## 💡 What is NumPy?

**NumPy** (Numerical Python) is a powerful Python library used to perform **fast mathematical operations** on **arrays**. It is especially popular in **data science**, **machine learning**, **scientific computing**, and **engineering**.

## 🚀 Why is NumPy Fast?

1. **Written in C**: Internally, NumPy uses C language which is much faster than Python.
2. **Vectorization**: Operates on entire arrays without loops (e.g., arr \* 2 instead of a for loop).
3. **Contiguous Memory**: Stores data in one block, improving speed.
4. **Fixed Data Type**: All elements in a NumPy array are of the same type.
5. **Multithreading**: Some operations use multiple CPU cores.

## 🔹 How to Install and Import

pip install numpy

import numpy as np

## 📊 Array Creation Methods

### 1. np.array()

* Converts a list or tuple to a NumPy array.
* ✅ Real-time use: Used to convert raw data into a format for processing in ML and data analysis.

np.array([1, 2, 3])  
np.array([[1, 2], [3, 4]])  
np.array((5, 6, 7))

### 2. np.zeros(shape)

* Creates an array of zeros.
* ✅ Real-time use: Initialize weights in machine learning or empty image frames.

np.zeros(3)  
np.zeros((2, 2))  
np.zeros((3, 1))

### 3. np.ones(shape)

* Creates an array of ones.
* ✅ Real-time use: Create baseline predictions or initialize masks in image processing.

np.ones(4)  
np.ones((2, 3))  
np.ones((1, 5))

### 4. np.arange(start, stop, step)

* Creates a range of numbers.
* ✅ Real-time use: Time intervals in simulations or feature creation in ML.

np.arange(5)  
np.arange(1, 10, 2)  
np.arange(10, 0, -2)

### 5. np.linspace(start, stop, num)

* Evenly spaced numbers between two limits.
* ✅ Real-time use: Plotting graphs with equal spacing, or in scientific simulations.

np.linspace(0, 1, 5)  
np.linspace(1, 10, 4)  
np.linspace(5, 6, 3)

## 📈 Math & Stats Methods

### 6. np.sum()

* Adds all elements.
* ✅ Real-time use: Calculate total sales, total users, etc.

np.sum([1, 2, 3])  
np.sum([[1, 2], [3, 4]])  
np.sum(np.array([[1, 2], [3, 4]]), axis=0)

### 7. np.mean()

* Average value of array elements.
* ✅ Real-time use: Find average temperature, salary, grades, etc.

np.mean([10, 20, 30])  
np.mean([[1, 2], [3, 4]])  
np.mean(np.array([[1, 2], [3, 4]]), axis=1)

### 8. np.median()

* Finds the middle value.
* ✅ Real-time use: Used in statistics when data has outliers (e.g., house prices).

np.median([10, 20, 30])  
np.median([10, 20, 30, 40])  
np.median([[5, 10], [15, 20]])

## 🔄 Array Operation Methods

### 9. np.reshape(shape)

* Changes the shape of the array.
* ✅ Real-time use: Reshaping image data (e.g., 784 to 28x28 for MNIST).

np.reshape([1, 2, 3, 4], (2, 2))  
np.reshape(np.arange(6), (3, 2))  
np.reshape(np.arange(12), (2, 2, 3))

### 10. np.flatten()

* Converts multi-dimensional to 1D.
* ✅ Real-time use: Prepare data for ML models that require 1D input.

np.array([[1, 2], [3, 4]]).flatten()  
np.array([[10, 20], [30, 40]]).flatten()  
np.array([[5], [6]]).flatten()

### 11. np.concatenate((a, b), axis)

* Combines arrays.
* ✅ Real-time use: Merge datasets or feature arrays.

a = np.array([1, 2])  
b = np.array([3, 4])  
np.concatenate((a, b))  
  
x = np.array([[1, 2]])  
y = np.array([[3, 4]])  
np.concatenate((x, y), axis=0)  
  
p = np.array([[1], [2]])  
q = np.array([[3], [4]])  
np.concatenate((p, q), axis=1)

## 🔢 Indexing and Logical Methods

### 12. np.where(condition)

* Finds elements based on condition.
* ✅ Real-time use: Filter users based on age, purchases, etc.

a = np.array([10, 20, 30])  
np.where(a > 15)  
np.where(a == 20)  
np.where(a % 10 == 0)

