```
import numpy
```

```
import numpy as np
```

```
a = np.array([1,2,3,44,5,76]) #creating a 1-D array using the method array() --> argument is l
type(a)
```

```
numpy.ndarray
```

```
b  = np.array([4,6,73,12,23])
b
```

```
array([ 4,  6, 73, 12, 23])
```

```
a
```

```
array([ 1,  2,  3, 44,  5, 76])
```

```
a[0] #accessing 0th element of the array a
```

```
1
```

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

```
a+b #adding two arrays -->same dimensions
```

```
array([ 5,  7, 75, 15, 27])
```

```
a+2 #adding any number to each element of the array
```

```
array([3, 3, 4, 5, 6])
```

```
a-3 #subtraction to each element
```

```
array([-2, -2, -1,  0,  1])
```

```
a*2  #multiplying with each element
```

```
array([2, 2, 4, 6, 8])
```

```
a*b #multiplication
```

```
array([  4,   6, 146,  36,  92])
```

```
p = np.array([[1,2,3],[2,4,6]]) #2-D array --> form ([[row1],[row2],...]) --> number of col i
```

```
p
```

```
array([[1, 2, 3],
       [2, 4, 6]])
```

```
p[1]
```

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

```
p[1][2]
```

```
6
```

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

```
array([[1, 2, 2],
       [3, 4, 5]])
```

```
p+q
```

```
array([[ 2,  4,  5],
       [ 5,  8, 11]])
```

```
p*q
```

```
array([[ 1,  4,  6],
       [ 6, 16, 30]])
```

```
p*2
```

```
array([[ 2,  4,  6],
       [ 4,  8, 12]])
```

```
p**3
```

```
array([[  1,   8,  27],
       [  8,  64, 216]])
```

```
x = np.array([[2,7],[3,8],[2,9],[9,4]])
x
```

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

```
x.shape #no. of rows and columns
```

```
x[0:2] #
```

```
array([[2, 7],
       [3, 8]])
```

```
x[0:,0] #format x[row_range,col_range]
```

```
array([2, 3, 2, 9])
```

```
p[0:,[0,2]] #discrete columns --> use col indexes as a list
```

```
array([[1, 3],
       [2, 6]])
```

```
x[[0,2]] #discrete rows --> use row indexes as a list
```

```
array([[2, 7],
       [2, 9]])
```

```
x
```

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

```
x[[0,3],[1]]
```

```
array([7, 4])
```

```
c = np.array([[1,9,7],[3,2,8],[4,5,6],[2,9,0]])
c
```

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

```
c[:,1]
```

```
array([9, 2, 5, 9])
```

```
c[[1,2],1]
```

```
array([2, 5])
```

```
c[1:4,2]
```

```
array([8, 6, 0])
```

```
c[[1,2,3],2]
```

```
array([8, 6, 0])
```

```
c[1:,[0,2]]
```

```
array([[3, 8],
       [4, 6],
       [2, 0]])
```

```
x
```

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

```
p
```

```
array([[1, 2, 3],
       [2, 4, 6]])
```

```
p.reshape(3,2) #change the dimensions of an existing array
```

```
array([[1, 2],
       [3, 2],
       [4, 6]])
```

```
np.zeros((3,6)) #creating a zero matrix --> by defautl --> float
```

```
array([[0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

```
np.zeros((3,4),dtype = int) #for changing the data type
```

```
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]])
```

```
np.ones((2,3)) #all 1's
```

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

```
np.full((3,6),2) #for number >=2
```

```
       array([[2, 2, 2, 2, 2, 2],
```

```
for i in dir(np): #displays all the methods of the module np
  if 'zer' in i:
    print(i)
```

```
count_nonzero
flatnonzero
nonzero
trim_zeros
zeros
zeros_like
```

```
[i for  i  in dir(np)    if 'zer' in i]
```

```
['count_nonzero',
 'flatnonzero',
 'nonzero',
 'trim_zeros',
 'zeros',
 'zeros_like']
```

```
a
```