THADOMAL SHAHANI ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY

Roll no:I-62

5. NumPy: LO6

1)Aim:

Write a Python program to create a 1D, 2D, and 3D NumPy array.Perform basic operations like reshaping, slicing, and indexing.

Theory:

- NumPy supports arrays in one, two, and three dimensions.
- Data manipulation is enabled through operations such as reshaping, slicing, and indexing.
- Memory-efficient handling and vectorized computations enhance overall performance.
- Broadcasting simplifies working with arrays of varying shapes.
- Functions like np.zeros(), np.ones(), and np.arange() allow for quick and efficient array creation.

Program:

```
import numpy as np
my_list = [1, 23, 3, 4]
arr = np.array(my_list)
print(arr)
arr1 = np.array([[1, 2, 3], [4, 5, 6], [5, 6, 7]])
print(arr1)
print(arr1[0, 2])
arr2 = np.reshape(arr1, (3, 3))
print(arr2)
print(arr1[:1, :2])
```

Output:

```
[ 1 2 3 4]
[[1 2 3]
 [4 5 6]
 [5 6 7]]
```

```
[[1 2 3]
 [4 5 6]
 [5 6 7]]
[[1 2]]
```

Conclusion:

NumPy's flexibility in handling arrays enables streamlined and effective numerical operations.

2)Aim:

Develop a Python script to create two arrays of the same shape and perform element-wise addition, subtraction, multiplication, and division.Calculate the dot product and cross product of two vectors.

Theory:

- Arrays with identical shapes allow for element-wise arithmetic operations, such as addition, subtraction, multiplication, and division.
- NumPy’s broadcasting feature efficiently manages arrays of varying dimensions.
- Element-wise computations are designed for high-speed execution and optimal memory usage**.**

Program:

```
import numpy as np
l1 = [1, 2, 3]
l2 = [4, 5, 6]
a1 = np.array(l1)
a2 = np.array(l2)
a = a1 + a2
print(a)
s = a1 - a2
print(s)
d = np.dot(a1, a2)
print(d)
c = np.cross(a1, a2)
print(c)
```

```
a3 = np.reshape(a2, (1, 3))
print(a3)
```

Output:

```
[5 7 9]
[-3 -3 -3]
32
[-3  6 -3]
[[4 5 6]]
```

Conclusion:

NumPy's element-wise operations enable fast and effective mathematical calculations on arrays.

3)Aim:

Write a Python program to calculate mean, median,standard deviation, variance, and correlation coefficients of a given array.

Theory:

- NumPy offers a range of functions for statistical operations, including mean, median, standard deviation, variance, and correlation.
- These tools are crucial for performing statistical analysis in data science and machine learning.
- It ensures efficient processing of large datasets and simplifies complex calculations.
- The np.corrcoef() function specifically computes correlation coefficients with ease.

Program:

```
import numpy as np

# Create two NumPy arrays
a = np.array([1, 2, 3])
a2 = np.array([4, 5, 6])

mean = np.mean(a)
```

```
coeff = np.corrcoef(a, a2)
print("Mean of array a:", mean)
print("Correlation coefficient matrix:\n", coeff)
```

Output

2.0

[[1. 1.]

[1. 1.]]

Conclusion:

NumPy's statistical functions provide robust tools for analyzing data and extracting meaningful insights.