AY: 2024-25

Class:	**SE**	**Semester:**	**IV**
Course Code:	**CSL405**	**Course Name:**	**Skills Based Python Programming Lab**

Name of Student:	**Shravani Sandeep Raut**
Roll No. :	**48**
Experiment No.:	**9**
Title of the Experiment:	**Write a program to demonstrate different NumPy array creation techniques and different NumPy methods.**
Date of Performance:	**11/03/2025**
Date of Submission:	**18/03/2025**

Evaluation

Performance Indicator	**Max. Marks**	**Marks Obtained**
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	**Exceed Expectations (EE)**	**Meet Expectations (ME)**	**Below Expectations (BE)**
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Mr. Raunak Joshi

Signature :

Date:

Aim: Write a program to demonstrate different NumPy array creation techniques and different NumPy methods.

Theory:

NumPy (Numerical Python) is a powerful library in Python used for numerical computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays efficiently. NumPy is widely used in data analysis, scientific computing, and machine learning due to its fast performance and ease of use.

Introduction to NumPy:

- **NumPy Array:** A grid of values of the same data type, indexed by a tuple of non-negative integers. It is also known as an **ndarray** (N-dimensional array).
- **Why Use NumPy?**
  - Faster than Python lists due to optimized C implementation.
  - Requires less memory as arrays are of a fixed data type.
  - Provides various mathematical and logical operations on arrays.
  - Facilitates complex operations like broadcasting, vectorization, and linear algebra calculations.

NumPy Array Creation Techniques:

NumPy provides several ways to create arrays, including:

- Using lists or tuples
- Using built-in functions like arange(), linspace(), zeros(), ones(), and eye()
- Using random number generators

Using arange()

Creates arrays with regularly spaced values.

Syntax: np.arange(start, stop, step, dtype)

Using linspace()

Generates arrays with evenly spaced numbers over a specified range.

Syntax: np.linspace(start, stop, num, endpoint=True)

Using zeros() and ones()

Creates arrays filled with zeros or ones.

Syntax:

- np.zeros(shape, dtype) - Creates an array of zeros.
- np.ones(shape, dtype) - Creates an array of ones.

Using eye()

Creates an identity matrix (square array with ones on the diagonal and zeros elsewhere).

Syntax: np.eye(N, M=None, k=0, dtype)

Using Random Number Generators:

NumPy provides a module np.random to generate arrays with random values.

a. rand() and randn()

- rand() generates random numbers between 0 and 1 (uniform distribution).
- randn() generates numbers from a standard normal distribution (mean 0, variance 1).

randint()

Generates random integers within a specified range.

Syntax: np.random.randint(low, high=None, size, dtype)

Implementation:

```
pip install numpy
```

```
import numpy as np
```

In [3]:

```
# 1. Array Creation Techniques
print("1. Array Creation Techniques")
```

```
1. Array Creation Techniques
```

In [4]:

```
# a. Creating an array from a list
array_from_list = np.array([1, 2, 3, 4, 5])
array_from_list
```

Out[4]:

```
array([1, 2, 3, 4, 5])
```

In [5]:

```
# b. Using arange()
array_arange = np.arange(0, 10, 2)
array_arange
```

Out[5]:

```
array([0, 2, 4, 6, 8])
```

In [6]:

```
# c. Using linspace()
array_linspace = np.linspace(0, 10, 5)  # Divides 0 to 10 into 5 points
array_linspace
```

Out[6]:

```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

In [7]:

```
# d. Using zeros()
array_zeros = np.zeros((3, 3))
array_zeros
```

Out[7]:

```
array([[0., 0., 0.],
       [0., 0., 0.],
      [0., 0., 0.]])
```

In [8]:

```
# e. Using ones()
array_ones = np.ones((2, 2))
array_ones
```

Out[8]:

```
array([[1., 1.],
      [1., 1.]])
```

In [9]:

```
# f. Using eye() for identity matrix
array_eye = np.eye(3)
array_eye
```

Out[9]:

```
array([[1., 0., 0.],
       [0., 1., 0.],
      [0., 0., 1.]])
```

In [10]:

```
# g. Using random() for random values
array_random = np.random.random((3, 3))
array_random
```

Out[10]:

```
array([[0.76371382, 0.71325488, 0.09819297],
       [0.71422995, 0.99949683, 0.86859826],
      [0.27954435, 0.63003605, 0.85325615]])
```

In [11]:

```
# 2. Different NumPy Methods
print("\n2. NumPy Methods")
```

```
2. NumPy Methods
```

In [12]:

```
# a. Reshaping an array
```

```
reshaped_array = np.arange(1, 10).reshape(3, 3)
reshaped_array
```

Out[12]:

```
array([[1, 2, 3],
       [4, 5, 6],
      [7, 8, 9]])
```

In [13]:

```
# b. Transposing an array
transposed_array = reshaped_array.T
transposed_array
```

Out[13]:

```
array([[1, 4, 7],
       [2, 5, 8],
      [3, 6, 9]])
```

In [14]:

```
# c. Mathematical operations
array_math = np.array([1, 2, 3])
array_math + 2
array_math * 3
np.sqrt(array_math)
```

Out[14]:

```
array([1.        , 1.41421356, 1.73205081])
```

In [15]:

```
# d. Aggregation methods
np.sum(array_math)
np.mean(array_math)
np.max(array_math)
np.min(array_math)
```

Out[15]:

```
np.int64(1)
```

In [16]:

```
# e. Concatenation of arrays
array_a = np.array([1, 2, 3])
```

```
array_b = np.array([4, 5, 6])
concat_array = np.concatenate((array_a, array_b))
concat_array
```

Out[16]:

```
array([1, 2, 3, 4, 5, 6])
```

In [17]:

```
# f. Sorting an array
unsorted_array = np.array([3, 1, 4, 2])
sorted_array = np.sort(unsorted_array)
sorted_array
```

Out[17]:

```
array([1, 2, 3, 4])
```

In [18]:

```
# g. Indexing and Slicing
indexed_value = array_math[1]  # Indexing
indexed_value
sliced_array = array_math[1:3]  # Slicing
sliced_array
```

Out[18]:

```
array([2, 3])
```

In [19]:

```
# h. Boolean Masking
boolean_mask = array_math > 2
boolean_mask
array_math[boolean_mask]
```

Out[19]:

```
array([3])
```

Conclusion : NumPy provides a rich set of functionalities to create, manipulate, and perform mathematical operations on arrays. Its high performance, flexibility, and ease of use make it an indispensable tool for data analysis and scientific computing.