NAME: **Meet Raut**

DIV: **S2-1**

ROLL NO: **2201084**

EXPERIMENT – 9:

- ***AIM:*** **To study and implement program to demonstrate use of NumPy: Array objects.**

- ***THEORY:***

Python Numpy:

NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements.
Travis Oliphant created NumPy package in 2005 by injecting the features of the ancestor module Numeric into another module Numarray.

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.

Accessing the array Index: In a numpy array, indexing or accessing the array index can be done in multiple ways. To print a range of an array, slicing is done. Slicing of an array is defining a range in a new array which is used to print a range of elements from the original array. Since, sliced array holds a range of elements of the original array, modifying content with the help of sliced array modifies the original array content.

Basic Array Operations: In numpy, arrays allow a wide range of operations which can be performed on a particular array or a combination of Arrays. These operation include some basic Mathematical operation as well as Unary and Binary operations.

Data Types in Numpy:

Every Numpy array is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. Every ndarray has an associated data type (dtype) object. This data type object (dtype) provides information about the layout of the array. The values of an ndarray are stored in a buffer which can be thought of as a contiguous block of memory bytes which can be interpreted by the dtype object. Numpy provides a large set of numeric datatypes that can be used to construct arrays. At the time of Array creation, Numpy tries to guess a datatype, but functions that construct arrays usually also include an optional argument to explicitly specify the datatype.

Constructing a Datatype Object: In Numpy, datatypes of Arrays need not to be defined unless a specific datatype is required. Numpy tries to guess the datatype for Arrays which are not predefined in the constructor function.

Math Operations on DataType array: In Numpy arrays, basic mathematical operations are performed element-wise on the array. These operations are applied both as operator overloads and as functions. Many useful functions are provided in Numpy for performing computations on Arrays such as **sum**: for addition of Array elements, **T**: for Transpose of elements, etc.

The need of NumPy:

With the revolution of data science, data analysis libraries like NumPy, SciPy, Pandas, etc. have seen a lot of growth. With a much easier syntax than other programming languages, python is the first choice language for the data scientist.

NumPy provides a convenient and efficient way to handle the vast amount of data. NumPy is also very convenient with Matrix multiplication and data reshaping. NumPy is fast which makes it reasonable to work with a large set of data.

There are the following advantages of using NumPy for data analysis.

1. NumPy performs array-oriented computing.
2. It efficiently implements the multidimensional arrays.
3. It performs scientific computations.
4. It is capable of performing Fourier Transform and reshaping the data stored in multidimensional arrays.
5. NumPy provides the in-built functions for linear algebra and random number generation.

Nowadays, NumPy in combination with SciPy and Mat-plotlib is used as the replacement to MATLAB as Python is more complete and easier programming language than MATLAB.

NumPy Ndarray:

Ndarray is the n-dimensional array object defined in the numpy which stores the collection of the similar type of elements. In other words, we can define a ndarray as the collection of the data type (dtype) objects.

The ndarray object can be accessed by using the 0 based indexing. Each element of the Array object contains the same size in the memory.

Creating a ndarray object

The ndarray object can be created by using the array routine of the numpy module. For this purpose, we need to import the numpy.

The parameters are described in the following table.

SN	Parameter	Description
1	object	It represents the collection object. It can be a list, tuple, dictionary, set, etc.
2	dtype	We can change the data type of the array elements by changing this option to the specified type. The default is none.
3	copy	It is optional. By default, it is true which means the object is copied.
4	order	There can be 3 possible values assigned to this option. It can be C (column order), R (row order), or A (any)
5	subok	The returned array will be base class array by default. We can change this to make the subclasses passes through by setting this option to true.
6	ndmin	It represents the minimum dimensions of the resultant array.

Finding the dimensions of the Array

The ndim function can be used to find the dimensions of the array.

Finding the size of each array element

The itemsize function is used to get the size of each array item. It returns the number of bytes taken by each array element.

Finding the data type of each array item

To check the data type of each array item, the dtype function is used. Consider the following example to check the data type of the array items.