### 13. np.unique()

* Returns unique values.
* ✅ Real-time use: Identify distinct products, categories, tags, etc.

np.unique([1, 2, 2, 3])  
np.unique([[1, 1], [2, 3]])  
np.unique(['a', 'b', 'a'])

### 14. np.clip(arr, min, max)

* Limit values within a range.
* ✅ Real-time use: Normalize image pixel values or limit predictions.

a = np.array([10, 20, 30])  
np.clip(a, 15, 25)  
  
b = np.array([-5, 0, 5, 10])  
np.clip(b, 0, 8)  
  
c = np.array([100, 200, 300])  
np.clip(c, 150, 250)

## 🔢 Linear Algebra Methods

### 15. np.dot(a, b)

* Matrix multiplication.
* ✅ Real-time use: Used in deep learning, physics simulations, or economic models.

np.dot([1, 2], [10, 20])  
a = np.array([[1, 2], [3, 4]])  
b = np.array([[5, 6], [7, 8]])  
np.dot(a, b)  
np.dot([[2, 3]], [[4], [5]])

### 16. np.transpose()

* Swap rows and columns.
* ✅ Real-time use: Re-orient images or matrices for matrix multiplication.

np.transpose([[1, 2], [3, 4]])  
np.transpose([[1, 2, 3]])  
np.transpose([[5], [6]])

### 17. np.linalg.inv()

* Inverse of a matrix.
* ✅ Real-time use: Solve systems of linear equations or transformations in graphics.

from numpy.linalg import inv  
inv([[1, 2], [3, 4]])  
inv([[2, 1], [7, 4]])  
inv([[5, 3], [2, 1]])

## 🌀 Random Number Methods

### 18. np.random.rand()

* Random floats [0, 1).
* ✅ Real-time use: Initialize neural network weights or generate test inputs.

np.random.rand(3)  
np.random.rand(2, 2)  
np.random.rand(1, 4)

### 19. np.random.randint(start, stop, size)

* Random integers in range.
* ✅ Real-time use: Random sample generation, password simulation.

np.random.randint(1, 10)  
np.random.randint(5, 15, size=3)  
np.random.randint(0, 2, size=(2, 2))

### 20. np.random.randn()

* Random numbers from a normal distribution.
* ✅ Real-time use: Simulate stock price changes, error terms in models.

np.random.randn(4)  
np.random.randn(2, 3)  
np.random.randn(1, 2)

## 🤢 Array Properties

### 21. .shape

* Dimensions of array.
* ✅ Real-time use: Understand image size, model input format.

np.array([[1, 2, 3], [4, 5, 6]]).shape  
np.array([1, 2, 3]).shape  
np.array([[7], [8], [9]]).shape

### 22. .ndim

* Number of dimensions.
* ✅ Real-time use: Inspect if input is 1D, 2D, or 3D for models.

np.array([1, 2, 3]).ndim  
np.array([[1, 2], [3, 4]]).ndim  
np.array([[[1], [2]], [[3], [4]]]).ndim

### 23. .size

* Total number of elements.
* ✅ Real-time use: Memory analysis and reshaping validation.

np.array([1, 2, 3]).size  
np.array([[1, 2], [3, 4]]).size  
np.array([[[1, 2], [3, 4]]]).size

## 📃 Real-life Uses

| Use Case | Example |
| --- | --- |
| Data Science | Matrix transformations |
| Machine Learning | Feature scaling, array broadcasting |
| Engineering | Scientific computation, signal analysis |
| Finance | Portfolio risk calculations |
| Game Development | Collision grids, character animation |

## 📆 Summary Table

| Category | Methods | Real-Time Use Example |
| --- | --- | --- |
| Creation | array, zeros, ones, arange, linspace | Setup data inputs, placeholder arrays |
| Math & Stats | sum, mean, median | Statistics, KPIs, data analytics |
| Array Ops | reshape, flatten, concatenate | Model input formatting, dataset merging |
| Indexing/Logic | where, unique, clip | Filtering, de-duplication, normalizing data |
| Linear Algebra | dot, transpose, linalg.inv | Neural networks, simulations, math models |
| Random | random.rand, random.randint, random.randn | Data sampling, simulation, testing |
| Attributes | .shape, .ndim, .size | Data validation, reshaping, debugging |

Let me know if you’d like this exported as a PDF, or continued with more examples and exercises!