**Experiment No: 9**

**Aim: Write a program to demonstrate different numpy array creation techniques and different numpy methods**

**NumPy** stands for Numerical Python. It is a Python library used for working with an array. In Python, we use the list for purpose of the array but it's slow to process. NumPy array is a powerful N-dimensional array object and its use in linear algebra, Fourier transform, and random number capabilities. It provides an array object much faster than traditional Python lists.

**Types of Array:**

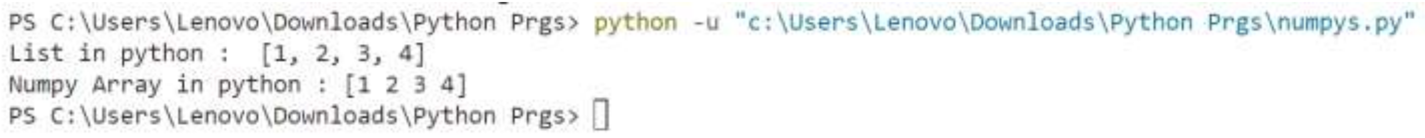
1. One Dimensional Array
2. Multi-Dimensional Array

**One Dimensional Array:**
A one-dimensional array is a type of linear array.

***One Dimensional Array***

```
# importing numpy module

import numpy as np


# creating list

my_list = [1, 2, 3, 4]

# creating numpy array

sample_array = np.array(my_list)

print("List in python : ", my_list)

print("Numpy Array in python :", sample_array)
```

**Output:**



```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\numpys.py"
List in python :  [1, 2, 3, 4]
Numpy Array in python : [1 2 3 4]
PS C:\Users\Lenovo\Downloads\Python Prgs> 
```

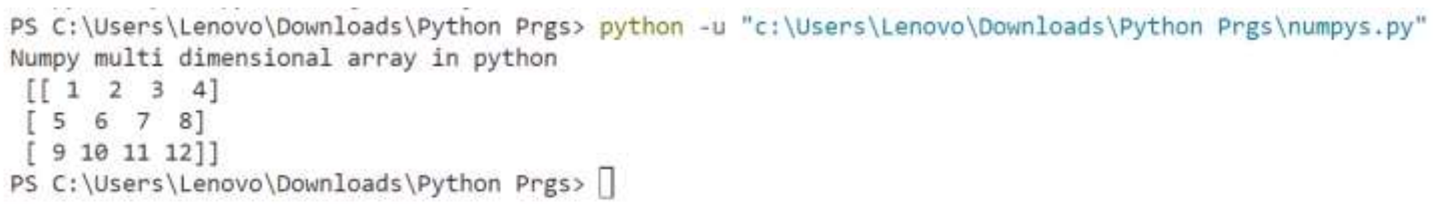
**Multi-Dimensional Array:**

Data in multidimensional arrays are stored in tabular form.

***Two Dimensional Array***

```
# importing numpy module

import numpy as np

# creating list

list_1 = [1, 2, 3, 4]

list_2 = [5, 6, 7, 8]

list_3 = [9, 10, 11, 12]

# creating numpy array

sample_array = np.array([list_1, list_2, list_3])

print("Numpy multi dimensional array in python\n", sample_array)
```

**Output:**



```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\numpys.py"
Numpy multi dimensional array in python
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
PS C:\Users\Lenovo\Downloads\Python Prgs> 
```

**1. Axis:** The Axis of an array describes the order of the indexing into the array.
*Axis 0 = one dimensional*

*Axis 1 = Two dimensional*

*Axis 2 = Three dimensional*

**2.Shape:** The number of elements along with each axis. It is from a tuple. **3. Rank:** The rank of an array is simply the number of axes (or dimensions) it has. **The one-dimensional array has rank 1.**

***Rank 1***
**The two-dimensional array has rank 2.**
***Rank 2***

**3. Data type objects (dtype):** Data type objects (dtype) is an instance of **numpy.dtype** class. It describes how the bytes in the fixed-size block of memory corresponding to an array item should be interpreted.

