NumPy:

TO IMPORT LIBRARY:

import numpy as np or import numpy

Syntax:

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

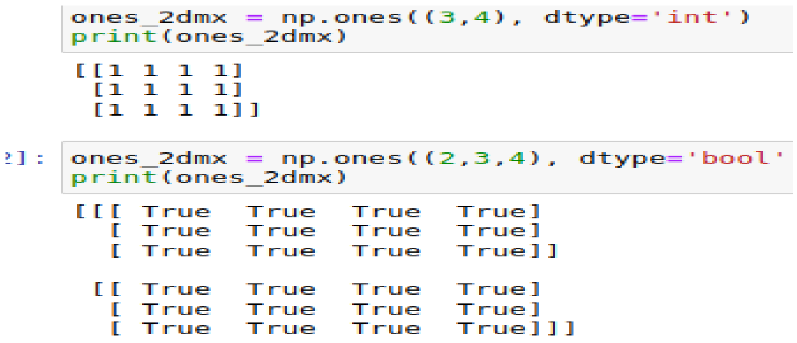
(3d array)

Basic Functions of NumPy:

a.shape : (dimension,row,column)
a.size : No. of elements in an array
a.ndim : Shows the dimension

Create an ones matrix:

a = np.ones((dim,row,column),dtype = “int/bool”)
For ex.,



```
ones_2dmx = np.ones((3,4), dtype='int')
print(ones_2dmx)
```

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

```
2]: ones_2dmx = np.ones((2,3,4), dtype='bool'
print(ones_2dmx)
```

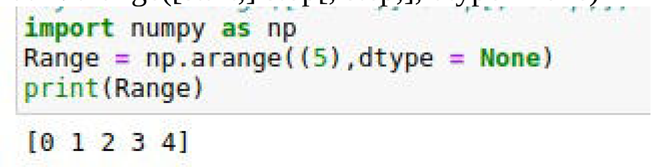
```
[[[ True  True  True  True]
  [ True  True  True  True]
  [ True  True  True  True]]

 [[ True  True  True  True]
  [ True  True  True  True]
  [ True  True  True  True]]]
```

Cretae a zeros matrix:

a = np.zeros((dim,row,column),dtype = “int/bool”)

Arrange Function :

synatx : arange([start,] stop[, step,], dtype=None)

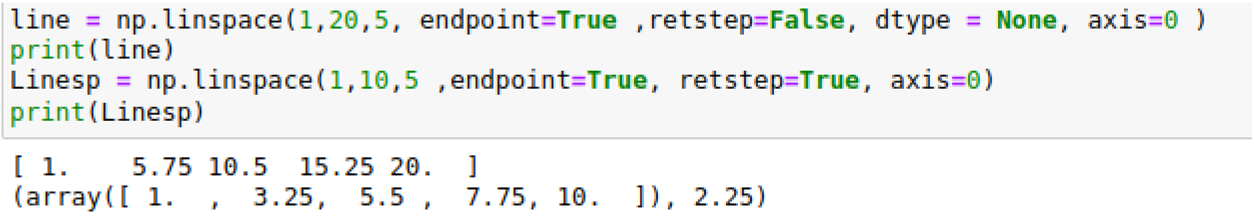


```
import numpy as np
Range = np.arange((5),dtype = None)
print(Range)
```

```
[0 1 2 3 4]
```

Linespace():

Syntax : np.linspace(start,stop,num=50,endpoint=True,retstep=False,dtype=None,axis=0)

