

# Numpy Handbook for Ai

*Ashraful Islam Mahi*

*Full Stack Python Developer | ML & Robotics Enthusiast*



## Understanding the Role of Numpy in Data Science & Machine Learning.

---

### What is Numpy?

- Numpy or 'Numerical Python' is a mathematical library of Python for numerical calculation. It provides fast and efficient ways to perform numerical operations on large datasets using Arrays and supports advanced mathematical functions.

### Why use Numpy in Ai?

- Performance: Numpy arrays are more efficient than python lists.
- Ease of Use: Builtin operations for mathematical and linear algebra functions.
- Integration: Works seamlessly with the libraries like 'Pandas', 'Matplotlib', 'Scikit-learn', 'Tensorflow', 'Pytorch' etc.

## Creating and Manipulating Numpy Arrays

---

### 0. Installing Numpy Library

```
In [ ]: %pip install numpy
```

### 1. Importing Numpy Library

```
In [3]: import numpy as np
```

## 2. Creating Numpy Array

```
In [5]: # step-1: Create an list first
list = [1,2,3,4,5]
# step-2: turn that list into numpy array
arr = np.array(list)

print(arr)
```

```
[1 2 3 4 5]
```

## 3. Build a matrix full of zeros

```
In [ ]: #create a 3*3 matrix of zeros
dimention = (3,3)
Zeros = np.zeros(dimention)

print(Zeros)
```

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

## 4. Build a matrix full of ones

```
In [ ]: #create a 2*4 matrix of ones
dimention = (2,4)
Ones = np.ones(dimention)

print(Ones)
```

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

## 5. Create an array with "arange" function

```
In [15]: #Create array using arange

range_array = np.arange(1,10,2) #start , end+1, step
print(range_array)
```

```
[1 3 5 7 9]
```

## 6. linspace

```
In [ ]: linspace_array = np.linspace(0,1,5) #start, end , step
print(linspace_array)
```

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

## 7. Array Reshaping

```
In [ ]: list = [1,2,3,4,5,6,7,8,9]
arr = np.array(list)
dimention = (3,3)
reshaped_arr = arr.reshape(dimention)

print(reshaped_arr)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

## 8. Array Newaxis

```
In [22]: list = [1,2,3,4,5,6,7,8,9]
arr = np.array(list)
expanded_arr = arr[:,np.newaxis]
print(expanded_arr)
```

```
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]
 [9]]
```

# Basic Operations of Numpy Arrays

---

## 1. Elementwise operation of arrays

```
In [5]: arr_1 = np.array([1,2,3,4])
arr_2 = np.array([1,4,9,16])

print(f"Sum:{arr_1+arr_2}")
print(f"Subs:{arr_2-arr_1}")
print(f"Mul:{arr_2*arr_1}")
print(f"Div:{arr_2/arr_1}")
```

```
Sum:[ 2  6 12 20]
Subs:[ 0  2  6 12]
Mul:[ 1  8 27 64]
Div:[1.  2.  3.  4.]
```

## 2. Mathematical Operations of arrays:

```
In [9]: arr_1 = np.array([1,2,3,4])
arr_2 = np.array([1,4,9,16])

print(f"Square root of all elements of arr_2:{np.sqrt(arr_2)}")
print(f"Sum of all elements of arr_2:{np.sum(arr_2)}")
print(f"Max of all elements of arr_1:{np.max(arr_1)}")
```

Square root of all elements of arr\_2:[1. 2. 3. 4.]  
Sum of all elements of arr\_2:30  
Max of all elements of arr\_1:4

## Numpy Array Indexing & Slicing

---

### 1. Indexing

```
In [11]: arr = np.array([1,2,3,4,5])

print(arr[2])
print(arr[-2])
```

3  
4

### 2. Slicing

```
In [12]: arr = np.array([1,2,3,4,5])

print(arr[1:4])
```

[2 3 4]

## Matrix Creation & Operations in Numpy

---

### 1. Matrix Creation

```
In [19]: matrix = np.array([[1,2,3],
                             [4,5,6],
                             [7,8,9]])

print(f"Matrix: \n {matrix}")
```

Matrix:  
[[1 2 3]  
[4 5 6]  
[7 8 9]]

### 2. Determinant of a Matrix