**Some different way of creating Numpy Array :**
**1. numpy.array():** The Numpy array object in Numpy is called **ndarray.** We can create ndarray using **numpy.array()** function.
***Syntax:*** *numpy.array(parameter)*

```
import numpy as np

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

print(arr)
```

**2. numpy.fromiter():** The fromiter() function create a new one-dimensional array from an iterable object.
***Syntax:*** *numpy.fromiter(iterable, dtype, count=-1)*

**3. numpy.arange():** This is an inbuilt NumPy function that returns evenly spaced values within a given interval.
***Syntax:*** *numpy.arange([start, ]stop, [step, ]dtype=None)*

**4. numpy.linspace():** This function returns evenly spaced numbers over a specified between two limits.
***Syntax:*** *numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)*

**5. numpy.empty():** This function create a new array of given shape and type, without initializing value.
***Syntax:*** *numpy.empty(shape, dtype=float, order='C')*

**6. numpy.ones():** This function is used to get a new array of given shape and type, filled with ones(1).
***Syntax:*** *numpy.ones(shape, dtype=None, order='C')*

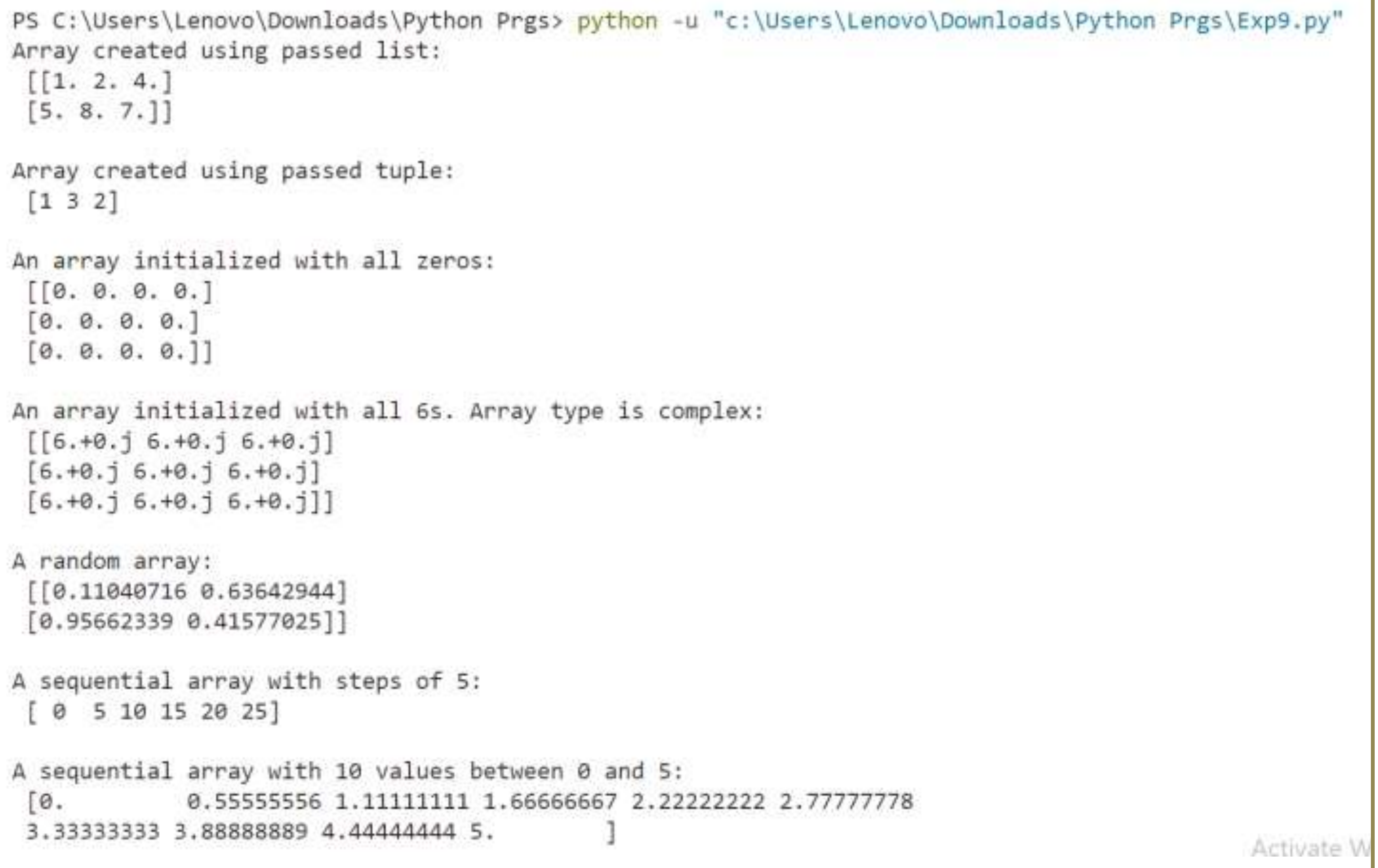
**7. numpy.zeros():** This function is used to get a new array of given shape and type, filled with zeros(0).
***Syntax:*** *numpy.ones(shape, dtype=None)*