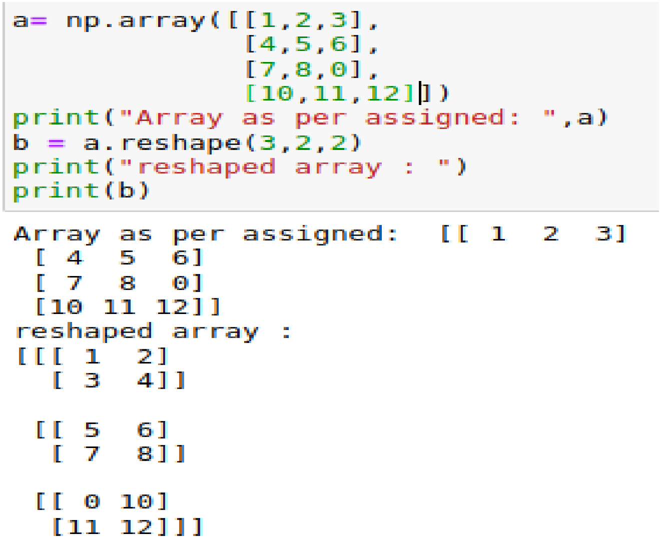


```
line = np.linspace(1,20,5, endpoint=True ,retstep=False, dtype = None, axis=0 )
print(line)
Linesp = np.linspace(1,10,5 ,endpoint=True, retstep=True, axis=0)
print(Linesp)
```

```
[ 1.    5.75 10.5  15.25 20.  ]
(array([ 1.  ,  3.25,  5.5 ,  7.75, 10.  ]), 2.25)
```

reshape():

Syntax : np.reshape(a,new_shape)
Reshape the elements



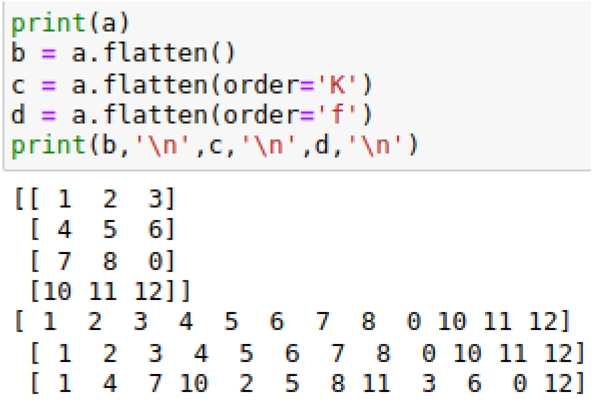
```
a= np.array([[1,2,3],
             [4,5,6],
             [7,8,0],
             [10,11,12]])
print("Array as per assigned: ",a)
b = a.reshape(3,2,2)
print("reshaped array : ")
print(b)
```

```
Array as per assigned:  [[ 1  2  3]
 [ 4  5  6]
 [ 7  8  0]
 [10 11 12]]
reshaped array : 
[[[ 1  2]
  [ 3  4]]

 [[ 5  6]
  [ 7  8]]

 [[ 0 10]
  [11 12]]]
```

ravel():

Convert to 1D array of any array
Syntax : b.ravel()

flatten(): convert according to the order



```
print(a)
b = a.flatten()
c = a.flatten(order='K')
d = a.flatten(order='f')
print(b,'\n',c,'\n',d,'\n')
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  0]
 [10 11 12]]
[ 1  2  3  4  5  6  7  8  0 10 11 12]
 [ 1  2  3  4  5  6  7  8  0 10 11 12]
 [ 1  4  7 10  2  5  8 11  3  6  0 12]
```

transpose():

converts row into columns and vice-versa

Operations:

np.add(array1,array2)
np.subract(array1,array2)
np.multiply(array1,array2)

Matrix Product:

matrix multiplication : array.dot()
to find maximum value in an array : array.max()

to find the index of maximum value: array.argmax()
mean : array.mean()
square : b = np.square(array)
square root : b = np.sqrt(array)
stanrdization: array.std()
log : np.log(array)
log $_{10}$:np.log10(array)
exponent: np.exp(array)

Slicing of an array:

It can be done in only two ways: row and column
array[row,column,step]

concatenate :

concate: np.concatenate((array1,array2))
vertical-concate : np.vstack((array1,array2))
horizontal-concate : np.hstack((array1,array2))
transpose-concate : np.concatenate((array1,array2)).T

Split:

To split the function :
np.split(array, no. of parts,axis =0/1)   (if axis=0 ->split in rows,axis=1=> split in columns)

Trignometric functions :

degree: np.sin(degree)
radian : np.sin(180 *np.pi/180)

String operation, Comparison and Information:

To add two characters : np.char.add(string1,string2)

To multiply two characters : np.char.multiply(string1,string2)

To do in lower case : np.char.lower(string1)

To split: np.char.split(string1)