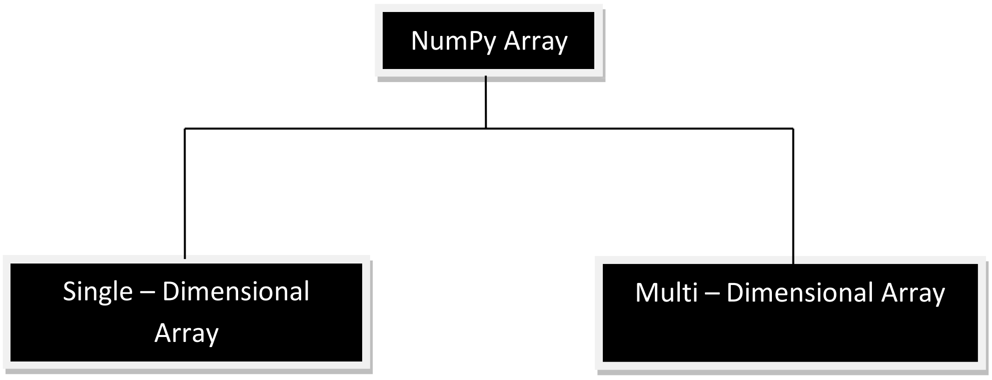
NumPy – Numerical and Scientific Computing with Python

1. What is NumPy?

- NumPy is a Python package that stands for 'Numerical Python'
- NumPy is a general-purpose array-processing package.
- It provides a high-performance multidimensional array object, and tools for working withthese arrays.
- It is the fundamental package for scientific computing with Python. It contains variousfeatures including these important ones:
  - A powerful N-dimensional array object
  - Sophisticated (broadcasting) functions
  - Tools for integrating C/C++ and Fortran code
  - Useful linear algebra, Fourier transform, and random number capabilities.

2. What are NumPy Arrays?

- It is the core library for scientific computing, which contains a powerful n-dimensional arrayobject.
- We can initialize numpy arrays from nested Python lists and access its elements.



3. Operations using NumPy:

Using NumPy we can perform following operations:

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulations.
- NumPy has in-built functions for operations related to linear algebra and random numbergeneration.

4. How to install NumPy?

Steps to install NumPy:

- Go to Command Prompt and type "pip install numpy"
- Once the installation is completed, go to your IDE (e.g. jupyter notebook) and simply import it by typing: "import numpy as np"

5. Creating NumPy Array:

- *Single Dimensional Array:* One dimensional array contains elements only in one dimension.In other words, the shape of the NumPy array should contain only one value in the tuple.

Example 1:

```
import numpy as                    #importing numpy as np

A = np.array([1,2,3,4])             #creating a one dimensional array using

print(A)                             #printing the array 'A'



Output: [1 2 3 4 ]
```

- *Multi Dimensional Array:* An ndarray is a (usually fixed-size) multidimensional container ofitems of the same type and size. The number of dimensions and items in an array is definedby its shape, which is a tuple of *N* non-negative integers that specify the sizes of each dimension.

Example 2:

```
#creating a two dimensional
                                        #importing numpy as np
B = np.array([(1,2,3), (4,5,6)])        #creating two dimensional
print(B)                                 #printing array 'B'



Output:  [ [1,2,3]
```

6 NumPy Vs Lists:

- NumPy data structures occupy less memory as compared to list.
- It is pretty much fast in terms of execution than list.
- It is more convenient to use as compared to list.

Example 3: To check how much size is occupied by List and NumPy.

```
#importing required libraries

import numpy as np

import time

import sys

c = range(1000)

print(“The space occupied by lists: ”, sys.getsizeof(c)*len(c))

d = np.arange(1000)

print(“The space occupied by numpy data structures: ”, d.size*d.itemsize)
```

```
Output: The space occupied by lists: 48000

          The space occupied by numpy data structures: 8000
```

7 NumPy Operations: It help us finding the:

- Dimension of array.
- Byte size of each element.
- Data.
- List item.

a. **ndim( ):** This function helps us to find the dimension of the array, whether it is amultidimensional array or a single dimensional array.

Example 5: to find the dimension of array

```
                                              #importing numpy as np

A = np.array([(1,2,3), (2,3,4)])              #creating

print(A.ndim)                               #printing the dimension of
```

```
Output: 2
```

a. **itemsize( ):** This function helps us to calculate the byte size of each element.

