**Introduction to NumPy**

NumPy (Numerical Python) is a fundamental library in Python used for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

**Why Do We Use NumPy?**

* Python’s built-in lists are not optimized for numerical operations. NumPy provides a more efficient alternative.
* It allows for fast computations and efficient memory handling.
* Many data science, machine learning, and scientific computing libraries (like Pandas, SciPy, TensorFlow) are built on top of NumPy.

**Advantages of NumPy**

1. **Performance** – Faster than Python lists due to optimized C-based implementations.
2. **Memory Efficiency** – Uses less memory than Python lists.
3. **Vectorization** – Allows element-wise operations without explicit loops.
4. **Built-in Mathematical Functions** – Provides functions for linear algebra, statistics, and complex computations.
5. **Interoperability** – Works well with other Python libraries like Pandas, SciPy, and scikit-learn.

**Disadvantages of NumPy**

1. **Consumes More Memory for Small Data** – Compared to lists, NumPy arrays may take up more memory for small datasets due to fixed data types.
2. **Requires Learning Curve** – New users need to learn NumPy syntax.
3. **Less Flexibility** – NumPy arrays are homogeneous, meaning all elements must be of the same type.

**What Can We Do with NumPy?**

1. **Array Creation & Manipulation** – Create 1D, 2D, and nD arrays and reshape, slice, or modify them.
2. **Mathematical Operations** – Perform element-wise addition, subtraction, multiplication, division, etc.
3. **Statistical Computations** – Calculate mean, median, variance, standard deviation, etc.
4. **Linear Algebra** – Matrix operations, eigenvalues, determinants, dot products, etc.
5. **Random Number Generation** – Generate random numbers for simulations and probabilistic modeling.
6. **Fourier Transform & Signal Processing** – Apply FFT and other mathematical transformations.
7. **Integration with Other Libraries** – Works seamlessly with Pandas, Matplotlib, SciPy, TensorFlow, etc.
8. Here are **more essential NumPy functions** categorized for better understanding:

### ****Comprehensive NumPy Functions Table****

| **Function** | **Description** | **Usage Example** |
| --- | --- | --- |
| **Array Creation & Initialization** |  |  |
| np.array() | Creates an array from lists or tuples. | np.array([1, 2, 3]) |
| np.zeros() | Creates an array filled with zeros. | np.zeros((2,3)) |
| np.ones() | Creates an array filled with ones. | np.ones((3,3)) |
| np.full() | Creates an array filled with a specified value. | np.full((2,2), 7) |
| np.eye() | Creates an identity matrix. | np.eye(3) |
| np.empty() | Creates an uninitialized array with random values. | np.empty((2,3)) |
| **Array Indexing & Slicing** |  |  |
| np.where() | Returns indices where a condition is met. | np.where(arr > 5) |
| np.take() | Extracts elements at specific indices. | np.take(arr, [0,2]) |
| np.put() | Replaces elements at specified indices. | np.put(arr, [0,1], [10,20]) |
| **Array Reshaping & Manipulation** |  |  |
| np.reshape() | Reshapes an array into a different shape. | arr.reshape(2,3) |
| np.expand\_dims() | Expands the dimensions of an array. | np.expand\_dims(arr, axis=0) |
| np.squeeze() | Removes single-dimensional entries from the shape. | np.squeeze(arr) |
| np.ravel() | Returns a flattened 1D array. | np.ravel(arr) |
| **Mathematical Operations** |  |  |
| np.add() | Element-wise addition. | np.add(arr1, arr2) |
| np.subtract() | Element-wise subtraction. | np.subtract(arr1, arr2) |
| np.multiply() | Element-wise multiplication. | np.multiply(arr1, arr2) |
| np.divide() | Element-wise division. | np.divide(arr1, arr2) |
| np.power() | Raises each element to a specified power. | np.power(arr, 2) |
| np.sqrt() | Computes the square root of elements. | np.sqrt(arr) |
| np.cumsum() | Cumulative sum of elements. | np.cumsum(arr) |
| np.cumprod() | Cumulative product of elements. | np.cumprod(arr) |
| **Statistical Functions** |  |  |
| np.median() | Computes the median of elements. | np.median(arr) |
| np.std() | Computes the standard deviation. | np.std(arr) |
| np.var() | Computes the variance of elements. | np.var(arr) |
| np.percentile() | Computes a specific percentile of elements. | np.percentile(arr, 75) |
| np.histogram() | Computes the histogram of an array. | np.histogram(arr, bins=5) |
| **Random Number Generation** |  |  |
| np.random.seed() | Sets a random seed for reproducibility. | np.random.seed(42) |
| np.random.randn() | Generates random numbers from a normal distribution. | np.random.randn(3,3) |
| np.random.choice() | Selects random elements from an array. | np.random.choice(arr, 3) |
| **Linear Algebra** |  |  |
| np.linalg.norm() | Computes the norm (magnitude) of a vector. | np.linalg.norm(arr) |
| np.linalg.solve() | Solves a system of linear equations. | np.linalg.solve(A, B) |
| np.linalg.svd() | Computes Singular Value Decomposition. | np.linalg.svd(matrix) |
| **Logical & Comparison Functions** |  |  |
| np.all() | Returns True if all elements are True. | np.all(arr > 0) |
| np.any() | Returns True if any element is True. | np.any(arr > 0) |
| np.isfinite() | Checks if elements are finite numbers. | np.isfinite(arr) |
| np.isnan() | Checks for NaN (Not a Number) values. | np.isnan(arr) |
| np.isinf() | Checks for infinite values. | np.isinf(arr) |
| np.logical\_and() | Element-wise logical AND operation. | np.logical\_and(arr1, arr2) |
| np.logical\_or() | Element-wise logical OR operation. | np.logical\_or(arr1, arr2) |
| **Fourier Transform & Signal Processing** |  |  |
| np.fft.fft() | Computes the Fast Fourier Transform (FFT). | np.fft.fft(arr) |
| np.fft.ifft() | Computes the inverse FFT. | np.fft.ifft(arr) |
| np.fft.fftshift() | Shifts the zero-frequency component to the center. | np.fft.fftshift(arr) |