```
In [22]: matrix = np.array([[2,3],
                             [4,5]])

# Calculate the determinant
det = np.linalg.det(matrix)

print(f"Determinant:\n {det}")
```

Determinant:  
-2.0

### 3. Transpose operation of Matrix

```
In [23]: matrix = np.array([[1,2,3],
                             [4,5,6],
                             [7,8,9]])

transpose_matrix = matrix.T
print(f"Transpose Matrix:\n {transpose_matrix}")
```

Transpose Matrix:  
[[1 4 7]  
[2 5 8]  
[3 6 9]]

### 4. Inverse Matrix Calculation

```
In [25]: matrix = np.array([[4, 7],
                             [2, 6]])

if np.linalg.det(matrix) != 0:
    inverse_matrix = np.linalg.inv(matrix)
    print("Inverse Matrix:")
    print(inverse_matrix)
else:
    print("The matrix is singular and cannot be inverted.")
```

Inverse Matrix:  
[[ 0.6 -0.7]  
[-0.2 0.4]]

## Broadcasting in Numpy

---

### What is Broadcasting?

- Broadcasting helps to perform operation between arrays of different shapes. Smaller arrays are automatically expanded to the large array.

### Rules of Broadcasting:

- Dimensions are aligned from the right.
- A dimension is compatible if:
  1. It matches the other array's dimension.
  2. One of the dimensions is 1

```
In [31]: matrix_1 = np.array([[1,2,3],
                              [4,5,6]])

matrix_2 = np.array([1,0,1])

summation = matrix_1 + matrix_2

print(f"Summation:\n {summation}")
```

Summation:

```
[[2 2 4]
 [5 5 7]]
```

## Aggregation Functions

---

- Aggregation functions compute summary statistics for arrays

```
In [39]: matrix = np.array([[1,2,3],
                             [4,5,6]])

print(f"Sum: {np.sum(matrix)}") #Total sum of all elements of matrix
print(f"Mean: {np.mean(matrix)}")
print(f"Max: {np.max(matrix)}")
print(f"Min: {np.min(matrix)}")
print(f"Standard Daviation: {np.std(matrix):.2f}")
print(f"Sum along columns: {np.sum(matrix,axis=0)}")
print(f"Sum along rows: {np.sum(matrix,axis=1)}")
```

```
Sum: 21
Mean: 3.5
Max: 6
Min: 1
Standard Daviation: 1.71
Sum along columns: [5 7 9]
Sum along rows: [ 6 15]
```

## Broadcasting & Filtering

---

### What is Boolean Indexing?

- You'll applied boolean arrays to filter elements from an array.

#### 1. Filtering Arrays with boolean condition

```
In [45]: arr = np.array([1,2,3,4,5,6])

evens = arr[arr % 2 == 0]
print(f"Evens: {evens}")
```

```
Evens: [2 4 6]
```

#### 2. Modifying Arrays with boolean condition

```
In [49]: arr = np.array([1,2,3,4,5,6])

arr[arr > 3] = 0
print(f"Modified Array: {arr}")
```

```
Modified Array: [1 2 3 0 0 0]
```

## Random Number Generation & Setting Seeds

---

### 1. Random Array Generation:

```
In [50]: random_matrix = np.random.rand(3,3)
print(f"Random Matrix:\n {random_matrix}")
```

```
Random Matrix:
[[0.13870713 0.62088144 0.41554576]
 [0.81084632 0.42972827 0.99144259]
 [0.94222085 0.76331568 0.86548507]]
```

### 2. Random Integer Array with certain range

```
In [52]: random_integer = np.random.randint(0,10,size=(2,3))
print(f"Random Integer: \n {random_integer}")
```

```
Random Integer:
[[1 6 7]
 [2 6 0]]
```

### 3. Setting Random Seeds (Always keep the random values same)

```
In [60]: np.random.seed(1)

random_integer = np.random.randint(0,10,size=(2,3))
print(f"Random Integer: \n {random_integer}")
```

```
Random Integer:
[[5 8 9]
 [5 0 0]]
```