```
#import numpy package
import numpy as np

sayan = [2,3,5,6,7,9,6]

type(sayan)

list

#converted the list into array
arr = np.array(sayan)

type(arr)

numpy.ndarray

arr

array([2, 3, 5, 6, 7, 9, 6])

arr.shape

(7,)

#reshape the array
arr.reshape(1,7)

array([[2, 3, 5, 6, 7, 9, 6]])

#Create multidimensional array
lst1 = [1,2,3,9]
lst2 = [4,5,6,8]
lst3 = [7,8,9,3]

 arr2 = np.array([lst1,lst2,lst3])

arr2

array([[1, 2, 3, 9],
       [4, 5, 6, 8],
       [7, 8, 9, 3]])

arr2.shape

(3, 4)

arr2.reshape(6,2)

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

```
arr2.shape
```

```
(3, 4)
```

Indexing in 1-Dimentional Array

```
arrr = [2,3,5,6,7,9,6,8]
```

```
arrr
```

```
[2, 3, 5, 6, 7, 9, 6, 8]
```

```
type(arrr)
```

```
list
```

```
arrr = np.array(arrr)
```

```
type(arrr)
```

```
numpy.ndarray
```

```
arrr[3]
```

```
6
```

Indexing in 2-Dimentional Array

```
lst1 = [1,2,3,9]
lst2 = [4,5,6,8]
lst3 = [7,8,9,3]
```

```
 arr2 = np.array([lst1,lst2,lst3])
```

```
arr2
```

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

```
arr2[:,:] #left side is row index
          #rifght side is column index
          #if we do not specify any value this show entire array
```

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

```
 arr2[0:3,0:2]
```

```
array([[1, 2],
       [4, 5],
       [7, 8]])

arr2[0:2,2:]

array([[3, 9],
       [6, 8]])

arr2[1:,2:]

array([[6, 8],
       [9, 3]])

arr2[1:,1:3]

array([[5, 6],
       [8, 9]])

arr2[1:2,0:]

array([[4, 5, 6, 8]])
```

IN BUILT FUNCTION

```
arr3 = np.arange(0,20) # 20 not print

arr3

array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
16,
       17, 18, 19])

arr4 = np.arange(0,20,step=2) # It take gaps 2 between two number

arr4

array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])

arr5 = np.linspace(1,10,100) #it print 100 number between 1 and 100

arr5

array([ 1.        ,  1.09090909,  1.18181818,  1.27272727,
1.36363636,
        1.45454545,  1.54545455,  1.63636364,  1.72727273,
1.81818182,
        1.90909091,  2.        ,  2.09090909,  2.18181818,
2.27272727,
        2.36363636,  2.45454545,  2.54545455,  2.63636364,
2.72727273,
        2.81818182,  2.90909091,  3.        ,  3.09090909,
3.18181818,
        3.27272727,  3.36363636,  3.45454545,  3.54545455,
3.63636364,
        3.72727273,  3.81818182,  3.90909091,  4.        ,
4.09090909,
        4.18181818,  4.27272727,  4.36363636,  4.45454545,
4.54545455,
        4.63636364,  4.72727273,  4.81818182,  4.90909091,
5.        ,
        5.09090909,  5.18181818,  5.27272727,  5.36363636,
5.45454545,
        5.54545455,  5.63636364,  5.72727273,  5.81818182,
5.90909091,
        6.        ,  6.09090909,  6.18181818,  6.27272727,
6.36363636,
        6.45454545,  6.54545455,  6.63636364,  6.72727273,
6.81818182,
        6.90909091,  7.        ,  7.09090909,  7.18181818,
7.27272727,
        7.36363636,  7.45454545,  7.54545455,  7.63636364,
7.72727273,
        7.81818182,  7.90909091,  8.        ,  8.09090909,
8.18181818,
        8.27272727,  8.36363636,  8.45454545,  8.54545455,
8.63636364,
        8.72727273,  8.81818182,  8.90909091,  9.        ,
9.09090909,
        9.18181818,  9.27272727,  9.36363636,  9.45454545,
9.54545455,
        9.63636364,  9.72727273,  9.81818182,  9.90909091,
10.        ])
```

```
#copy function and broadcasting
arrrr = [1,2,3,4,5,6,7,8,9,10]
arrrr
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
type(arrrr)
```

```
list
```

```
arrrr = np.array(arrrr)
```

```
type(arrrr)
```

```
numpy.ndarray
```

```
arr7=arrrr # arrrr will copy to arr7
```

```
arr7
```

```
array([ 1,  2,  3, 50, 50, 50, 50, 50, 50, 50])

arrrr[3:] = 50

arrrr

array([ 1,  2,  3, 50, 50, 50, 50, 50, 50, 50])

arrrr

array([ 1,  2,  3, 50, 50, 50, 50, 50, 50, 50])

arr7


array([ 1,  2,  3, 50, 50, 50, 50, 50, 50, 50])

val=2

arr7*val #all the elements in arr7 multiply by 2

array([  2,   4,   6, 100, 100, 100, 100, 100, 100, 100])

arr7<val #all the elements in arr7 which is less then 2 return true
else return false

array([ True, False, False, False, False, False, False, False, False,
       False])

#create array and reshape
ar = np.arange(0,10).reshape(2,5)

ar2 = np.arange(0,10).reshape(2,5)

ar

array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])

ar2

array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])

ar*ar2

array([[ 0,  1,  4,  9, 16],
       [25, 36, 49, 64, 81]])

one = np.ones(6)

one

array([1., 1., 1., 1., 1., 1.])
```

```
one_1 = np.ones((2,5),dtype=int)

one_1

array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])

#select random value of a given shape
ran = np.random.rand(3,3)
ran

array([[0.98143114, 0.74120911, 0.26968709],
       [0.57397883, 0.08537979, 0.63062547],
       [0.38074246, 0.6631301 , 0.20788108]])

#random standard distribution
ran1 = np.random.randn(4,4)
ran1

array([[-0.12053131,  0.15646078, -1.30802853,  1.27837213],
       [-0.7593452 ,  0.24689049,  0.26410463,  0.08335342],
       [-0.21323286, -2.18102882, -1.03865189,  0.39098902],
       [ 1.00243049, -0.24686629, -0.61408231,  0.82394226]])

#between 0 to 100 it take 8 elements randomly
np.random.randint(0,100,8)

array([37,  4, 55, 62, 89, 57, 81, 83])

#Return random floats in half-open interval [0.0,1.0]
#here(1,6)is shape
np.random.random_sample((1,6))

array([[0.61286026, 0.8462877 , 0.46012941, 0.58772981, 0.68994551,
        0.17164959]])
```