

# *Python NumPy library*

Abdesselam FILALI

February 5, 2026

## 1 What is NumPy?

NumPy (Numerical Python) is a powerful Python library used for mathematical and numerical operations. It provides a high-performance multidimensional array object and tools for working with these arrays.

### 1.1 Why use NumPy?

- Much faster than Python lists for numerical tasks.
- Used as the foundation for libraries like Pandas, TensorFlow, and Scikit-learn.
- Makes matrix operations, statistics, and data transformations easy and efficient.

### 1.2 Installation

```
pip install numpy
```

## 2 Data Types in NumPy

*NumPy* has some extra data types, and refer to data types with one character, like *i* for integers, *u* for unsigned integers etc.

Below is a list of all data types in NumPy and the characters used to represent them.

- *i*: integer
- *b*: boolean
- *u*: unsigned integer
- *f*: float
- *c*: complex float
- *m*: timedelta
- *M*: datetime
- *O*: object
- *S*: string
- *U*: unicode string
- *V*: fixed chunk of memory for other type (void)

## 3 What is a NumPy Array?

A NumPy array is a powerful data structure that stores elements of the same data type in a grid-like format.

It's much faster and more efficient than Python lists for numerical operations.

```
import numpy as np

np.array([1, 2, 3, 4, 5])           # 1D array from a list
np.array([[1, 2, 3], [4, 5, 6]])    # 2D array from a list of lists
np.zeros((2, 3))                    # 2x3 array of zeros
np.ones((3, 3))                     # 3x3 array of ones
np.empty((2, 3))                    # 2x3 array of uninitialized values
np.arange(0, 10, 2)                  # array of values from 0 to 10, step 2
```

## 4 Basic array operations

### 4.1 1. Indexing & Slicing

```
arr = np.array([1, 2, 3, 4, 5])
arr[0]           # Get the first element
arr[2:4]         # Get elements from index 2 to 4 (exclusive)
arr[::-2]        # Get every second element
```

### 4.2 2. Shape, dimensions & Reshaping

```
arr = np.array([
    [1, 2, 3],
    [4, 5, 6]])
arr.shape        # Get the shape of the array (rows, columns)
arr.ndim         # Get the number of dimensions (2)
arr.size         # Get the total number of elements (6)
arr.reshape((3, 2)) # Reshape the array to 3 rows and 2 columns
```

### 4.3 3. Data types

```
arr = np.array([1, 2, 3, 4, 5])
arr.dtype        # Get the data type of the array elements (int64)
arr.astype(float) # Convert the array elements to float
```

## 5 Mathematical operations

NumPy makes math easy — no need for loops! Just write it once and it works on the entire array.

### 5.1 1. Element-wise operations

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
a + b          # Element-wise addition
```

```

a - b          # Element-wise subtraction
a * b          # Element-wise multiplication
a / b          # Element-wise division
a ** b         # Element-wise exponentiation

```

## 5.2 2. Array aggregate functions

```

a = np.array([1, 2, 3, 4, 5])
np.sum(a)      # Sum of all elements
np.mean(a)     # Mean of all elements
np.max(a)      # Maximum value
np.min(a)      # Minimum value
np.std(a)      # Standard deviation

```

# 6 Advanced array operations

## 6.1 1. Broadcasting

Broadcasting allows NumPy to perform element-wise operations on arrays of different shapes.

```

a = np.array([[1, 2, 3], [4, 5, 6]])
b = np.array([10, 20, 30])
a + b          # Add b to each row of a

```

## 6.2 2. Linear algebra

```

a = np.array([[1, 2], [3, 4]])
np.linalg.inv(a) # Inverse of a
np.linalg.det(a) # Determinant of a
np.linalg.eig(a) # Eigenvalues and eigenvectors of a

```

## 6.3 3. Random numbers

```

np.random.rand(3, 3) # 3x3 array of random numbers between 0 and 1
np.random.randint(0, 10, size=(3, 3)) # 3x3 array of random integers between 0 and 10

```

## 6.4 4. Boolean indexing

```

arr=np.array(5,10,15, 20, 3)
print(arr[arr > 10]) # output [15 20]

```

## 6.5 5. Matrix manipulation

```

a = np.array([[1,2],[3,4]])
b = np.array([[5,6],[7,8]])
print(np.dot(a, b)) # output [[19 22] [43 50]]

```

## 6.6 6. flatten an array

```

arr = np.array([[1,2],[3,4]])
print(arr.flatten()) # output [1 2 3 4]

```

## 7 Conclusion

NumPy is a powerful library for numerical computing in Python. It provides a fast and efficient way to work with arrays and perform mathematical operations on them. Whether you're a data scientist, a machine learning engineer, or just someone who wants to do some serious number crunching, NumPy is a must-have tool in your toolkit.