**WEEK-5**

**NumPy Basics and Operations**

**DESCRIPTION**

* This unit provides a comprehensive introduction to **NumPy**, one of the most essential libraries in Python for numerical computing. NumPy offers efficient array operations and forms the foundation for libraries like Pandas, SciPy, and scikit-learn.
* Students will explore the creation of **NumPy arrays**, understand their attributes such as **shape, size, and dimension**, and learn the importance of **vectorization** and broadcasting over traditional looping methods. The unit also covers **array indexing, slicing, reshaping, and flattening**.
* Advanced mathematical operations like **dot product, matrix multiplication, transpose, and inversion** are explored, helping students grasp how NumPy simplifies linear algebra tasks. By the end of this unit, students will be proficient in performing a wide variety of data manipulation tasks using NumPy arrays efficiently and concisely.

**Additional Programs**

**1.Write a Python program to demonstrate array creation, indexing, slicing, reshaping, and flattening using NumPy.**

**Description:**

This program demonstrates basic array manipulations using the NumPy library. It covers:

* Creating 1D and 2D arrays
* Indexing and slicing operations
* Reshaping arrays to different dimensions
* Flattening a multi-dimensional array into a 1D array

These operations form the foundation for more advanced data processing and numerical tasks using NumPy.

**Program:**

import numpy as np

# Creating a 1D array

arr1 = np.array([10, 20, 30, 40, 50])

print("1D Array:", arr1)

# Indexing and slicing

print("Element at index 2:", arr1[2])

print("Sliced array (1:4):", arr1[1:4])

# Creating a 2D array

arr2 = np.array([[1, 2, 3], [4, 5, 6]])

print("\n2D Array:\n", arr2)

# Indexing and slicing 2D

print("Element at (1,2):", arr2[1, 2])

print("First row:", arr2[0, :])

print("Second column:", arr2[:, 1])

# Reshaping

reshaped = np.reshape(arr1, (5, 1))

print("\nReshaped Array (5x1):\n", reshaped)

# Flattening

flat = arr2.flatten()

print("Flattened 2D array:", flat)

**Output :**

****

**2.Write a program to perform vectorized operations on NumPy arrays (addition, multiplication, exponentiation) and compare it with traditional loops.**

 **Description:**

This experiment demonstrates how NumPy leverages vectorization to perform operations efficiently on entire arrays, compared to the slower approach using Python loops. This is crucial in data science and numerical computing where performance matters.

**Program :**

import numpy as np

import time

# Using loops

a = list(range(1, 100001))

b = list(range(1, 100001))

start = time.time()

c = [x + y for x, y in zip(a, b)]

print("Time using loops:", time.time() - start)

# Using NumPy

a\_np = np.array(a)

b\_np = np.array(b)

start = time.time()

c\_np = a\_np + b\_np

print("Time using NumPy vectorization:", time.time() - start)

**Output:**

****

**3.Write a program to perform matrix operations: addition, multiplication,transpose, and inverse using NumPy.**

 **Description:**

This experiment covers basic linear algebra operations using NumPy. It’s important for applications in ML, image processing, and engineering simulations. Students learn how to perform matrix operations easily and efficiently.

**Program:**

import numpy as np

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

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

# Addition

print("Addition:\n", A + B)

# Multiplication (element-wise)

print("Element-wise Multiplication:\n", A \* B)

# Matrix Multiplication

print("Matrix Multiplication:\n", np.dot(A, B))

# Transpose

print("Transpose of A:\n", A.T)

# Inverse

inv\_A = np.linalg.inv(A)

print("Inverse of A:\n", inv\_A)

**Output:**

 ****

**4.Write a program to demonstrate broadcasting in NumPy with scalar and array operations.**

**Description:**

This experiment explores NumPy’s powerful broadcasting feature, which allows operations on arrays of different shapes without explicit looping. Broadcasting is a key concept for writing efficient numerical code in Python.

**Program:**

import numpy as np

# Scalar broadcasting

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

print("Original array:", arr)

print("After adding 10 (scalar):", arr + 10)

# 2D and 1D array broadcasting

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

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

print("\n2D Array:\n", A)

print("1D Array:\n", B)

print("Broadcasted Addition:\n", A + B)

**Output:**

 ****

**Viva Questions with Answers (Unit–2: NumPy)**

**1. What is the difference between reshape() and flatten() in NumPy?**

reshape() changes the shape of an array without changing its data, while flatten() converts a multi-dimensional array into a 1D array (it returns a copy).

**2. How do you perform matrix multiplication in NumPy?**

Matrix multiplication is performed using np.dot() or the @ operator. For example: np.dot(A, B) multiplies matrices A and B.

**3.What function is used to find the inverse of a matrix in NumPy?**

The function np.linalg.inv() is used to compute the inverse of a square matrix.

**4. Can NumPy arrays store elements of different data types?**

No, NumPy arrays are homogeneous, meaning all elements must be of the same data type, unlike Python lists which can hold heterogeneous types.