Example 6: To check the size of a single element in the array

```
import numpy as np

A = np.array([1, 2, 3, 4, 5, 6], dtype = np.int8)

print(A.itemsize)
```

```
Output: 1
```

a. **dtype( ):** Using this function we can find the data type of the elements stored in an array

Example 7: To find the data type of the elements of the array

```
import numpy as np

a = np.array([4.5, 22.8, 6.2, 1.1])

print("the data type of the elements of array is: ", a.dtype)
```

```
Output: the data type of the elements of array is: float64
```

b. **Size of array:** The size of array means the total number of elements present in thearray.

Example 8: To check the size of array

```
import numpy as np

a = np.array([(1,2,3,4,5,6)])
print(a.size)
```

Output: 6

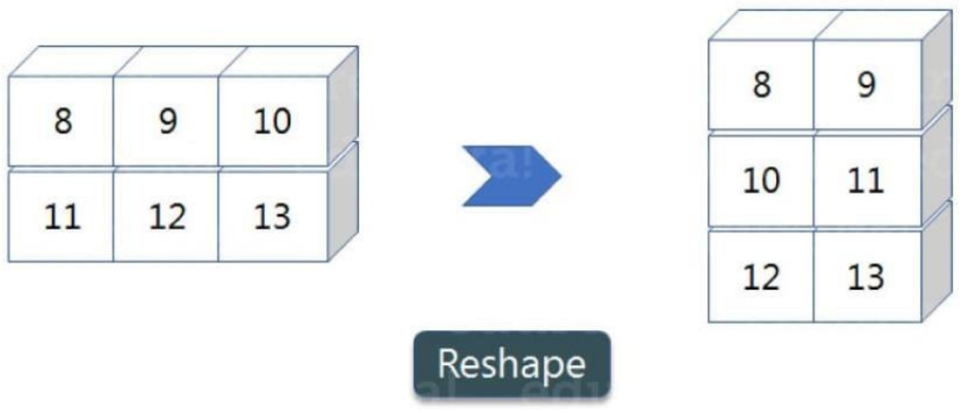
a. **Shape of Array:** It means the number of rows and columns present in it.

Example 9: To check the shape of the array.

```
import numpy as np

a = np.array([(1,2,3,4,5,6)])
print(a.shape)
```

Output: (1,6)

a. **numpy.reshape( ):** Reshape is when you change the number of rows and columns which gives a new view to an object. Now, let us take an example to reshape the below array:



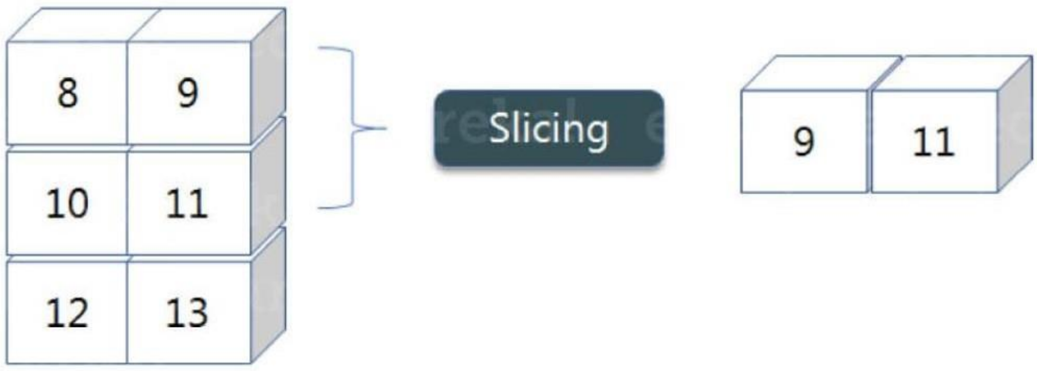
As you can see in the above image, we have 3 columns and 2 rows which has convertedinto 2 columns and 3 rows. Let us understand through an example:

Example 10:

```
import numpy as np

a = np.array([(8,9,10),(11,12,13)])
print(a)
a=a.reshape(3,2)
print(a)
```

```
Output:  [[ 8 9 10] [11 12 13]] [[ 8 9] [10 11] [12 13]]
```

a. **Slicing:** Slicing is basically extracting particular set of elements from an array. This slicing operation is pretty much similar to the one which is there in the list as well. Consider the following example:



Example 11: To slice ndarray

```
import  numpy  as  np

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

print( a [0, 2] )
```

Here, in the above example the array(1,2,3,4) is your index 0 and (3,4,5,6) is index 1 of the python NumPy array. Therefore, we have printed the second element from the zeroth index.