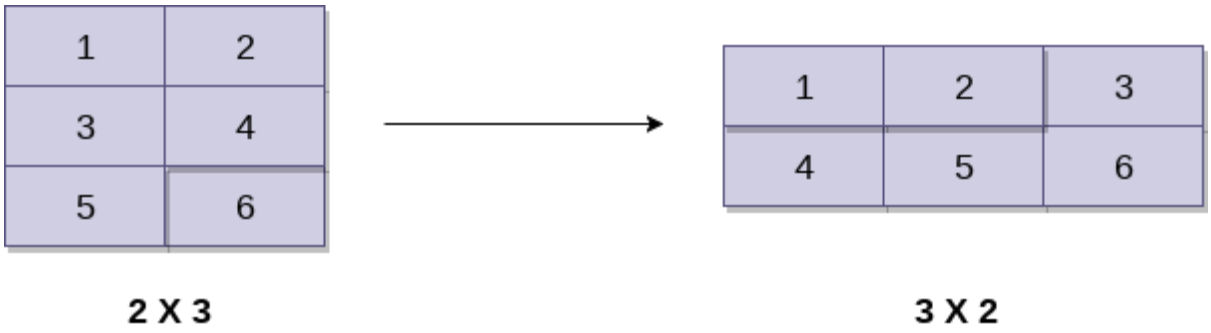
Finding the shape and size of the array

To get the shape and size of the array, the size and shape function associated with the numpy array is used.

Reshaping the array objects

By the shape of the array, we mean the number of rows and columns of a multi-dimensional array. However, the numpy module provides us the way to reshape the array by changing the number of rows and columns of the multi-dimensional array.

The reshape() function associated with the ndarray object is used to reshape the array. It accepts the two parameters indicating the row and columns of the new shape of the array.



Slicing in the Array

Slicing in the NumPy array is the way to extract a range of elements from an array. Slicing in the array is performed in the same way as it is performed in the python list.

Linspace

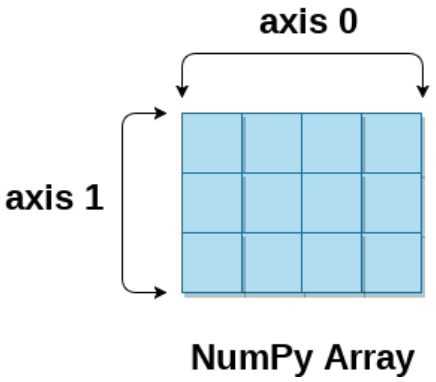
The linspace() function returns the evenly spaced values over the given interval.

Finding the maximum, minimum, and sum of the array elements

The NumPy provides the max(), min(), and sum() functions which are used to find the maximum, minimum, and sum of the array elements respectively.

NumPy Array Axis

A NumPy multi-dimensional array is represented by the axis where axis-0 represents the columns and axis-1 represents the rows. We can mention the axis to perform row-level or column-level calculations like the addition of row or column elements.



- **PROGRAM:**

```
import numpy as np

a = np.array([1, 22, 3, 10, 15, 4, 55, 23, 44, 70])
b = np.zeros((2, 3, 4, 5))
d = np.full((3, 3, 3), 2)

print("1D array:", a)

print("The maximum element:", a.max())
print("The minimum element:", a.min())
print("The Length of array is:", len(a))
print("The sum of the elements:", a.sum())

print("Numbers after the 2nd index:", a[2:])
print("Last 2 numbers:", a[-2:])
```

```
print("-----------------------------------------------")



print("\n3D array with all 2s:")
print(d)

print("-----------------------------------------------")

print("4D array with all 0s:")
print(b)

print("-----------------------------------------------")

A = np.array([[11, 12, 13], [14, 15, 16], [17, 18, 19]])
C = np.array([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

D = np.add(A, C)


print("Matrix A:")
print(A)
print("\nMatrix C:")
print(C)

print("\nAddition of matrix A & C:")
print(D)

print("-----------------------------------------------")

x = np.random.randint(100)
print("Random Number:", x)

print("-----------------------------------------------")

matrices = [np.random.randint(1, 4, size=(3, 3)) for _ in range(4)]
```

```
for i, matrix in enumerate(matrices):
    print(f"\nMatrix {i+1}:")
    print("Random matrix with no b/w 1 & 3:")
    print(matrix)

print("-----------------------------------------------")

array1 = np.arange(0, 15, 3)
print("ARANGE ARRAY:")
print(array1)

print("-----------------------------------------------")

array2 = np.linspace(0, 40, 5)
print("LINESPACE FUNCTION:")
print(array2)

print("-----------------------------------------------")

a1 = np.random.randint(1, 10, size=(3, 3))
b1 = np.random.randint(1, 10, size=(3, 3))
c1 = np.random.randint(1, 10, size=(3, 3))

print("Array a1:")
print(a1)
print("\nArray b1:")
print(b1)
print("\nArray c1:")
print(c1)

min_a1 = np.min(a1)
max_a1 = np.max(a1)
std_a1 = np.std(a1)
avg_a1 = np.average(a1)
var_a1 = np.var(a1)
med_a1 = np.median(a1)
mean_a1 = np.mean(a1)
```

```
min_b1 = np.min(b1)
max_b1 = np.max(b1)
std_b1 = np.std(b1)
avg_b1 = np.average(b1)
var_b1 = np.var(b1)
med_b1 = np.median(b1)
mean_b1 = np.mean(b1)

min_c1 = np.min(c1)
max_c1 = np.max(c1)
std_c1 = np.std(c1)
avg_c1 = np.average(c1)
var_c1 = np.var(c1)
med_c1 = np.median(c1)
mean_c1 = np.mean(c1)

print("\n---FOR MATRIX a1---")
print("Min of a1:", min_a1)
print("Max of a1:", max_a1)
print("Standard deviation of a1:", std_a1)
print("Average of a1:", avg_a1)
print("Variance of a1:", var_a1)
print("Median of a1:", med_a1)
print("Mean of a1:", mean_a1)

print("\n---FOR MATRIX b1---")
print("Min of b1:", min_b1)
print("Max of b1:", max_b1)
print("Standard deviation of b1:", std_b1)
print("Average of b1:", avg_b1)
print("Variance of b1:", var_b1)
print("Median of b1:", med_b1)
print("Mean of b1:", mean_b1)

print("\n---FOR MATRIX c1---")
print("Min of c1:", min_c1)
```

```
print("Max of c1:", max_c1)
print("Standard deviation of c1:", std_c1)
print("Average of c1:", avg_c1)
print("Variance of c1:", var_c1)
print("Median of c1:", med_c1)
print("Mean of c1:", mean_c1)
```

