Installation of Numpy library for python

```
pip install numpy    #pip command used to install
```

```
Note: you may need to restart the kernel to use updated packages.
```

```
ERROR: Invalid requirement: '#pip'
```

Importing the Numpy library in python

```
import numpy as np          #'import' used to load the library
print("numpy imported")
```

```
numpy imported
```

Lets check some working of Numpy on array

Array creation

```
ls = [25,14,36,96,85,74,41,52,636,545,55]
arr = np.array(ls)           # Using numpy to convert list into array
arr
```

```
array([ 25,  14,  36,  96,  85,  74,  41,  52, 636, 545,  55])
```

Type of array

```
type(arr)
#nd n dimensional array
```

```
numpy.ndarray
```

Creating a list

```
ls2 = [25,'upflairs',True,52.36]
ls2
```

```
[25, 'upflairs', True, 52.36]
```

Conversion of list into a array

```
## Array creation
ls = [25,14,36,96,85,74,41,52,636,545,55,100.25]      # A single data
type change in a list
```

```
arr = np.array(ls)      #can covert a arr of different integer
arr
```

```
array([ 25.  ,  14.  ,  36.  ,  96.  ,  85.  ,  74.  ,  41.  ,  52.  ,
       636.  , 545.  ,  55.  , 100.25])
```

```
## Array creation
ls = [25,14,36,96,85,74,41,52,636,545,55,'upflairs']        #giving a
string in a list
arr = np.array(ls)      # Whole array is converted into string data
type
arr
```

```
array(['25', '14', '36', '96', '85', '74', '41', '52', '636', '545',
'55',
       'upflairs'], dtype='<U11')
```

```
## Array creation
ls = [25,14,36,96,85,74,41,52,636,545,55,251]  ## reverse print
arr = np.array(ls)
arr[::-1]
```

```
array([251,  55, 545, 636,  52,  41,  74,  85,  96,  36,  14,  25])
```

Ways to count number of stored item/element in a array

```
len(arr)     # Number of elements in a array
```

```
12
```

```
arr.size   # or len(arr)
```

```
12
```

Data type of the given array

```
arr.dtype
```

```
dtype('int32')
```

Array slicing

```
arr[1:]
# positive and negative
```

```
array([ 14,  36,  96,  85,  74,  41,  52, 636, 545,  55, 251])
```

arange function

'numpy.arange' is a function in Python's NumPy library that returns an array of evenly spaced values within a specified range.

```
np.arange(1,60)    #Provide a array acc. to specified range
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
51,
       52, 53, 54, 55, 56, 57, 58, 59])
```

```
arr2= np.arange(60)
arr2
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50,
       51, 52, 53, 54, 55, 56, 57, 58, 59])
```

```
arr2= np.arange(0,101,2)    # range(starting, stoping , jump )
arr2         #adding range in array can change results in specified
range
```

```
array([  0,   2,   4,   6,   8,  10,  12,  14,  16,  18,  20,  22,
24,
        26,  28,  30,  32,  34,  36,  38,  40,  42,  44,  46,  48,
50,
        52,  54,  56,  58,  60,  62,  64,  66,  68,  70,  72,  74,
76,
        78,  80,  82,  84,  86,  88,  90,  92,  94,  96,  98, 100])
```

```
len(arr2)
```

```
51
```

function creates a new array filled with the given condition

```
np.ones(10)   #function in NumPy creates an array of shape (10,)
filled with ones.
```

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

```
np.zeros(10)    #function in NumPy creates an array of shape (10,)
filled with zeros.

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

np.linspace(1,50,20)   # in between 1 to  50 , item 20

array([ 1.        ,  3.57894737,  6.15789474,  8.73684211,
11.31578947,
       13.89473684, 16.47368421, 19.05263158, 21.63157895,
24.21052632,
       26.78947368, 29.36842105, 31.94736842, 34.52631579,
37.10526316,
       39.68421053, 42.26315789, 44.84210526, 47.42105263,
50.        ])
```