Example 12: To slice ndarray.

```
import numpy as np

a=np.array([(8,9),(10,11),(12,13)])


print(a[0:2,1])
```

Output: [9,11]

Hence, in the above code, only 9 and 11 gets printed. Now when I have written 0:2, this does not include the second index of the third row of an array. Therefore, only 9 and 11 gets printed else you will get all the elements i.e [9 11 13].

b. numpy.linspace( ):

- It is a tool in Python for creating numeric sequences.
- Creates sequences of evenly spaced values within a defined interval.
- We have to specify a starting point and ending point of an interval, and then specify the total number of breakpoints you want within that interval (including the start and end points).
- The np.linspace function will return a sequence of evenly spaced values on that interval.
- *Syntax:* np.linspace(start, stop, num) where
    - Start: indicates the point from which the sequence should start.
    - Stop: indicates the point from which the sequence should end.
    - Num: indicates the number of breakpoints required.

Example 13: To get the evenly spaced values of the given range

```
import numpy as np

a=np.linspace(1,3,10)
print(a)
```

Output: [1. 1.22222222 1.44444444 1.66666667 1.88888889 2.11111111 2.33333333 2.55555556 2.77777778 3.]

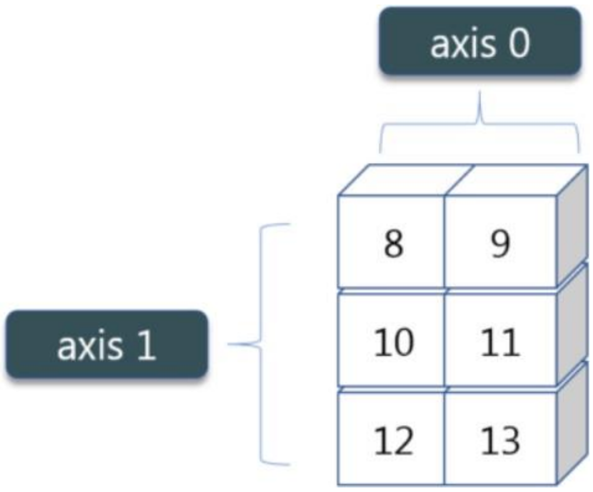
In the above example, in the result, it has printed 10 values between 1 to 3.

c. **max, min and sum:** These functions can be used to find the maximum, minimum values and the sum of NumPy array.

Example 14: To find the minimum, maximum and sum

```
import numpy as np

a= np.array([1,2,3])

print("The minimum value is ", a.min())

print("The maximum value is ", a.max())

print("The sum comes ", a.sum())


Output: the minimum value is 1

The maximum value is  3

The sum comes 6
```

d. **Axis 0 and Axis 1:** As you can see in the figure, we have a NumPy array 2*3. Here the rows are called as axis 1 and the columns are called as axis 0. We can use this function when need to calculate the sum of all the columns.



Example 15:

```
import numpy as np a=np.array([(2,4),(5,8)])

#prints the sum of all the columns

print(“the sum of all the columns = ”, a.sum(axis = 0))#prints the

sum of all the rows

print(“the sum of all the rows = ”, a.sum(axis = 1))
```

```
Output: the sum of all the columns = [ 7 12 ]
```

Therefore, the sum of all the columns are added where 2+5=7, 4+8=12 . Similarly, if youreplace the axis by 1, then it will print [6 13] where all the rows get added.

e. **Square Root and Standard Deviation:** We can also find the square root and standard deviation (i.e. how much element varies from the mean value of Python NumPy array) of array using the following functions:
   - Sqrt(name of array)
   - Std(name of array)

Example 16: To find the square root and standard deviation

```
import numpy as np

a=np.array([(2, 3, 4),(16, 25, 30)])

#prints the square root of each element of the array
print(“Square root: ”, np.sqrt(a))

#prints the standard deviation of the elements
print(“Standard Deviation: ”, np.std(a))
```

```
Output:
Square root: [[ 1.41421356 1.73205081 2. ] [4. 5. 5.47722558]]
```

8. **Basic Mathematical Operations:** We can also perform the basic mathematical operationssuch as addition, subtraction, multiplication and division.

Example 17: To perform addition of two ndarrays

```
import numpy as np

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

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

print("Addition: ", x+y)
```

Output:

```
Addition: [[ 2 4 6] [ 6 8 10]]
```

Thus, we can see in the output that the addition is performed element by element as it isdone in the case of matrices.

9 References:

- https://www.edureka.co/blog/python-numpy-tutorial/
- https://www.tutorialspoint.com/numpy/index.htm
- https://www.w3schools.com/python/numpy_creating_arrays.asp
- https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/

10 Video Link:

- https://www.youtube.com/watch?v=xECXZ3tyONo&feature=youtu.be