In [6]:
```
import numpy as np

x=[[81,71,57,63],[54,68,81,45]]
print(type(x)) #list
test_scores  = np.array(x) #array
print(type(test_scores))
print(test_scores)
pass_score= test_scores>60 #test criteria
print(pass_score)
print("display student who have passed")
print(test_scores[pass_score]) #Retriving the students who have scored greater than 60
```

```
<class 'list'>
<class 'numpy.ndarray'>
[[81 71 57 63]
 [54 68 81 45]]
[[ True  True False  True]
 [False  True  True False]]
display student who have passed
[81 71 63 68 81]
```

In [15]:
```
#Example of 2D array
import numpy as np
listOfNumbers = [ [2,3,5,7],[1,9,24,15],[5,12,19,21]]
arrOfNums= np.array(listOfNumbers) #This will create an ndarray object
print(arrOfNums)
arrOfNums.ndim #ndim tells us the dimension of the ndarray object
```

```
[[ 2  3  5  7]
 [ 1  9 24 15]
 [ 5 12 19 21]]
```

Out[15]:
```
2
```

In [14]:
```
#Example of 1D array -- using tuple also we can create a 2 D array
#import numpy as np
tupleOfNumbers=(1,2,3,4,5,6)
arrOfTuple= np.array(tupleOfNumbers)
print(arrOfTuple)
print(type(arrOfTuple))
arrOfTuple.ndim #ndim tells us the dimension of the ndarray object
```

```
[1 2 3 4 5 6]
<class 'numpy.ndarray'>
```

Out[14]:
```
1
```

In [20]:
```
#Example of 3 D array
listOfNumbers = [[[2,3,5,7],[1,9,24,15]],[[5,12,19,21],[1,2,3,4]]]
threeDarray=np.array(listOfNumbers)
print(threeDarray)
threeDarray.ndim #ndim tells us the dimension of the ndarray object
threeDarray.size
```

```
[[[ 2  3  5  7]
  [ 1  9 24 15]]

 [[ 5 12 19 21]
  [ 1  2  3  4]]]
```

Out[20]:
```
16
```

In [19]:
```
#ndmin -- An array can have any number of dimension.At the time of creation of an array we can define the dimension
x=[1,2,3,4,5,6]
nDimArr=np.array(x,ndmin=5)
print(nDimArr)
print(nDimArr.ndim)
nDimArr.size
```

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

Out[19]:
```
6
```

In [24]:
```
#size
x=[1,2,3,4,5,6]
arrOfX=np.array(x)
print(arrOfX)
arrOfX.size #Gives us the count of elements or items present in the array
```

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

Out[24]:
```
6
```

In [23]:
```
arr=np.array([[1,2,3],[4,5,6]])
print(arr.size)
print(arr.ndim)
[[1,2,3],
[4,5,6],
[7,8,9]]
```

```
6
2
```

In [26]:
```
#Using numpy zeros we are creating an array of the dimension(it will be passed as an argument).This will array whole items
ex2= np.zeros((3,4))
print(ex2)
```

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

In [27]:
```
ex2+5
```

Out[27]:
```
array([[5., 5., 5., 5.],
       [5., 5., 5., 5.],
       [5., 5., 5., 5.]])
```

In [28]:
```
ex2+10
```

Out[28]:
```
array([[10., 10., 10., 10.],
       [10., 10., 10., 10.],
       [10., 10., 10., 10.]])
```

In [32]:
```
ex1=np.array([ [2,3,5,7],[1,9,24,15],[5,12,19,21]  ])
print(ex1)
ex1.resize(4,3)
print(ex1)
```

```
[[ 2  3  5  7]
 [ 1  9 24 15]
 [ 5 12 19 21]]
[[ 2  3  5]
 [ 7  1  9]
 [24 15  5]
 [12 19 21]]
```

In [33]:
```
ex1+11 #11 will be added in all the items in an array
```

Out[33]:
```
array([[13, 14, 16],
       [18, 12, 20],
       [35, 26, 16],
       [23, 30, 32]])
```

In [35]:
```
twoDarray =np.array([[1,2,3],[4,5,6]])
print(twoDarray)
twoDarray.resize((3,2))
print(twoDarray)
```

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

In [41]:
```
#Reading the inpurt to create a matrix
print("Enter the number of rows : ")
rows=int(input())
print("Enter the number of cols : ")
cols=int(input())
arr=np.zeros((rows,cols))
print(arr)
#Read the data from standard I/O and add it into arr variable -ndarray
#print("Enter the data to be added in array ",str(rows)+"x"+str(cols))
for i in range(rows):
    for j in range(cols):
       #arr[i][j]=int(input())
       arr[i][j]=int(input("Enter the data to be added in array "+str(rows)+"x"+str(cols)+" "))
print(arr)
```

```
Enter the number of rows :
3
Enter the number of cols :
3
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
Enter the data to be added in array 3x3 1
Enter the data to be added in array 3x3 2
Enter the data to be added in array 3x3 3
Enter the data to be added in array 3x3 4
Enter the data to be added in array 3x3 5
Enter the data to be added in array 3x3 6
Enter the data to be added in array 3x3 7
Enter the data to be added in array 3x3 8
Enter the data to be added in array 3x3 9
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
```