**PROGRAM:**

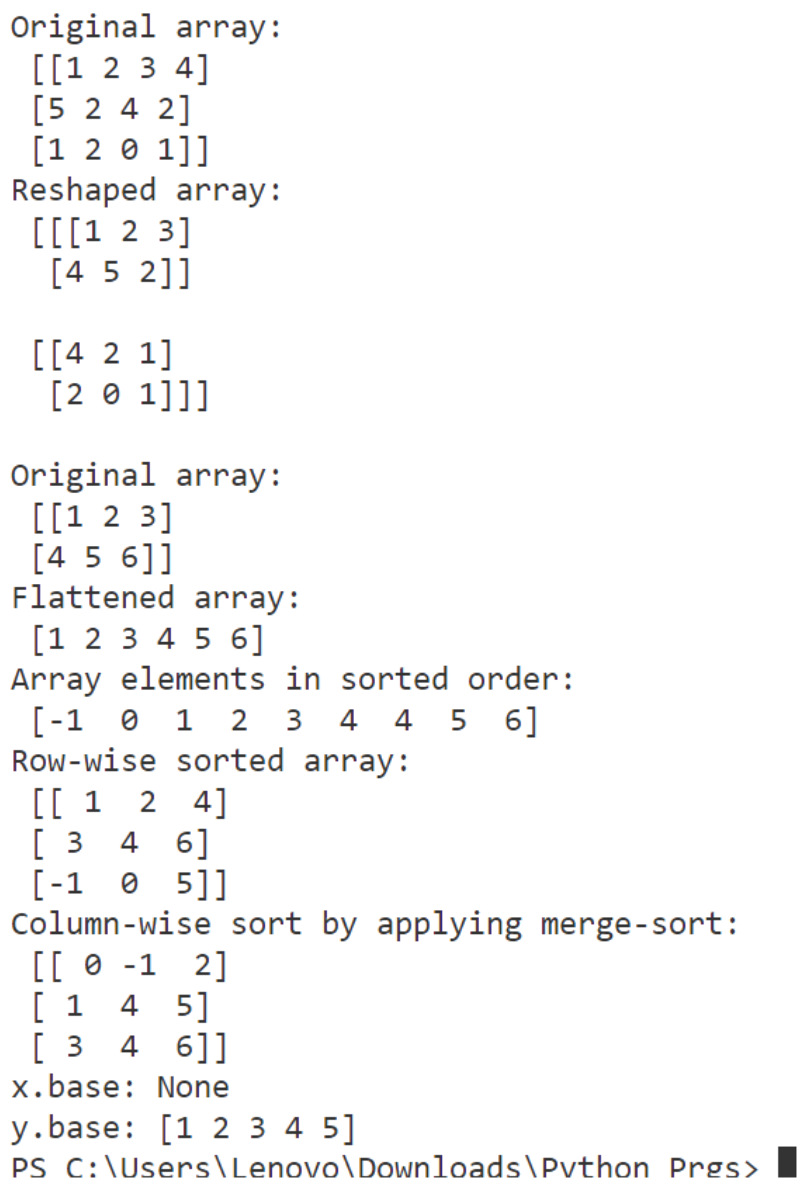
```
# Python program to demonstrate
# array creation techniques and different array methods.
import numpy as np

# Creating array from list with type float
a = np.array([[1, 2, 4], [5, 8, 7]], dtype='float')
print("Array created using passed list:\n", a)

# Creating array from tuple
b = np.array((1, 3, 2))  # Initialize b
print("\nArray created using passed tuple:\n", b)

# Creating a 3X4 array with all zeros
c = np.zeros((3, 4))
print("\nAn array initialized with all zeros:\n", c)

# Create a constant value array of complex type
d = np.full((3, 3), 6, dtype='complex')
print("\nAn array initialized with all 6s. Array type is complex:\n", d)

# Create an array with random values
e = np.random.random((2, 2))
print("\nA random array:\n", e)

# Create a sequence of integers from 0 to 30 with steps of 5
f = np.arange(0, 30, 5)
print("\nA sequential array with steps of 5:\n", f)

# Create a sequence of 10 values in range 0 to 5
g = np.linspace(0, 5, 10)
print("\nA sequential array with 10 values between 0 and 5:\n", g)

# Reshaping 3X4 array to 2X2X3 array
arr = np.array([[1, 2, 3, 4], [5, 2, 4, 2], [1, 2, 0, 1]])
newarr = arr.reshape(2, 2, 3)
print("\nOriginal array:\n", arr)
print("Reshaped array:\n", newarr)

# Flatten array
arr = np.array([[1, 2, 3], [4, 5, 6]])
flarr = arr.flatten()
print("\nOriginal array:\n", arr)
print("Flattened array:\n", flarr)

a = np.array([[1, 4, 2], [3, 4, 6], [0, -1, 5]])

# sorted array
print("Array elements in sorted order:\n", np.sort(a, axis=None))

# sort array row-wise
print("Row-wise sorted array:\n", np.sort(a, axis=1))
```

```
# specify sort algorithm
print("Column-wise sort by applying merge-sort:\n", np.sort(a, axis=0, kind='mergesort'))

# Demonstrate array copy and view
arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
y = arr.view()

print("x.base:", x.base)
print("y.base:", y.base)
```

