NUMPY:

*It is the fundamental package for scientific computing with python.It is a python c extention library for array oriented computing.*Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined and this allows NumPy to seamlessly and speedily integrate with a wide variety of projects.

In [3]:
```
#example to print single dimentional array usig numpy
import numpy as np
a=np.array([1,2,3])
print(a)
```

```
[1 2 3]
```

In [4]:
```
#example to print multi dimentional array
a=np.array([(1,2,3),(4,5,6)])
print(a)
```

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

We use python NumPy array instead of a list because of the below three reasons:

Less Memory Fast *Convenient

In [8]:
```
#example to show that numpy array utilizes less memory than the list
import numpy as np

import time
import sys
S= range(1000)
print(sys.getsizeof(5)*len(S))

D= np.arange(1000)
print(D.size*D.itemsize)
```

```
28000
4000
```

In [9]:
```
#example to show that NumPy array is faster and more convenient when compar
import time
import sys

SIZE = 1000000

L1= range(SIZE)
L2= range(SIZE)
A1= np.arange(SIZE)
A2=np.arange(SIZE)

start= time.time()
result=[(x,y) for x,y in zip(L1,L2)]
print((time.time()-start)*1000)

start=time.time()
result= A1+A2
print((time.time()-start)*1000)
```

```
262.32385635375977
63.80128860473633
```

NUMPY OPERATIONS

1)ndim: You can find the dimension of the array, whether it is a two-dimensional array or a single dimensional array.

In [10]:
```
import numpy as np
a = np.array([(1,2,3),(4,5,6)])
print(a.ndim)
```

```
2
```

2)itemsize:You can calculate the byte size of each element. In the below code, I have defined a single dimensional array and with the help of 'itemsize' function, we can find the size of each element.

In [14]:
```
import numpy as np
a = np.array([(1,2,3)])
print(a.itemsize)
```

```
4
```

3)dtype:You can find the data type of the elements that are stored in an array. So, if you want to know the data type of a particular element, you can use 'dtype' function which will print the datatype along with the size. In the below code, I have defined an array where I have used the same function.

In [15]:
```
import numpy as np
a = np.array([(1,2,3)])
print(a.dtype)
```

```
int32
```

4)size and shape:you can find the size and shape of the array using 'size' and 'shape' function respectively.

In [16]:
```
import numpy as np
a = np.array([(1,2,3,4,5,6)])
print(a.size)
print(a.shape)
```

```
6
(1, 6)
```

5)reshape:Reshape is when you change the number of rows and columns which gives a new view to an object.

In [17]:
```
import numpy as np
a = np.array([(8,9,10),(11,12,13)])
print(a)
a=a.reshape(3,2)
print(a)
```

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

6)slicing: Slicing is basically extracting particular set of elements from an array.

In [18]:
```
#example to extract a single element 3 from the array
import numpy as np
a=np.array([(1,2,3,4),(3,4,5,6)])
print(a[0,2])
```

```
3
```

In [20]:
```
#example for extracting elements in 2nd postion from all the arrays
import numpy as np
a=np.array([(1,2,3,4),(3,4,5,6)])
print(a[0:,2])
#here ':' represents all the rows.
```

```
[3 5]
```

7)linspace:This is another operation in python numpy which returns evenly spaced numbers over a specified interval.

In [21]:
```
import numpy as np
a=np.linspace(1,3,10)
print(a)
```

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

8)max,min and sum:this calculates the maximum,minimum and sum of the numpy array.

In [22]:
```
import numpy as np

a= np.array([1,2,3])
print(a.min())
print(a.max())
print(a.sum())
```

```
1
```

```
3
6
```

9)axis:Suppose you want to calculate the sum of all the columns, then you can make use of axis.

In [27]:
```
a= np.array([(1,2,3),(3,4,5)])
print(a.sum(axis=0))
```

```
[4 6 8]
```

Therefore, the sum of all the columns are added where 1+3=4, 2+4=6 and 3+5=8.

10)squareroot and standard deviation:There are various mathematical functions that can be performed using python numpy. You can find the square root, standard deviation of the array.

In [28]:
```
import numpy as np
a=np.array([(1,2,3),(3,4,5,)])
print(np.sqrt(a))
print(np.std(a))
```

```
[[1.         1.41421356 1.73205081]
 [1.73205081 2.         2.23606798]]
1.2909944487358056
```

11)addition operation

In [29]:
```
import numpy as np
x= np.array([(1,2,3),(3,4,5)])
y= np.array([(1,2,3),(3,4,5)])
print(x+y)
```

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

12)subtraction operation

In [30]:
```
import numpy as np
x= np.array([(1,2,3),(3,4,5)])
y= np.array([(1,2,3),(3,4,5)])
print(x-y)
```

```
[[0 0 0]
 [0 0 0]]
```

13)multiplication operation

In [31]:
```
import numpy as np
x= np.array([(1,2,3),(3,4,5)])
y= np.array([(1,2,3),(3,4,5)])
print(x*y)
```

```
[[ 1  4  9]
 [ 9 16 25]]
```

14)division operation

In [32]:
```
import numpy as np
x= np.array([(1,2,3),(3,4,5)])
y= np.array([(1,2,3),(3,4,5)])
print(x/y)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

15)vertical and horizontal stocking

In [35]:
```
import numpy as np
x= np.array([(1,2,3),(3,4,5)])
y= np.array([(1,2,3),(3,4,5)])
print(np.vstack((x,y)))
print(np.hstack((x,y)))
```

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

16)ravel:There is one more operation where you can convert one numpy array into a single column

In [36]:
```
import numpy as np
x= np.array([(1,2,3),(3,4,5)])
print(x.ravel())
```

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

In [ ]:
```
```