Day 15:

**Introduction to NumPy**

**NumPy** (short for Numerical Python) is a powerful library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays. Here are some key points about NumPy:

1. **Arrays**: The core data structure in NumPy is the **array**. An array is a grid of values, all of the same type, indexed by a tuple of non-negative integers. You can think of it as a more versatile version of Python lists.
2. **Vectorization**: NumPy allows you to perform element-wise operations on entire arrays without writing explicit loops. This vectorized approach leads to efficient and concise code.
3. **Broadcasting**: Broadcasting is a powerful feature that allows NumPy to handle operations between arrays of different shapes. It automatically expands smaller arrays to match the shape of larger arrays.
4. **Mathematical Functions**: NumPy provides a plethora of mathematical functions (trigonometry, logarithms, exponentials, etc.) that work seamlessly with arrays.

**Basic Array Operations**

Let’s start with some basic array operations:

1. **Creating Arrays**:
   * You can create arrays using the numpy.array() function or other specialized functions like numpy.zeros(), numpy.ones(), and numpy.arange().
   * Example:

**Python**

import numpy as np

my\_array = np.array([1, 2, 3])

1. **Array Attributes**:
   * You can access various attributes of an array, such as its shape, data type, and number of dimensions.
   * Example:

**Python**

print(my\_array.shape) # Prints (3,)

print(my\_array.dtype) # Prints int64

1. **Element-wise Operations**:
   * You can perform arithmetic operations on arrays element-wise.
   * Example:

**Python**

another\_array = np.array([4, 5, 6])

result = my\_array + another\_array

1. **Indexing and Slicing**:
   * You can access individual elements of an array using indexing (similar to lists).
   * Example:

**Python**

print(my\_array[0]) # Access the first element

print(my\_array[1:]) # Slice from the second element onwards

**Techniques for Indexing, Slicing, and Reshaping Arrays**

1. **Indexing**:
   * Use square brackets to access specific elements of an array.
   * Remember that indexing is zero-based (the first element is at index 0).
2. **Slicing**:
   * Slicing allows you to extract a portion of an array.
   * Syntax: array[start:stop:step]
   * Example:

**Python**

sub\_array = my\_array[1:3] # Extract elements at index 1 and 2

1. **Reshaping**:
   * You can change the shape of an array using the reshape() method.
   * Example:

**Python**

reshaped\_array = my\_array.reshape(3, 1) # Convert to a column vector