Create a array using random function

```
arr = np.random.randint(1,300,500)    # in between 1 - 300 items ==>
500
arr

array([ 86, 133,  77, 145,  88,  39, 104,  62, 294, 285, 248,  65,
84,
       197, 101, 233,  63,  39, 208, 293,  35, 241,  80, 266,  29,
211,
       189,  69,  67, 253,  69, 215, 294,  47, 157, 298, 122,   6,
32,
        52,  73, 278, 281, 138, 202, 288, 129, 222, 123, 128, 160,
129,
        10, 188, 224, 281, 176, 296, 243, 281, 175, 165, 278, 266,
198,
        42, 115, 215,  19, 218,  95, 133, 181, 221,  47, 286,  57,
135,
       286,  35, 108, 284, 214, 144, 284, 200, 171,  67, 291, 170,
145,
        30, 200, 167,   1, 111,  77, 224, 147,  34, 139, 289, 220,
51,
        12, 105,  82, 281, 199,  52, 292, 225, 295, 209,  47, 184,
248,
        78,  95,  22,  86, 231, 221, 261, 148, 180, 277, 116, 199,
98,
       198,  65, 152, 223,  58,  29, 164, 231, 107, 149, 267, 121,
209,
       225, 160, 127,  36, 136, 228, 193, 203, 205, 152,  84, 264,
174,
       150,  26, 167, 106,  31, 232, 230, 116,  63, 188,  92,  64,
228,
        75,  56, 272, 296,  51, 148, 131, 262, 106, 274, 242, 213,
249,
```

```
        77, 104,   4, 291, 210, 225, 245,  97, 280, 212, 214, 177,
156,
       132, 265, 283,  17, 154,  38,  88, 191,  56, 267, 298,  34,
291,
       135, 222,   2,  81,  54,  83,  57, 270, 123,  55, 133, 162,
250,
       137,  96,  91,  47, 214, 150, 245, 129, 224,  42, 238,  81,
20,
        17, 284,  30, 247,  74, 167, 192,   9, 170, 182, 154, 254,
71,
       230, 259,   4,  95, 197, 297, 155, 205,  43, 288, 143, 247,
164,
       128,  38, 297, 200, 172, 171, 260, 217,   8,  29, 250, 289,
239,
       132, 227,  67,  92, 162,  81,   5,  10, 154, 104,  59, 121,
136,
       209, 185, 208, 186,  22, 275, 182, 176, 218, 295, 239, 148,
30,
       174, 247, 127, 198, 129, 225,  17, 205,  80, 288, 254,  32,
116,
       140, 154,  88, 239,  13,   9,  18, 279, 180, 295, 164, 113,
35,
       206,   1, 269, 299,  93, 266,  86, 143, 280,  26,  63, 271,
59,
        64,  48, 120, 256, 239, 273, 281, 217,  24, 204, 275, 184,
246,
       178, 154, 206, 229,  22,  11,  87, 220,  53, 240, 112, 105,
267,
       126, 282, 252, 120,  98, 221, 282, 269, 187,  18, 271, 155,
256,
       169, 171,  81, 188,  94, 268, 140, 120,  39, 177, 262,  17,
248,
       234,  77, 227,  96,  90,  84, 199, 208,  55,  60, 102, 200,
278,
       136,  53, 271, 203,  53, 247,  30,  33, 259, 113,  22, 145,
273,
        21, 143, 182,  89, 133, 251,  94, 122, 229, 125, 138, 278,
22,
       216, 270, 191, 134, 281, 298, 167, 273,  54, 207, 187,  83,
159,
       159, 182, 280, 190,  34, 144, 128,   5, 191,  55, 158,  24,
156,
        63, 266,  74, 127,  83, 267, 183, 251,  91, 253, 248, 196,
162,
       108,  99, 264, 157, 131, 179, 196,  97, 191, 156,  92, 130,
293,
       107, 120, 138,  94,  29, 100, 209, 288,   1,  66, 105, 147,
150,
         9, 258, 179,  16, 228,  88])
```

```
arr.size
```

```
500
```

