Devanshu Surana
1032210755
PC-12.

PP Lab Assignment - 8
Prog Problem Statement
Use of Numpy library for multi-dimensional array operation.
Aim:
Write a python code to use a Numpy module create an array and checks the following:
1. Type of array. 2. Axes of array 3. Shape of Array
4. Type of elements in Array.

Objectives:
To learn and implement Functions of Numpy Library.

Theory:
Numpy Library and its functions:
- Numpy is a python library used for working with arrays. It also functions for working in domain of linear algebra, fourier transform and matrices.

- arrange() = creates an array by using the evenly spaced values over the given interval.
- shape = returns a tuple with each index having the no. of corresponding elements.
- reshape = means changing the shape of array.
- size = counts no. of elements on a given axis.
- ndim = returns an integer that tells how many dimension the array has.
- dtype = returns the data type of array.
- diag = extracts a diagonal or constructs a diagonal array.
- Slicing = It means taking elements from one given index to another given index.

Platform: Windows - Python Editor (Jupyter Notebook)

Conclusion:
Studied python numpy library functions on arrays.

FAQ's

Ans 1. Numpy Array:
- Can store only one data type in an array at any time.
- Need to install external library Numpy.
- No extra functions, so it will not store more memory.

List.
- Able to store different data types in the same list.
- Inbuilt data type.
- Has more inbuilt functions.

Ans 2) Type:
Python type() is a built-in functn that returns the type of data elements stored in any data type.

eg:
```
a = 10
b = 10.5
c = True
d = 1+5j
print (type(a))
print (type(b))
print (type(c))
print (type(d))
```

o/p
```
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'complex'>
```

dtype:
dtype gives information about type of data.

eg:
```
a = n.array([1.2, 3.5, 5.1, 4])
print (a)
a.dtype
```

o/p
```
[1.2, 3.5, 5.1, 4]
dtype ('float 64')
```

Ans 3. Axes are defined for arrays with more than one dimension. A 2-dimensional array has 2 corresponding axes - the first running vertically downwards across rows (axis 0) and the second running horizontally across columns (axis 1).

In [1]:
```
import numpy
arr = numpy . array ([1 , 2, 3, 4, 5])
print ( arr )

import numpy as np
a = np.array([1, 2, 3, 4, 5])
print(a)
print(type(a))
```

```
[1 2 3 4 5]
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

In [3]:
```
import numpy as np
arr = np. array (42)
print ( arr )
arr1 = np.array([1, 2, 3, 4, 5])
print(arr1)
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2)
arr3 = np.array([[[0, 1], [2, 3]], [[4, 5], [6, 7]]])
```

```
42
[1 2 3 4 5]
[[1 2 3]
 [4 5 6]]
```

In [17]:
```
import numpy as np
a = np.array([(1 ,2 ,3)])
print(a.dtype)
arr = a.astype('float64')
print(arr)
print(arr.dtype)
s = np.array(['Ram','Robert','Rahim'])
s.dtype
```

```
int32
[[1. 2. 3.]]
float64
```

Out[17]:
```
dtype('<U6')
```

In [16]:
```
a = np.array([1, 2, 3, 4])
print(a + 1)
print(a**2)
b = np.ones(4) + 1
print(b)
b = np.zeros((3,3)) + 1
print(b)
c = np.eye(3)
d = np.eye(3, 2)
a = np.diag([1, 2, 3, 4])
a
print(a[2, 2])
```

```
[2 3 4 5]
[ 1  4  9 16]
[2. 2. 2. 2.]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
3
```