Experiment No. 9
Program to manipulate arrays using NumPy
Date of Performance: 25/03/2024
Date of Submission: 01/04/2024

Experiment No. 9

Title: Program to manipulate arrays using NumPy

Aim: To study and implement arrays manipulation using NumPy

Objective: To introduce NumPy package

Theory:

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

Arrays in Numpy

Array in Numpy is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In Numpy, number of dimensions of the array is called rank of the array.A tuple of integers giving the size of the array along each dimension is known as shape of the array. An array class in Numpy is called as **ndarray**. Elements in Numpy arrays are accessed by using square brackets and can be initialized by using nested Python Lists.

Creating a Numpy Array

Arrays in Numpy can be created by multiple ways, with various number of Ranks, defining the size of the Array. Arrays can also be created with the use of various data types such as lists, tuples, etc. The type of the resultant array is deduced from the type of the elements in the sequences.

Note: Type of array can be explicitly defined while creating the array.

Program:

```
import numpy as np


# Define a function to demonstrate array manipulation using NumPy

def numpy_array_manipulation():
```

```
# Creating arrays using different methods
array1 = np.array([1, 2, 3, 4, 5])  # Creating array from a list
array2 = np.zeros((3, 3))  # Creating a 3x3 array filled with zeros
array3 = np.ones((2, 4))  # Creating a 2x4 array filled with ones
array4 = np.random.randint(0, 10, size=(2, 3))  # Creating a 2x3 array with random integers between 0 and 10

# Displaying the created arrays
print("Array 1:")
print(array1)
print("\nArray 2:")
print(array2)
print("\nArray 3:")
print(array3)
print("\nArray 4:")
print(array4)

# Array manipulation
array5 = np.arange(10)  # Creating an array with numbers from 0 to 9
array6 = array5.reshape(2, 5)  # Reshaping the array to a 2x5 array
array7 = array6.transpose()  # Transposing the array
array8 = np.flip(array5)  # Flipping the array

# Displaying the manipulated arrays
```

```
    print("\nArray 5:")
    print(array5)
    print("\nArray 6 (Reshaped):")
    print(array6)
    print("\nArray 7 (Transposed):")
    print(array7)
    print("\nArray 8 (Flipped):")
    print(array8)


# Define a function to search for an element in an array
def search_element(array, element):
    if element in array:
        print(f"Element {element} found in the array.")
    else:
        print(f"Element {element} not found in the array.")


# Call the function to demonstrate array manipulation
numpy_array_manipulation()


# Call the function to search for an element in an array
search_element(np.array([1, 2, 3, 4, 5]), 3)
```

Output:

Array 1:

[1 2 3 4 5]

Array 2:

[[0. 0. 0.]

[0. 0. 0.]

[0. 0. 0.]]

Array 3:

[[1. 1. 1. 1.]

[1. 1. 1. 1.]]

Array 4:

[[7 2 0]

[3 1 4]]

Array 5:

[0 1 2 3 4 5 6 7 8 9]

Array 6 (Reshaped):

[[0 1 2 3 4]

[5 6 7 8 9]]

Array 7 (Transposed):

[[0 5]

[1 6]

[2 7]

[3 8]

[4 9]]

Array 8 (Flipped):

[9 8 7 6 5 4 3 2 1 0]

Element 3 found in the array.

Conclusion:

After performing Experiment No. 9, it's evident that NumPy offers powerful tools for array manipulation in Python, facilitating efficient scientific computing. Through this experiment, arrays were created, reshaped, transposed, and flipped, showcasing NumPy's versatility. Additionally, the capability to search for elements within arrays was demonstrated, enhancing the practical utility of NumPy in data analysis and scientific research. Overall, this experiment provided a comprehensive introduction to NumPy and its essential functionalities for array processing in Python.