Quiz time : filter all items that are less than or equal too 70 and how many items are present in your array.

```
# Program using for loop and if-else condition

count = 0
for item in arr:
    if item <=70:
        print(item)
        count = count + 1

print("No of Item : ",count)
```

```
39
62
65
63
39
35
29
69
67
69
47
6
32
52
10
42
19
47
57
35
67
30
1
34
51
12
52
47
22
65
58
29
36
```

```
26
31
63
64
56
51
4
17
38
56
34
2
54
57
55
47
42
20
17
30
9
4
43
38
8
29
67
5
10
59
22
30
17
32
13
9
18
35
1
26
63
59
64
48
24
22
11
53
18
```

```
39
17
55
60
53
53
30
33
22
21
22
54
34
5
55
24
63
29
1
66
9
16
No of Item :  104
```

```
#Creates a new array with elements from arr that are less than or
equal to 70

arr[arr<=70]
```

```
array([39, 62, 65, 63, 39, 35, 29, 69, 67, 69, 47,  6, 32, 52, 10, 42,
19,
       47, 57, 35, 67, 30,  1, 34, 51, 12, 52, 47, 22, 65, 58, 29, 36,
26,
       31, 63, 64, 56, 51,  4, 17, 38, 56, 34,  2, 54, 57, 55, 47, 42,
20,
       17, 30,  9,  4, 43, 38,  8, 29, 67,  5, 10, 59, 22, 30, 17, 32,
13,
        9, 18, 35,  1, 26, 63, 59, 64, 48, 24, 22, 11, 53, 18, 39, 17,
55,
       60, 53, 53, 30, 33, 22, 21, 22, 54, 34,  5, 55, 24, 63, 29,  1,
66,
        9, 16])
```

```
arr[arr<=70].size      #just adding '.size' funtion can tell number
elements in a array
```

```
104
```

The 'ndim' attribute in NumPy provides the number of dimensions of an array.

```
arr.ndim
```

```
1
```

Creating a 2d array using 1d array

```
lst = [[1,2,3],[4,5,6],[7,8,9]]
type(lst)
```

```
list
```

```
arr = np.array(lst)
arr
```

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

```
arr.ndim
```

```
2
```

The 'arr.shape' attribute provides information about the shape of a NumPy array.

```
arr.shape
# (row,column)
```

```
(3, 3)
```

```
arr[2]  # row single
```

```
array([7, 8, 9])
```

```
arr[0:2]   # multiple row
```

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

```
arr[1:3,1:3]    #multiple rows and multiple columns simultaneously
```

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

```
arr2 = np.random.randint(1,200,(10,6))     #in between 1 - 200 range,
rows - 10, columns - 6
arr2
```

```
array([[ 50, 171,  47,  82, 107,  54],
       [ 99, 112, 154,  34, 191, 173],
       [ 91,  54,  33, 138,  52,  14],
       [160, 175, 158, 152, 116,  95],
       [154,  20, 100, 152, 144,  81],
```

```
       [ 91, 107, 135,  59,  78, 160],
       [144,  89,  69, 163,  79,   6],
       [180, 102,  54, 107, 194, 191],
       [185, 157, 103, 121, 116, 177],
       [138,  61,  94,  71, 183, 162]])
```

```
arr2[-6:-1,-3:]     # array slicing for rows and columns
```

```
array([[152, 144,  81],
       [ 59,  78, 160],
       [163,  79,   6],
       [107, 194, 191],
       [121, 116, 177]])
```

```
arr2[::-1]    #array[start:stop:step]
```

```
array([[138,  61,  94,  71, 183, 162],
       [185, 157, 103, 121, 116, 177],
       [180, 102,  54, 107, 194, 191],
       [144,  89,  69, 163,  79,   6],
       [ 91, 107, 135,  59,  78, 160],
       [154,  20, 100, 152, 144,  81],
       [160, 175, 158, 152, 116,  95],
       [ 91,  54,  33, 138,  52,  14],
       [ 99, 112, 154,  34, 191, 173],
       [ 50, 171,  47,  82, 107,  54]])
```

```
arr2
```

```
array([[ 50, 171,  47,  82, 107,  54],
       [ 99, 112, 154,  34, 191, 173],
       [ 91,  54,  33, 138,  52,  14],
       [160, 175, 158, 152, 116,  95],
       [154,  20, 100, 152, 144,  81],
       [ 91, 107, 135,  59,  78, 160],
       [144,  89,  69, 163,  79,   6],
       [180, 102,  54, 107, 194, 191],
       [185, 157, 103, 121, 116, 177],
       [138,  61,  94,  71, 183, 162]])
```