**OUTPUT**



```
PS C:\Users\Lenovo\Downloads\Python Prgs> python -u "c:\Users\Lenovo\Downloads\Python Prgs\Exp9.py"
Array created using passed list:
 [[1. 2. 4.]
 [5. 8. 7.]]

Array created using passed tuple:
 [1 3 2]

An array initialized with all zeros:
 [[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]

An array initialized with all 6s. Array type is complex:
 [[6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]
 [6.+0.j 6.+0.j 6.+0.j]]

A random array:
 [[0.11040716 0.63642944]
 [0.95662339 0.41577025]]

A sequential array with steps of 5:
 [ 0  5 10 15 20 25]

A sequential array with 10 values between 0 and 5:
 [0.         0.55555556 1.11111111 1.66666667 2.22222222 2.77777778
 3.33333333 3.88888889 4.44444444 5.        ]
```



```
Original array:
 [[1 2 3 4]
 [5 2 4 2]
 [1 2 0 1]]
Reshaped array:
 [[[1 2 3]
  [4 5 2]]

 [[4 2 1]
  [2 0 1]]]

Original array:
 [[1 2 3]
 [4 5 6]]
Flattened array:
 [1 2 3 4 5 6]
Array elements in sorted order:
 [-1  0  1  2  3  4  4  5  6]
Row-wise sorted array:
 [[ 1  2  4]
 [ 3  4  6]
 [-1  0  5]]
Column-wise sort by applying merge-sort:
 [[ 0 -1  2]
 [ 1  4  5]
 [ 3  4  6]]
x.base: None
y.base: [1 2 3 4 5]
PS C:\Users\Lenovo\Downloads\Python Prgs> █
```

**Conclusion:**

We successfully showcased diverse techniques for creating numpy arrays and employing various numpy methods. This demonstration not only illustrated the flexibility and efficiency of numpy in array manipulation but also highlighted its utility in facilitating complexmathematical operations and data analysis tasks with ease.