Introduction to NumPy

NumPy, short for Numerical Python, is a powerful library in Python that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. This document outlines the fundamental concepts of NumPy, including array creation, operations, and useful functions, making it an essential tool for data science and machine learning applications.

1. Creating NumPy Arrays

To create a NumPy array, you first need to import the library:

```
import numpy as np

arr = np.array([10, 20, 30, 40])
print(arr)
print(type(arr))
```

👉 NumPy arrays are faster than Python lists, making them ideal for numerical computations.

2. 2D Arrays

NumPy also supports multi-dimensional arrays. For example, you can create a 2D array (matrix) as follows:

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

👉 2D arrays are commonly used in images, machine learning, and matrix operations.

4. Array Operations (Power of NumPy)

One of the key features of NumPy is its ability to perform operations on arrays without the need for explicit loops. Here's how you can perform basic operations:

```
arr = np.array([10, 20, 30])

print(arr + 5)  # Addition
print(arr * 2)  # Multiplication
```

👉 These are vectorized operations, which means they are executed at once, leading to faster computations.

5. Useful Functions

NumPy provides a variety of built-in functions to perform statistical operations on arrays. Here are some examples:

```
numbers = np.array([5, 10, 15, 20])

print("Mean:", np.mean(numbers))
print("Sum:", np.sum(numbers))
print("Max:", np.max(numbers))
print("Min:", np.min(numbers))
```

These functions help in quickly analyzing data without writing extensive code.

6. Indexing & Slicing

Indexing and slicing in NumPy arrays are straightforward and similar to Python lists. Here's how you can access elements:

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

print(arr[0])    # Accessing the first element
print(arr[1:4])  # Slicing from index 1 to 3
```

This feature allows for efficient data manipulation and retrieval.

7. Special Arrays

NumPy also provides functions to create special types of arrays:

```
print(np.zeros(3))  # Array of zeros
print(np.ones(4))   # Array of ones
print(np.arange(1, 10))  # Array with a range of values
```

These functions are useful for initializing arrays for various applications.

🎯 Mini Project: Marks Analysis System

Let's analyze student marks using NumPy. This mini project demonstrates how to utilize NumPy for basic data analysis:

```
marks = np.array([78, 85, 90, 66, 88])
print("Average Marks:", np.mean(marks))
print("Highest Marks:", np.max(marks))
print("Passed Students:", marks[marks > 70])
```

This simple analysis can help educators understand student performance effectively.

In conclusion, NumPy is an essential library for anyone working with numerical data in Python. Its efficiency and ease of use make it a cornerstone for data analysis, scientific computing, and machine learning tasks. By mastering NumPy, you can significantly enhance your data manipulation and analysis capabilities.