Changing element of a Array form a specific place

```
arr2[-1,-1]        #giving specific location of array we want change
```

```
162
```

```
arr2[-1,-1]  = 1000     # replacing value of array with we want to
change
```

```
arr2[-1,-1]        #Confirimg the replaced value
```

```
1000
```

```
arr2
```

```
array([[  50,  171,   47,   82,  107,   54],
       [  99,  112,  154,   34,  191,  173],
       [  91,   54,   33,  138,   52,   14],
       [ 160,  175,  158,  152,  116,   95],
       [ 154,   20,  100,  152,  144,   81],
       [  91,  107,  135,   59,   78,  160],
       [ 144,   89,   69,  163,   79,    6],
       [ 180,  102,   54,  107,  194,  191],
       [ 185,  157,  103,  121,  116,  177],
       [ 138,   61,   94,   71,  183, 1000]])
```

```
arr2[0,3]=500          # Another example of changing the array value
```

```
arr2
```

```
array([[  50,  171,   47,  500,  107,   54],
       [  99,  112,  154,   34,  191,  173],
       [  91,   54,   33,  138,   52,   14],
       [ 160,  175,  158,  152,  116,   95],
       [ 154,   20,  100,  152,  144,   81],
       [  91,  107,  135,   59,   78,  160],
       [ 144,   89,   69,  163,   79,    6],
       [ 180,  102,   54,  107,  194,  191],
       [ 185,  157,  103,  121,  116,  177],
       [ 138,   61,   94,   71,  183, 1000]])
```

```
arr2[-10,-3] =  500          #using negative indexing
```

```
arr2[0][3]  =  8000     #another method to denote row and columns
```

```
arr2
```

```
array([[  50,  171,   47, 8000,  107,   54],
       [  99,  112,  154,   34,  191,  173],
       [  91,   54,   33,  138,   52,   14],
       [ 160,  175,  158,  152,  116,   95],
       [ 154,   20,  100,  152,  144,   81],
       [  91,  107,  135,   59,   78,  160],
       [ 144,   89,   69,  163,   79,    6],
       [ 180,  102,   54,  107,  194,  191],
       [ 185,  157,  103,  121,  116,  177],
       [ 138,   61,   94,   71,  183, 1000]])
```

Creation a 3d array using 2d array

```
# 3D = [2d,2d,2d]
```

```
arr3d = np.random.randint(1,200,(5,10,5))       # in between 1-200 range, tables = 5, rows = 10, column = 5
arr3d
```

```
array([[[151,  40, 110,   1, 109],
        [ 12,  83,  27,  98, 118],
        [104, 148,  47, 187,  99],
        [119, 144,  62, 101,  96],
        [  9,  32,  27, 103, 188],
        [ 71,  77,  95,  73,  52],
        [119,  18,  55,  80,  17],
        [137,  57, 175, 186, 136],
        [134, 169,  22, 150,  38],
        [153, 193, 100, 150, 184]],

       [[ 57, 122,  63, 132,  67],
        [146, 145,  55,  78,  78],
        [ 41, 189, 158,  84, 177],
        [166,  84,  32, 190,  97],
        [ 93, 192, 113,  39,  69],
        [110, 187, 138,  90,   3],
        [ 65, 163, 180, 105,  10],
        [ 16,  12,  91, 165, 114],
        [195, 156, 147, 114, 194],
        [193,  38,  19, 164, 133]],

       [[ 66, 113,  56,  56, 153],
        [185,   7, 162,  11,  52],
        [117, 152,  40,   5,  48],
        [ 53,  47,  74,  78, 156],
        [102,  56, 113,  32,  20],
        [ 72, 148, 175,  93, 158],
        [153, 149, 181, 177,  70],
        [113,  93,  32,  43,  38],
        [ 55, 194,  16,  13,  64],
        [175,  81, 119, 180, 113]],

       [[161, 110,  88,  24,  32],
        [155, 154,  60,  67,  77],
        [ 13,  86,  81,  78,  91],
        [147, 199, 195, 188, 128],
        [ 67,  67, 180, 152,  50],
        [165,  82, 197,   8,  36],
        [ 53,  46, 169, 128, 197],
        [ 11,  30,  80, 150, 197],
        [ 83, 166, 144,  40,  69],
        [ 30,   7, 102,  50,  59]],

       [[ 95, 142, 110, 136,  68],
        [ 24,  82, 187, 199,  52],
```

```
        [113, 183, 114, 170, 197],
        [ 75,  97,  70,  34, 138],
        [  2,  85, 199, 162, 171],
        [102,  46,  70,  24, 104],
        [102, 114,  24,   9, 196],
        [143,  52, 175,  84, 147],
        [ 78,  10, 132,  77, 163],
        [111, 104,  29, 181, 147]]])

arr3d.shape
# (5, 10, 5)
# 5  ==>  Tables
# 10 ==>  Rows
# 5  ==>  columns

(5, 10, 5)

arr3d.ndim

3

arr3d[0]   # table

array([[151,  40, 110,   1, 109],
       [ 12,  83,  27,  98, 118],
       [104, 148,  47, 187,  99],
       [119, 144,  62, 101,  96],
       [  9,  32,  27, 103, 188],
       [ 71,  77,  95,  73,  52],
       [119,  18,  55,  80,  17],
       [137,  57, 175, 186, 136],
       [134, 169,  22, 150,  38],
       [153, 193, 100, 150, 184]])

arr3d

array([[[151,  40, 110,   1, 109],
        [ 12,  83,  27,  98, 118],
        [104, 148,  47, 187,  99],
        [119, 144,  62, 101,  96],
        [  9,  32,  27, 103, 188],
        [ 71,  77,  95,  73,  52],
        [119,  18,  55,  80,  17],
        [137,  57, 175, 186, 136],
        [134, 169,  22, 150,  38],
        [153, 193, 100, 150, 184]],

       [[ 57, 122,  63, 132,  67],
        [146, 145,  55,  78,  78],
        [ 41, 189, 158,  84, 177],
        [166,  84,  32, 190,  97],
```

```
        [ 93, 192, 113,  39,  69],
        [110, 187, 138,  90,   3],
        [ 65, 163, 180, 105,  10],
        [ 16,  12,  91, 165, 114],
        [195, 156, 147, 114, 194],
        [193,  38,  19, 164, 133]],

       [[ 66, 113,  56,  56, 153],
        [185,   7, 162,  11,  52],
        [117, 152,  40,   5,  48],
        [ 53,  47,  74,  78, 156],
        [102,  56, 113,  32,  20],
        [ 72, 148, 175,  93, 158],
        [153, 149, 181, 177,  70],
        [113,  93,  32,  43,  38],
        [ 55, 194,  16,  13,  64],
        [175,  81, 119, 180, 113]],

       [[161, 110,  88,  24,  32],
        [155, 154,  60,  67,  77],
        [ 13,  86,  81,  78,  91],
        [147, 199, 195, 188, 128],
        [ 67,  67, 180, 152,  50],
        [165,  82, 197,   8,  36],
        [ 53,  46, 169, 128, 197],
        [ 11,  30,  80, 150, 197],
        [ 83, 166, 144,  40,  69],
        [ 30,   7, 102,  50,  59]],

       [[ 95, 142, 110, 136,  68],
        [ 24,  82, 187, 199,  52],
        [113, 183, 114, 170, 197],
        [ 75,  97,  70,  34, 138],
        [  2,  85, 199, 162, 171],
        [102,  46,  70,  24, 104],
        [102, 114,  24,   9, 196],
        [143,  52, 175,  84, 147],
        [ 78,  10, 132,  77, 163],
        [111, 104,  29, 181, 147]]])

arr3d[-1,-1,-1]    # retreating value of array by defining specific
location

147

arr = np.arange(60)    #1-D
arr

array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
16,
```

```
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50,
       51, 52, 53, 54, 55, 56, 57, 58, 59])
```

Changing dimensions of array

```
arr2 = arr.reshape(20,3)  # 20 , 3    #Changed 1d array to 2d array
arr2
```

```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11],
       [12, 13, 14],
       [15, 16, 17],
       [18, 19, 20],
       [21, 22, 23],
       [24, 25, 26],
       [27, 28, 29],
       [30, 31, 32],
       [33, 34, 35],
       [36, 37, 38],
       [39, 40, 41],
       [42, 43, 44],
       [45, 46, 47],
       [48, 49, 50],
       [51, 52, 53],
       [54, 55, 56],
       [57, 58, 59]])
```

```
arr3 = arr.reshape(2,15,2)    # converting 1-d array into 2-d array
arr3
```

```
array([[[ 0,  1],
        [ 2,  3],
        [ 4,  5],
        [ 6,  7],
        [ 8,  9],
        [10, 11],
        [12, 13],
        [14, 15],
        [16, 17],
        [18, 19],
        [20, 21],
        [22, 23],
        [24, 25],
        [26, 27],
        [28, 29]],
```

```
       [[30, 31],
        [32, 33],
        [34, 35],
        [36, 37],
        [38, 39],
        [40, 41],
        [42, 43],
        [44, 45],
        [46, 47],
        [48, 49],
        [50, 51],
        [52, 53],
        [54, 55],
        [56, 57],
        [58, 59]]])
```

```
arr4 = arr3.reshape (10,6)      #converting 3d array into 2d
arr4
```

```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35],
       [36, 37, 38, 39, 40, 41],
       [42, 43, 44, 45, 46, 47],
       [48, 49, 50, 51, 52, 53],
       [54, 55, 56, 57, 58, 59]])
```

```
arr3.ravel()   #3d array conversion to 1d array
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50,
       51, 52, 53, 54, 55, 56, 57, 58, 59])
```

```
arr2.flatten()    #3d array conversion to 1d array
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50,
       51, 52, 53, 54, 55, 56, 57, 58, 59])
```

```
arr = np.random.randint(1,200,(5,3))     # in between 1-200, 5 rows
and 3 columns
arr
```

```
array([[160,   3,  93],
       [ 48, 135,  32],
       [133, 133,  92],
       [127, 159,  33],
       [ 38,   2,  88]])
```

```
arr.shape      #checking dimensions of array
```

```
(5, 3)
```

'transpose' is a function used to reverse or permute the axes of an array

```
np.transpose(arr)     #changed rows - 3 and columns - 5
```

```
array([[160,  48, 133, 127,  38],
       [  3, 135, 133, 159,   2],
       [ 93,  32,  92,  33,  88]])
```

```
np.transpose(arr).shape
```

```
(3, 5)
```

```
arr.T    # Using 'T' instead of 'transpose'
```

```
array([[160,  48, 133, 127,  38],
       [  3, 135, 133, 159,   2],
       [ 93,  32,  92,  33,  88]])
```

```
arr2= arr.ravel()
arr2
```

```
array([160,   3,  93,  48, 135,  32, 133, 133,  92, 127, 159,  33,
38,
         2,  88])
```

Some basic numerical use of Numpy library

```
arr2.mean()     #mean()    #max()  #min()
```

```
85.06666666666666
```

```
np.median(arr2)    #middle value
```

```
92.0
```

```
arr = np.array([9,7,8])
arr
```

```
array([9, 7, 8])

arr.argmin()    #"arg" stands for "argument" refers to the input value that a function takes.

1

arr.argmax()

0

arr

array([9, 7, 8])

arr.sort()  # ascending order

arr

array([7, 8, 9])

arr[::-1]   # descending order

array([9, 8, 7])

arr = np.array([9,7,8])
arr

array([9, 7, 8])

arr

array([9, 7, 8])

arr.argsort()  # [7,8,9]

array([1, 2, 0], dtype=int64)
```