- **OUTPUT:**

```
1D array: [ 1 22  3 10 15  4 55 23 44 70]
The maximum element: 70
The minimum element: 1
The Length of array is: 10
The sum of the elements: 247
Numbers after the 2nd index: [ 3 10 15  4 55 23 44 70]
Last 2 numbers: [44 70]
-------------------------------------------------

3D array with all 2s:
[[[2 2 2]
  [2 2 2]
  [2 2 2]]

 [[2 2 2]
  [2 2 2]
  [2 2 2]]

 [[2 2 2]
  [2 2 2]
  [2 2 2]]]
-------------------------------------------------
```

```
------------------------------------------------
4D array with all 0s:
[[[[0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]]

  [[0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]]

  [[0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]]]


 [[[0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]]

  [[0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]]

  [[0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]
   [0. 0. 0. 0. 0.]]]]
```

```
------------------------------------------------
Matrix A:
[[11 12 13]
 [14 15 16]
 [17 18 19]]

Matrix C:
[[9 8 7]
 [6 5 4]
 [3 2 1]]

Addition of matrix A & C:
[[20 20 20]
 [20 20 20]
 [20 20 20]]
------------------------------------------------
Random Number: 9
------------------------------------------------

Matrix 1:
Random matrix with no b/w 1 & 3:
[[2 2 1]
 [2 2 1]
 [3 3 3]]

Matrix 2:
Random matrix with no b/w 1 & 3:
[[1 1 1]
 [1 2 2]
 [1 2 3]]
```

```
Matrix 3:
Random matrix with no b/w 1 & 3:
[[2 1 2]
 [3 2 3]
 [3 1 1]]

Matrix 4:
Random matrix with no b/w 1 & 3:
[[1 1 2]
 [1 1 2]
 [3 1 1]]
--------------------------------------------------
ARANGE ARRAY:
[ 0  3  6  9 12]
--------------------------------------------------
LINESPACE FUNCTION:
[ 0. 10. 20. 30. 40.]
--------------------------------------------------
Array a1:
[[9 7 7]
 [9 9 5]
 [8 8 2]]

Array b1:
[[3 6 3]
 [2 5 5]
 [8 5 6]]
```

```
Array c1:
[[9 6 1]
 [6 6 1]
 [8 1 8]]

---FOR MATRIX a1---
Min of a1: 2
Max of a1: 9
Standard deviation of a1: 2.1829869671542776
Average of a1: 7.111111111111111
Variance of a1: 4.765432098765432
Median of a1: 8.0
Mean of a1: 7.111111111111111

---FOR MATRIX b1---
Min of b1: 2
Max of b1: 8
Standard deviation of b1: 1.7497795275581802
Average of b1: 4.777777777777778
Variance of b1: 3.0617283950617282
Median of b1: 5.0
Mean of b1: 4.777777777777778

---FOR MATRIX c1---
Min of c1: 1
Max of c1: 9
Standard deviation of c1: 3.0711722135745005
Average of c1: 5.111111111111111
Variance of c1: 9.432098765432098
Median of c1: 6.0
Mean of c1: 5.111111111111111
```

- ***CONCLUSION:*** **Hence, we have successfully implemented program to demonstrate use of NumPy: Array objects; LO 1.**