[i])
```

Output:

```
A[ 0 ] = 10
A[ 1 ] = 20
A[ 2 ] = 30
A[ 3 ] = 40
```

8.2. Direct Traversal

```
A = [10, 20, 30, 40]
for x in A:
    print(x)
```

9. Filling an Array

9.1. By User Input

```
n = int(input("Enter the size of the array: "))
A = []
for i in range(n):
    val = float(input(f"Enter A[{i}] : "))
```

```
    A.append(val)
print("Array =", A)
```

9.2. Initializing an Array with the Same Value

```
n = 5
A = [0] * n
print(A)
```

Output:

```
[0, 0, 0, 0, 0]
```

10. Basic Operations on Arrays

10.1. Sum of Elements

```
A = [1, 2, 3, 4, 5]
s = sum(A)
print("Sum =", s)
```

Output:

```
Sum = 15
```

10.2. Maximum and Minimum

```
A = [5, 8, 2, 10, 3]
print("Max =", max(A))
print("Min =", min(A))
```

10.3. Average

```
A = [10, 20, 30, 40]
avg = sum(A) / len(A)
print("Average =", avg)
```

11. Searching in an Array

```
A = [4, 7, 9, 2, 5]
x = 9
if x in A:
    print("Value found")
else:
    print("Value not found")
```

12. Sorting an Array

```
A = [5, 2, 8, 1, 9]
A.sort()
print(A)
```

Output:

[1, 2, 5, 8, 9]

13. Reversing an Array

```
A = [1, 2, 3, 4, 5]
A.reverse()
print(A)
```

Output:

[5, 4, 3, 2, 1]

14. Subarrays (Slicing)

Slicing allows us to extract part of an array.

Example

```
A = [10, 20, 30, 40, 50, 60]
print(A[1:4])
```

Output:

[20, 30, 40]