Day 16&17

**Broadcasting and Vectorization in NumPy**

**Broadcasting**

* **Broadcasting** is a powerful feature in NumPy that allows you to perform element-wise operations on arrays with different shapes.
* When you perform an operation between two arrays, NumPy automatically broadcasts the smaller array to match the shape of the larger array.
* Broadcasting rules:
  + If the arrays have different dimensions, pad the smaller array with ones on the left until their dimensions match.
  + If the sizes along a dimension don’t match, the smaller array is repeated along that dimension.
* Example:

**Python**

import numpy as np

a = np.array([1, 2, 3])

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

result = a + b # Broadcasting: a is broadcasted to match b's shape

**Vectorization**

* **Vectorization** is the practice of performing operations on entire arrays (or chunks of arrays) without explicit loops.
* It leverages low-level optimizations and hardware acceleration for efficiency.
* Vectorized code is concise, readable, and often faster than equivalent loop-based code.
* Example:

**Python**

# Non-vectorized (loop-based) approach

def multiply\_elements(arr1, arr2):

result = np.empty\_like(arr1)

for i in range(len(arr1)):

result[i] = arr1[i] \* arr2[i]

return result

# Vectorized approach

def multiply\_elements\_vectorized(arr1, arr2):

return arr1 \* arr2

**Advanced Multidimensional Array Operations**

1. **Matrix Operations**:
   * NumPy provides efficient matrix multiplication (np.dot() or @ operator) and element-wise matrix operations.
   * Example:

**Python**

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

B = np.array([[5, 6], [7, 8]])

C = A @ B # Matrix multiplication

1. **Aggregation Functions**:
   * NumPy offers functions like np.sum(), np.mean(), np.max(), etc., to aggregate values along specific axes.
   * Example:

**Python**

data = np.random.rand(3, 4)

total\_sum = np.sum(data) # Sum of all elements

row\_sums = np.sum(data, axis=1) # Sum along rows

1. **Array Manipulation**:
   * Use functions like np.reshape(), np.transpose(), and np.concatenate() to manipulate arrays.
   * Example:

**Python**

arr = np.arange(12).reshape(3, 4)

transposed\_arr = np.transpose(arr) # Transpose the array