```
# introduction to numpy
# numpy is an open source library that is used for handle scintific calculat
# different b/w numpy and array---->
# (1) list hetrogeneous data type and numpy is homogeneous data type.
# (2) list takes more memory as compare to numpy.
# (3) list takes more execution time as compare to numpy.
```

```
import numpy as np
li=[x for x in range(1,10)]
```

```
import numpy as np
a=np.ones((5,5))
a.shape
a.size
a.ndim
a
```

```
a=np.array([1,25])
a
b=np.ones((5,5))
b
```

```
np.zeros((5,5),dtype=int)
```

```
#import numpy as np
#a=np.array([1,2,3,4,5])
#np.arange(1,91,1).reshape(8,8)
```

```
import numpy as np
```

```
a=[1,2,34,4,5,67]
type(a)
```

```
b=np.array(a)
b
```

```
type(b)
```

```
# user defined array
```

```
a=[]
size=int(input("enter size "))
for i in range(size):
  val=int(input("enter number"))
```

```
  a.append(val)
b=np.array(a)
b
```

```
import numpy as np
a= [[[1,2,3],[4,5,6],[4,5,6]]]
b = np.array(a)
b
```

```
type(a)
```

```
b=np.arange(2,20,2).reshape(3,3)
b
```

```
# matrix----> rows coloms
# row(1)=[1,2,3]
# row(2)=[4,5,6]
# row(3)=[4,7,8]
# col(1)=[1,4,4]
#col(2)=[2,5,7]
#col(3)=[3,6,8]
# total element=col*row=3*3=9
```

```
print("total rows and  columns:",b.shape)
print("total element:",b.size)
```

```
a=[[1,2,4,7],[3,4,5,3],[3,6,7,5]]
b=np.array(a)
b
```

```
print("total rows and  columns:",b.shape)
print("total element:",b.size)
```

```
# how to check dimension of an matrix.
```

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

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

```
a.ndim
```

```

```

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

```
a.ndim
```

```
# images----> pixels(0-255)----> normalize(0,1)----> matrix----> operations
# dl----> artifical neurous---> weight----> randonly creation----> numpy
# datasets----> dummy entry----> random module----> attach pandas
```

```
# predefined functions in numpy.
```

```
#(1) zero()----> it will create an array in which all the value will be one
```

```
import numpy as np

a=np.zeros(3)
a
```

```
a=np.zeros((3,4))
a
```

```
#(2) ones()----> it will create an array in which all the values will be 1 i
```

```
import numpy as np
a=np.ones((3))
a
```

```
import numpy as np
a=np.ones((3,5))
a
```

```
# (3).eye---> it will create an array in which digonal positional element wi
# symmetric matrix
a=np.eye(3)
a
```

```
# assymetric matrix
a=np.eye(3,4)
a
```

```
# (4) diag---->it will create an array in which we can set any custom values
a=np.diag([1,2,3,4])
a
```

```

```

```
# (5) random module
# (a) randint()---> it will create an array create randomly in a given range
# syntax: np.random.randint(min_num,max_num,total_num)
```

```
import numpy as np
```

```
import numpy as np
a=np.random.randint(1,5,2)
a
```

```
a=np.random.rand(4)
a
```

```
np.random.seed(23)
a=np.random.randint(1,5,2)
a
```

```
# view vs copy
```

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

```
b=a[1:4]
b[:]=0
b
```

```
b=a[1:4]
b[:]=0
b
```

```
a
```

```
# copy
```

```
a=np.array([1,2,3,4,5,6])
b=a[1:4].copy()
b[:]=0
a
```

```
# reshaping the array
# total_row*coloum=total_element
```

```
a=np.random.randint(1,50,12)
a
```

```
a.reshape(2,6)
```

```
a.reshape(3,3)
```

```
a.reshape(-1, 3)
```

```
a.shape
```

```
a.reshape(1,-2)
```

```
# operations on array
```

```
a=np.arange(1,9,3)
a
```

```
a=np.arange(1,10)
a
```

```
a*2
```

```
a=np.arange(5,9)
a
```

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

```
a=np.arange(4,8).reshape(2,2)
a
```

```
b=np.arange(5,9).reshape(2,2)
b
```

```
a+b
```

```
a-b
```

```
a.dot(b)
```

```
a=np.arange(1,8)
a
```

```
a>5
```

```
b=a>5
a[b]
```

```
a=np.arange(1,8)
a**2
```

```
a**3
```

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

```
np.sum(a)
```

```
np.sum(a,axis=1)                    # row operation
a
```

```
a=np.sum(a,axis=0)                # column operation
a
```

```
# unique()----> it will create  an array in which it will remove duplicate v
# arr ----> unique elements of array.
# return_index = true ----> indexing of unique element.
# return_counts = true ----> frequency of each unique element.
```

```
a = np.array([1,2,3,4,1,3,5,2,7,2,5,6,7])
a
```

```
b = np.unique(a,return_index = True, return_counts = True)
b
```

```
# linspace()----> it will return an array in which element will be same gap
# syntax : np.linspace(min_num , max_num ,total_numbers)
```

```
a = np.linspace(1,2,5)
a
```

```
# horizontal and vertical stacking---->
# horizontal stacking ----> it will add 2 and more arrays in horizontally.
```

```
a = np.arange(1,5)
a
```

```
b = np.arange(5,9)
b
```

```
c = np.arange(9,13)
c
```

```
np.hstack((a,b,c))
```

```
np.vstack((a,b,c))
```