In [45]:
```
#  0 1 2 3 4 5
x=[1,2,3,4,5,6]
arrX= np.array(x)
print(arrX)
```

```
print(arrX.size)
print(arrX[1])
print(arrX[4])
```

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

In [52]:
```
twoDData =[[1,2,3,4],[5,6,7,8]]
twoDarr =np.array(twoDData)
print(twoDarr)
print(twoDarr.size)
print(twoDarr[1][1])
print(twoDarr[1][3])
```

```
[[1 2 3 4]
 [5 6 7 8]]
8
6
8
```

In [53]:
```
ex1=np.array([ [2,3,5,7],[1,9,24,15],[5,12,19,21] ])
print(ex1)
```

```
[[ 2  3  5  7]
 [ 1  9 24 15]
 [ 5 12 19 21]]
```

In [54]:
```
'''
 (0,0) (0,1) (0,2) (0,3)
 [ 2    3      5    7]
 (1,0) (1,1) (1,2) (1,3)
 [ 1     9     24   15]
 (2,0) (2,1) (2,2) (2,3)
 [ 5     12    19   21]
'''
#3rd element in the 2nd row
print(ex1[1][2])
#first element in second row
print(ex1[1][0])
#third element in first row
print(ex1[0][2])
```

```
24
1
5
```

In [55]:
```
#column  wise
print(np.sum(ex1, axis=0))
#row wise
print(np.sum(ex1, axis=1))
```

```
[ 8 24 48 43]
[17 49 57]
```

In [56]:
```
ex1=ex1.ravel() #ravel  function is used to flattened the dimension of an array
print(ex1)
```

```
[ 2  3  5  7  1  9 24 15  5 12 19 21]
```

In [61]:
```
ex3 =np.array((1,2,4,5,6,7,8,9,10))
print("Sum : ",np.sum(ex3))
print("Min: ", np.min(ex3))
print("Max: ", np.max(ex3))
```

```
Sum :  52
Min:  1
Max:  10
```

In [66]:
```
#Slicing arrays
#ex3[start:end:step]  -- start is the index position then end index position not inclusive step -- by default
print(ex3)
print(ex3[2:7])
print(ex3[2:7:2])
print(ex3[2:9:2])
print(ex3[2::])
```

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

In [84]:
```
'''
In negative slicig-- arr[start:stop:step]
start -- will be the index from end of the array -1
stop -- end index position
step -- how element to skip while selecting the data
Negative slicing -- uses minus operator to refer to an index from the end
```

```
  8  7  6  5  4  3  2   1  0 -- index position
[ 1  2  4  5  6  7  8  9 10]
'''
print(ex3[-3:-1])
print(ex3[-7:-3])
#print(ex3[-8:-2:3])
print(ex3[-8:-2])
print(ex3[-8:-2:2])
print(ex3[-8:-2:3])
```

```
[8 9]
[4 5 6 7]
[2 4 5 6 7 8]
[2 5 7]
[2 6]
```

In [79]:

```
#Positive slice
'''
 arr[start:stop:step] start -- start index position stop end-1 position step to skip the number of element
  0  1  2  3  4  5  6  7  8
[ 1  2  4  5  6  7  8  9 10]
'''
ex3 =np.array((1,2,4,5,6,7,8,9,10))
print(ex3)
print(ex3[2:7])
print(ex3[2:8])
print(ex3[2:8:2])
```

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

In [85]:

```
ex4 = np.arange(2, 25, 2) # 2- starting number, 25 is end number ,2 -step
print(ex4)
```

```
[ 2  4  6  8 10 12 14 16 18 20 22 24]
```

In [89]:

```
ex5 = np.array([[1,2,3,4,5],[6,7,8,9,10]])
print(ex5)
print(ex5[1,1:4])
'''
     0  1  2  3  4
0 [[ 1  2  3  4  5]
1  [ 6  7  8  9 10]]
```

```
'''
print("************")
print(ex5[0:2,1]) #From both rows its returning data from col index position 1
print("******From both rows slice index 1 to index 4 (not included) this will return a 2D array******")
print(ex5[0:2,1:4])
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
[7 8 9]
************
[2 7]
******From both rows slice index 1 to index 4 (not included) this will return a 2D array******
[[2 3 4]
 [7 8 9]]
```

In [ ]: