Numpy Basic

In [2]:
```
# First of all we have to import the numpy

import numpy as np
```

In [3]:
```
# Then we take a list to make an array

l = [1,5,6,5,6]
```

In [4]:
```
# Then we convert into array with the use of array function

array = np.array(l)
```

In [5]:
```
array
```

Out[5]:
```
array([1, 5, 6, 5, 6])
```

In [6]:
```
# So check the the type of array we got..
# N dimensional array

type(array)
```

Out[6]:
```
numpy.ndarray
```

In [7]:
```
# Now we creating a 2D array

np.array([[1,2],[2,6]])
```

Out[7]:
```
array([[1, 2],
       [2, 6]])
```

In [8]:
```
# You can also use (asarray) and (asanyarray)to make array...

np.asarray(l)
```

Out[8]:
```
array([1, 5, 6, 5, 6])
```

In [9]:
```
np.asanyarray(l)
```

Out[9]:
```
array([1, 5, 6, 5, 6])
```

In [10]:
```
# Now we see the use of matrix
# When you use the matrix method your output always formed in 2D
# Matrix is a subset of a Array

b= np.matrix(l)
```

In [11]:
```
b
```

Out[11]:
```
matrix([[1, 5, 6, 5, 6]])
```

In [12]:
```
# Now b is the matrix function
# Whether we use b in array then output will be matrix because,
# Matrix is a subset of a Array
```

```
np.asanyarray(b)
```

Out[12]:
```
matrix([[1, 5, 6, 5, 6]])
```

In [13]:
```
a = np.array(l)
```

In [14]:
```
a
```

Out[14]:
```
array([1, 5, 6, 5, 6])
```

In [15]:
```
# Now we see that what  will we output of c=a
c=a
```

In [16]:
```
c
```

Out[16]:
```
array([1, 5, 6, 5, 6])
```

In [17]:
```
a
```

Out[17]:
```
array([1, 5, 6, 5, 6])
```

In [18]:
```
c[0] = 100
```

In [19]:
```
c
```

Out[19]:
```
array([100,   5,   6,   5,   6])
```

In [20]:
```
# If we can change the value of c then its also change in value of a also...
a
```

Out[20]:
```
array([100,   5,   6,   5,   6])
```

In [21]:
```
# But in a copy function the output will be different...
d = np.copy(a)
```

In [22]:
```
a
```

Out[22]:
```
array([100,   5,   6,   5,   6])
```

In [23]:
```
d
```

Out[23]:
```
array([100,   5,   6,   5,   6])
```

In [24]:
```
a[1] = 400
```

In [25]:
```
a
```

Out[25]:
```
array([100, 400,   6,   5,   6])
```

In [26]:
```
# You see that the value of a has changed
# But the value of d not changed

d
```

Out[26]:
```
array([100,   5,   6,   5,   6])
```

In [27]:
```
# Now we crate the 3D array


np.fromfunction(lambda i,j : i==j , (3,3))
```

Out[27]:
```
array([[ True, False, False],
       [False,  True, False],
       [False, False,  True]])
```

In [28]:
```
np.fromfunction(lambda i,j : i*j , (3,3))
```

Out[28]:
```
array([[0., 0., 0.],
       [0., 1., 2.],
       [0., 2., 4.]])
```

In [29]:
```
# You can also create the array by using iterable function...

iterable = (i*i for i in range(5))
np.fromiter(iterable , float)
```

Out[29]:
```
array([ 0.,  1.,  4.,  9., 16.])
```

In [30]:
```
np.fromstring('235 235' ,sep = ' ')
```

Out[30]:
```
array([235., 235.])
```

In [31]:
```
np.fromstring('4,5' ,sep = ',')
```

Out[31]:
```
array([4., 5.])
```

In [32]:
```
# Now we can see the Data types of numpy
```

In [33]:
```
l = [1,2,3,5,2,5,2]
```

In [34]:
```
array = np.array(l)
```

In [35]:
```
array
```

Out[35]:
```
array([1, 2, 3, 5, 2, 5, 2])
```

In [36]:
```
# This action shows the which dimensional we are creating an array
array.ndim
```

Out[36]:
```
1
```

In [37]:
```
array2 = np.array([[1,5,6,5] , [1,5,8,2]])
array2
```

Out[37]:
```
array([[1, 5, 6, 5],
       [1, 5, 8, 2]])
```

In [38]:
```
array2.ndim
```

Out[38]:
```
2
```

In [39]:
```
array.size
```

Out[39]:
```
7
```

In [40]:
```
array2
```

Out[40]:
```
array([[1, 5, 6, 5],
       [1, 5, 8, 2]])
```

In [41]:
```
array.shape
```

Out[41]:
```
(7,)
```

In [42]:
```
# (2,4) this indicates the that we have two rows and 4 columns
array2.shape
```

Out[42]:
```
(2, 4)
```

In [43]:
```
array.dtype
```

Out[43]:
```
dtype('int32')
```

In [44]:
```
array2.dtype
```

Out[44]:
```
dtype('int32')
```

In [45]:
```
array3 = np.array([(10.4,5,6) , (56,25,66)])
```

In [46]:
```
array3
```

Out[46]:
```
array([[10.4,  5. ,  6. ],
       [56. , 25. , 66. ]])
```

In [47]:
```
array3.dtype
```

Out[47]:
```
dtype('float64')
```

In [48]:
```
# Suppose we need 5 int number in the range() function..
list(range(5))
```

Out[48]:
```
[0, 1, 2, 3, 4]
```

In [49]:
```
# If we put a float number in range function then its output will be error!
list(range(1.5,6))
# range function dosen't consider a float function
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[49], line 2
      1 # If we put a float number in range function then its output will be erro
r!
----> 2 list(range(1.5,6))

TypeError: 'float' object cannot be interpreted as an integer
```

In [50]:
```
# While in numpy there is a provision to give a float number..

np.arange(2.3,6.6)
```

Out[50]:
```
array([2.3, 3.3, 4.3, 5.3, 6.3])
```

In [51]:
```
# you can also jump the numbers
np.arange(1.1,5.1,.3)
```

Out[51]:
```
array([1.1, 1.4, 1.7, 2. , 2.3, 2.6, 2.9, 3.2, 3.5, 3.8, 4.1, 4.4, 4.7,
       5. ])
```

In [52]:
```
list(np.arange(1.1,5.1,.3))
```

Out[52]:
```
[1.1,
 1.4000000000000001,
 1.7000000000000002,
 2.0,
 2.3000000000000003,
 2.6000000000000005,
 2.9000000000000004,
 3.2000000000000006,
 3.5000000000000004,
 3.8000000000000003,
 4.1000000000000005,
 4.4,
 4.700000000000001,
 5.0]
```

In [53]:
```
# Now we are using linspace
# This method gives the between number which you have given like..

np.linspace(1,5,10)

# we need number between 1 to 5 and range is 10
```

Out[53]:
```
array([1.        , 1.44444444, 1.88888889, 2.33333333, 2.77777778,
       3.22222222, 3.66666667, 4.11111111, 4.55555556, 5.        ])
```

In [54]:
```
np.zeros(5)
```

Out[54]:
```
array([0., 0., 0., 0., 0.])
```

In [55]:
```
np.zeros((3,4))
```

Out[55]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [56]:
```
# this will be 3D array method
# means we need 3 matrix with 4 row and 2 column

np.zeros((3,4,2))
```

Out[56]:
```
array([[[0., 0.],
        [0., 0.],
        [0., 0.],
        [0., 0.]],

       [[0., 0.],
        [0., 0.],
        [0., 0.],
        [0., 0.]],

       [[0., 0.],
        [0., 0.],
        [0., 0.],
        [0., 0.]]])
```

In [57]:
```
# You can also create the 4D array also ...

np.zeros((3,4,2,3))
```

Out[57]:
```
array([[[[0., 0., 0.],
         [0., 0., 0.]],

        [[0., 0., 0.],
         [0., 0., 0.]],

        [[0., 0., 0.],
         [0., 0., 0.]],

        [[0., 0., 0.],
         [0., 0., 0.]]],


       [[[0., 0., 0.],
         [0., 0., 0.]],

        [[0., 0., 0.],
         [0., 0., 0.]],

        [[0., 0., 0.],
         [0., 0., 0.]],

        [[0., 0., 0.],
         [0., 0., 0.]]],


       [[[0., 0., 0.],
         [0., 0., 0.]],

        [[0., 0., 0.],
         [0., 0., 0.]],

        [[0., 0., 0.],
         [0., 0., 0.]],

        [[0., 0., 0.],
         [0., 0., 0.]]]])
```

In [59]:
```
np.ones(5)
```

Out[59]:
```
array([1., 1., 1., 1., 1.])
```

In [61]:
```
np.ones((2,3))
```

Out[61]:
```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [63]:
```
on = np.ones((2,3,2))
```

In [65]:
```
on + 45
```

Out[65]:
```
array([[[46., 46.],
        [46., 46.],
        [46., 46.]],

       [[46., 46.],
        [46., 46.],
        [46., 46.]]])
```

In [66]:
```
# Empty function

np.empty((3,6))
```

Out[66]:
```
array([[0., 0., 0., 1., 1., 1.],
       [2., 2., 2., 0., 1., 2.],
       [0., 1., 2., 0., 1., 2.]])
```

In [69]:
```
# eye function shows the identity matrix

np.eye(4)
```

Out[69]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

In [70]:
```
np.linspace(2,6  ,10)
```

Out[70]:
```
array([2.        , 2.44444444, 2.88888889, 3.33333333, 3.77777778,
       4.22222222, 4.66666667, 5.11111111, 5.55555556, 6.        ])
```

In [72]:
```
# The main diferrence between linspace and logspace is..
# Linspace is giving a 10 number between 2,6
# While in logspace is also giving 10 number between 2,6 but with log value..

np.logspace(2,6  ,10 , base=2)
```

Out[72]:
```
array([ 4.        ,  5.44316   ,  7.4069977 , 10.0793684 , 13.71590373,
       18.66446463, 25.39841683, 34.56191164, 47.03150375, 64.        ])
```

In [73]:
```
# Now we see the some random function with the use of pandas

arr = np.random.randn(3,4)
```

In [74]:
```
arr
```

Out[74]:
```
array([[-1.31767867, -0.13338812,  1.80281132,  0.69971606],
       [ 0.15420013, -0.1572266 , -2.20678711,  0.69609453],
       [ 0.04974214,  0.7949654 ,  0.27020281, -0.04773799]])
```

In [75]:
```
import pandas as pd
```

In [79]:
```
pd.DataFrame(arr)
```

Out[79]:

	0	1	2	3
0	-1.317679	-0.133388	1.802811	0.699716
1	0.154200	-0.157227	-2.206787	0.696095
2	0.049742	0.794965	0.270203	-0.047738

In [81]:
```
np.random.rand(3,4)
```

Out[81]:
```
array([[0.24163734, 0.33380599, 0.67961155, 0.23090276],
       [0.40460097, 0.10367557, 0.88153756, 0.3312323 ],
       [0.01124106, 0.96064823, 0.6943064 , 0.78620754]])
```

In [82]:
```
np.random.randint(1,100 , (5,6))
```

Out[82]:
```
array([[ 4, 19, 36, 75, 57, 14],
       [47, 92, 30, 22, 94, 60],
       [86, 91, 67, 73, 57, 78],
       [16, 88, 10, 68, 38, 54],
       [70,  1, 56, 25, 41, 40]])
```

In [83]:
```
pd.DataFrame(np.random.randint(1,100 , (5,6)))
```

Out[83]:

	0	1	2	3	4	5
0	42	57	65	70	75	72
1	49	65	38	58	64	26
2	32	72	34	73	68	82
3	22	36	16	46	84	55
4	8	33	7	71	10	61

In [85]:
```
(pd.DataFrame(np.random.randint(1,100 , (5,6)))).to_csv("test.csv")
```

In [86]:
```
arr1 = np.random.rand(6,5)
```

In [90]:
```
arr1.reshape(15,2)
```

Out[90]:
```
array([[0.77159335, 0.75336031],
       [0.86616107, 0.72231033],
       [0.18096834, 0.01199802],
       [0.31917119, 0.60682305],
       [0.42009555, 0.70004061],
       [0.53041893, 0.59991337],
       [0.73561265, 0.44209428],
       [0.29651462, 0.24094038],
       [0.23470886, 0.96512621],
       [0.33499079, 0.02412723],
       [0.8519577 , 0.21377044],
       [0.04726122, 0.64714242],
       [0.64472957, 0.71288828],
       [0.46324458, 0.70864864],
       [0.75642862, 0.07575875]])
```

In [95]:
```
arr1[0][0]
```

Out[95]:
```
0.7715933474409167
```

In [99]:
```
arr1[2:5 ,1]
```

Out[99]:
```
array([0.59991337, 0.23470886, 0.21377044])
```

In [101…
```
arr2 = np.random.randint(1,100 , (5,5))
```

In [102…
```
arr2
```

Out[102]:
```
array([[79, 84, 48, 18, 16],
       [82, 64, 76, 74, 14],
       [59, 70, 54, 52, 83],
       [61, 80, 99, 81, 93],
       [59, 40, 37, 63, 59]])
```

In [104…
```
arr2>50
```

Out[104]:
```
array([[ True,  True, False, False, False],
       [ True,  True,  True,  True, False],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True, False, False,  True,  True]])
```

In [105…
```
arr2[arr2>50]
```

Out[105]:
```
array([79, 84, 82, 64, 76, 74, 59, 70, 54, 52, 83, 61, 80, 99, 81, 93, 59,
       63, 59])
```

In [108…
```
arr2[2:4 , [1,2]]
```

Out[108]:
```
array([[70, 54],
       [80, 99]])
```

In [109…
```
arr2[0][0] = 5000
```

In [110…
```
arr2
```

Out[110]:
```
array([[5000,   84,   48,   18,   16],
       [  82,   64,   76,   74,   14],
       [  59,   70,   54,   52,   83],
       [  61,   80,   99,   81,   93],
       [  59,   40,   37,   63,   59]])
```

In [112…
```
arr3 = np.random.randint(1,3 , (3,3))
arr4 = np.random.randint(1,3 , (3,3))
```

In [113…
```
arr3
```

Out[113]:
```
array([[1, 1, 1],
       [2, 2, 2],
       [1, 2, 2]])
```

In [115…
```
arr4
```

Out[115]:
```
array([[1, 1, 2],
       [1, 2, 2],
       [2, 2, 1]])
```

In [116…
```
arr3+arr4
```

Out[116]:
```
array([[2, 2, 3],
       [3, 4, 4],
       [3, 4, 3]])
```

In [117…
```
arr3*arr4
```

Out[117]:
```
array([[1, 1, 2],
       [2, 4, 4],
       [2, 4, 2]])
```

In [119…
```
# for matrix multiplication

arr3@arr4
```

Out[119]:
```
array([[ 4,  5,  5],
       [ 8, 10, 10],
       [ 7,  9,  8]])
```

In [120…
```
#numpy - Broadcasting
```

In [125…
```
arr5 = np.zeros((4,4))
```

In [126…
```
arr5
```

Out[126]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [127…
```
row = np.array([1,2,3,45])
```

In [128…
```
row
```

Out[128]:
```
array([ 1,  2,  3, 45])
```

In [130…
```
arr5+row
```

Out[130]:
```
array([[ 1.,  2.,  3., 45.],
       [ 1.,  2.,  3., 45.],
       [ 1.,  2.,  3., 45.],
       [ 1.,  2.,  3., 45.]])
```

In [133…
```
col = np.array([[1,2,3,45]])
```

In [135…
```
col.T
```

Out[135]:
```
array([[ 1],
       [ 2],
       [ 3],
       [45]])
```

In [137…
```
arr5+col.T
```

Out[137]:
```
array([[ 1.,  1.,  1.,  1.],
       [ 2.,  2.,  2.,  2.],
       [ 3.,  3.,  3.,  3.],
       [45., 45., 45., 45.]])
```

In [ ]:
```

```

In [1]:
```
import numpy as np
```

In [2]:
```
# Numpy - Array manipulation
```

In [5]:
```
arr = np.random.randint(1,10 ,(3,4))
```

In [6]:
```
arr
```

Out[6]:
```
array([[5, 1, 6, 3],
       [5, 5, 5, 5],
       [6, 2, 8, 4]])
```

In [7]:
```
arr.reshape(2,6)
```

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

In [8]:
```
arr.reshape(6,2)
```

Out[8]:
```
array([[5, 1],
       [6, 3],
       [5, 5],
       [5, 5],
       [6, 2],
       [8, 4]])
```

In [9]:
```
arr.reshape(6,-232652294595)
```

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

In [10]:
```
arr.T
```

Out[10]:
```
array([[5, 5, 6],
       [1, 5, 2],
       [6, 5, 8],
       [3, 5, 4]])
```

In [12]:
```
arr.flatten()

# Flatten is use to show data in one dimensions
```

Out[12]:
```
array([5, 1, 6, 3, 5, 5, 5, 5, 6, 2, 8, 4])
```

In [13]:
```
arr1 = np.array([1,2,3,6,5])
```

In [14]:
```
arr1
```

Out[14]:
```
array([1, 2, 3, 6, 5])
```

In [15]:
```
arr1.ndim
```

Out[15]:
```
1
```

In [17]:
```
np.expand_dims(arr1 , axis=1)
```

```
# axis = 1 shows the colunm
# asis = 0 shows the rows
```

Out[17]:
```
array([[1],
       [2],
       [3],
       [6],
       [5]])
```

In [18]:
```
arr
```

Out[18]:
```
array([[5, 1, 6, 3],
       [5, 5, 5, 5],
       [6, 2, 8, 4]])
```

In [20]:
```
data = np.array([[1],[2],[3]])
```

In [21]:
```
data
```

Out[21]:
```
array([[1],
       [2],
       [3]])
```

In [22]:
```
data.squeeze()
```

Out[22]:
```
array([1, 2, 3])
```

In [23]:
```
arr1
```

Out[23]:
```
array([1, 2, 3, 6, 5])
```

In [27]:
```
arr1.repeat(3)
```

Out[27]:
```
array([1, 1, 1, 2, 2, 2, 3, 3, 3, 6, 6, 6, 5, 5, 5])
```

In [29]:
```
arr1
```

Out[29]:
```
array([1, 2, 3, 6, 5])
```

In [31]:
```
np.roll(arr1 , 2)

# roll will help the move numbers
```

Out[31]:
```
array([6, 5, 1, 2, 3])
```

In [32]:
```
np.diag(arr1)

# diag shows the number present in diagonal matrix
```

Out[32]:
```
array([[1, 0, 0, 0, 0],
       [0, 2, 0, 0, 0],
       [0, 0, 3, 0, 0],
       [0, 0, 0, 6, 0],
       [0, 0, 0, 0, 5]])
```

In [33]:
```
# Numpy - Binary operators
```

In [35]:
```
arr2 = np.random.randint(1,12 , (3,4))
arr3 = np.random.randint(1,12 , (3,4))
```

In [36]:
```
arr2
```

Out[36]:
```
array([[ 5,  2, 11,  9],
       [ 4, 10,  7,  8],
       [11,  5,  7,  1]])
```

In [37]:
```
arr3
```

Out[37]:
```
array([[11,  8,  3,  9],
       [ 9,  8,  8,  7],
       [ 8,  9,  6,  2]])
```

In [38]:
```
arr2+arr3
```

Out[38]:
```
array([[16, 10, 14, 18],
       [13, 18, 15, 15],
       [19, 14, 13,  3]])
```

In [39]:
```
arr2/arr3
```

Out[39]:
```
array([[0.45454545, 0.25      , 3.66666667, 1.        ],
       [0.44444444, 1.25      , 0.875     , 1.14285714],
       [1.375     , 0.55555556, 1.16666667, 0.5       ]])
```

In [40]:
```
arr2*arr3
```

Out[40]:
```
array([[55, 16, 33, 81],
       [36, 80, 56, 56],
       [88, 45, 42,  2]])
```

In [41]:
```
arr2-arr3
```

Out[41]:
```
array([[-6, -6,  8,  0],
       [-5,  2, -1,  1],
       [ 3, -4,  1, -1]])
```

In [44]:
```
arr2%arr3
```

Out[44]:
```
array([[5, 2, 2, 0],
       [4, 2, 7, 1],
       [3, 5, 1, 1]])
```

In [45]:
```
arr2 ** arr3
```

Out[45]:
```
array([[ 48828125,       256,      1331, 387420489],
       [   262144, 100000000,   5764801,   2097152],
       [214358881,   1953125,    117649,         1]])
```

In [46]:
```
arr2&arr3
```

Out[46]:
```
array([[1, 0, 3, 9],
       [0, 8, 0, 0],
       [8, 1, 6, 0]])
```

In [48]:
```
-arr2
```

Out[48]:
```
array([[ -5,  -2, -11,  -9],
       [ -4, -10,  -7,  -8],
       [-11,  -5,  -7,  -1]])
```

In [49]:
```
# numpy - String function
```

In [50]:
```
str = np.array(['Abhi' , 'mishra'])
```

In [51]:
```
str
```

Out[51]:
```
array(['Abhi', 'mishra'], dtype='<U6')
```

In [52]:
```
np.char.upper(str)
```

Out[52]:
```
array(['ABHI', 'MISHRA'], dtype='<U6')
```

In [53]:
```
np.char.capitalize(str)
```

Out[53]:
```
array(['Abhi', 'Mishra'], dtype='<U6')
```

In [54]:
```
np.char.title(str)
```

Out[54]:
```
array(['Abhi', 'Mishra'], dtype='<U6')
```

In [56]:
```
# numpy - mathematical functions
```

In [59]:
```
arr2
```

Out[59]:
```
array([[ 5,  2, 11,  9],
       [ 4, 10,  7,  8],
       [11,  5,  7,  1]])
```

In [60]:
```
np.sin(arr1)
```

Out[60]:
```
array([ 0.84147098,  0.90929743,  0.14112001, -0.2794155 , -0.95892427])
```

In [62]:
```
np.cos(arr2)
```

Out[62]:
```
array([[ 0.28366219, -0.41614684,  0.0044257 , -0.91113026],
       [-0.65364362, -0.83907153,  0.75390225, -0.14550003],
       [ 0.0044257 ,  0.28366219,  0.75390225,  0.54030231]])
```

In [63]:
```
np.tan(arr2)
```

Out[63]:
```
array([[  -3.38051501,   -2.18503986, -225.95084645,   -0.45231566],
       [   1.15782128,    0.64836083,    0.87144798,   -6.79971146],
       [-225.95084645,   -3.38051501,    0.87144798,    1.55740772]])
```

In [64]:
```
np.exp(arr2)
```

Out[64]:
```
array([[1.48413159e+02, 7.38905610e+00, 5.98741417e+04, 8.10308393e+03],
       [5.45981500e+01, 2.20264658e+04, 1.09663316e+03, 2.98095799e+03],
       [5.98741417e+04, 1.48413159e+02, 1.09663316e+03, 2.71828183e+00]])
```

In [65]:
```
np.sqrt(arr2)
```

Out[65]:
```
array([[2.23606798, 1.41421356, 3.31662479, 3.        ],
       [2.        , 3.16227766, 2.64575131, 2.82842712],
       [3.31662479, 2.23606798, 2.64575131, 1.        ]])
```

In [66]:
```
np.power(arr2 , 2)
```

Out[66]:
```
array([[ 25,   4, 121,  81],
       [ 16, 100,  49,  64],
       [121,  25,  49,   1]], dtype=int32)
```

In [67]:
```
np.mean(arr2)
```

Out[67]:
```
6.666666666666667
```

In [68]:
```
np.median(arr2)
```

Out[68]:
```
7.0
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```
# sorting ,search & counting function
```

In [81]:
```
arr4 = np.array([2,4,34,5,4,465,8,45,21])
```

In [82]:
```
arr4
```

Out[82]:
```
array([  2,   4,  34,   5,   4, 465,   8,  45,  21])
```

In [84]:
```
np.sort(arr4)
```

Out[84]:
```
array([  2,   4,   4,   5,   8,  21,  34,  45, 465])
```

In [85]:
```
np.searchsorted(arr4 , 34)

# In this function 34 value is sort in ascending order for 7th place
```

Out[85]:
```
7
```

In [86]:
```
arr5 = np.array([1,2,0,0,0,25])
```

In [89]:
```
np.count_nonzero(arr5)
```

Out[89]:
```
3
```

In [90]:
```
arr4
```

Out[90]:
```
array([  2,   4,  34,   5,   4, 465,   8,  45,  21])
```

In [91]:
```
np.where(arr4>10)
```

Out[91]:
```
(array([2, 5, 7, 8], dtype=int64),)
```

In [93]:
```
np.extract(arr4>3 , arr4)
```

Out[93]:
```
array([  4,  34,   5,   4, 465,   8,  45,  21])
```

In [ ]:

In [94]:
```
#numpy - Byte swapping
```

In [95]:
```
arr4.byteswap()
```

Out[95]:
```
array([  33554432,   67108864,  570425344,   83886080,   67108864,
       -788463616,  134217728,  754974720,  352321536])
```

In [96]:
```
#numpy - copies&views
```

In [97]:
```
a = np.copy(arr4)
```

In [98]:
```
b = arr4.view()
```

In [99]:
```
a
```

Out[99]:
```
array([  2,   4,  34,   5,   4, 465,   8,  45,  21])
```

In [100…
```
b
```

Out[100]:
```
array([  2,   4,  34,   5,   4, 465,   8,  45,  21])
```

In [101…
```
b[0] = 56
```

In [102…
```
b
```

Out[102]:
```
array([ 56,   4,  34,   5,   4, 465,   8,  45,  21])
```

In [104…
```
arr4
```

Out[104]:
```
array([ 56,   4,  34,   5,   4, 465,   8,  45,  21])
```

In [105…
```
# numpy - matrix library
```

In [106…
```
import numpy.matlib as nm
```

In [107…
```
nm.zeros(5)
```

Out[107]:
```
matrix([[0., 0., 0., 0., 0.]])
```

In [108…
```
nm.ones([3,4])
```

Out[108]:
```
matrix([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
```

In [113…
```
nm.eye(5)
```

Out[113]:
```
matrix([[1., 0., 0., 0., 0.],
        [0., 1., 0., 0., 0.],
        [0., 0., 1., 0., 0.],
        [0., 0., 0., 1., 0.],
        [0., 0., 0., 0., 1.]])
```

In [114…
```
# numpy - linear algebra
```

In [118…
```
arr5 = np.random.randint([[1,12] , [3,4]])
arr6 = np.random.randint([[1,12] , [3,4]])
```

In [119…
```
np.dot(arr5 , arr6)
```

Out[119]:
```
array([[ 2,  2],
       [ 4, 24]])
```

In [120…
```
arr5@arr6
```

Out[120]:
```
array([[ 2,  2],
       [ 4, 24]])
```

In [ ]:
```

```