Basic functions

In [1]:
```
!pip install numpy
```

```
Requirement already satisfied: numpy in d:\python\anaconda\lib\site-packages (1.21.5)
```

In [2]:
```
import numpy as np
```

In [3]:
```
arr1 = np.array([3,2,1,4])
print(arr1)
print(type(arr1))
```

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

In [4]:
```
arr1.shape              # order of the matrix
```

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

In [5]:
```
arr1.dtype              # type of the data type
```

Out[5]:
```
dtype('int32')
```

In [6]:
```
np.zeros((4,6))         # zero matrix should be a tuple.
```

Out[6]:
```
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 [7]:
```
np.ones(6)
```

Out[7]:
```
array([1., 1., 1., 1., 1., 1.])
```

In [8]:
```
np.empty((4,6))
```

Out[8]:
```
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 [9]:
```
arr1
```

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

In [10]:
```
arr1*arr1
```

Out[10]:
```
array([ 9,  4,  1, 16])
```

In [11]:
```
1/arr1
```

Out[11]:
```
array([0.33333333, 0.5       , 1.        , 0.25      ])
```

Slicing

In [12]:
```
arr1
```

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

In [13]:
```
arr1[3:1]
```

Out[13]:
```
array([], dtype=int32)
```

In [14]:
```
ar1 = np.array(([1,2,3],
                [4,5,6]))
ar1.sum(axis=1)            # sum of numbers in row wise.
```

Out[14]:
```
array([ 6, 15])
```

In [15]:
```
ar1 = np.array(([1,2,3],
                [4,5,6]))
ar1.sum(axis=0)             # sum of numbers in column wise.
```

Out[15]:
```
array([5, 7, 9])
```

In [22]:
```
b = np.array([[7,8,9],
              [2,8,0]])
np.dot(ar1, b.transpose())
```

Out[22]:
```
array([[ 50,  18],
       [122,  48]])
```

In [20]:
```
dir(np)
```

Out[20]:
```
['ALLOW_THREADS',
 'AxisError',
 'BUFSIZE',
 'Bytes0',
 'CLIP',
 'ComplexWarning',
 'DataSource',
 'Datetime64',
 'ERR_CALL',
 'ERR_DEFAULT',
 'ERR_IGNORE',
 'ERR_LOG',
 'ERR_PRINT',
 'ERR_RAISE',
 'ERR_WARN',
 'FLOATING_POINT_SUPPORT',
 'FPE_DIVIDEBYZERO',
 'FPE_INVALID',
 'FPE_OVERFLOW',
```

In [21]:
```
len(dir(np))
```

Out[21]:
```
608
```

In [23]:
```
np.cross(ar1, b)          # cross function outputs rows x columns.
```

Out[23]:
```
array([[ -6,  12,  -6],
       [-48,  12,  22]])
```

In [25]:
```
a = np.arange(6)        # arange function gives values of the defined range.
a
```

Out[25]:
```
array([0, 1, 2, 3, 4, 5])
```

In [32]:
```
b = np.array([1,3,27,43,55,66,4,3,7,8,9,10,23,34])
b = b.reshape(2,7)                                  # reshape is used to create ordered matrix of required size
b
```

Out[32]:
```
array([[ 1,  3, 27, 43, 55, 66,  4],
       [ 3,  7,  8,  9, 10, 23, 34]])
```

In [36]:
```
c = np.array([1,3,27,43,55,66,4,3,7,8,9,10,23,34])
c
```

Out[36]:
```
array([ 1,  3, 27, 43, 55, 66,  4,  3,  7,  8,  9, 10, 23, 34])
```

In [37]:
```
np.argsort(c)                  # argsort function speaks of sorting in ordererd way with position of each eleme
```

Out[37]:
```
array([ 0,  1,  7,  6,  8,  9, 10, 11, 12,  2, 13,  3,  4,  5],
      dtype=int64)
```

In [38]:
```
np.argmin(c)              # it gives the position of the minimum element present in the array, which is 1.
```

Out[38]:
```
0
```

In [39]:
```
np.argmax(c)              # position of the maximum element present in the array, ie., 66.
```

Out[39]:
```
5
```

In [42]:
```
c = c.reshape(2,7)
c
```

Out[42]:
```
array([[ 1,  3, 27, 43, 55, 66,  4],
       [ 3,  7,  8,  9, 10, 23, 34]])
```

In [43]:
```
c = np.argsort(c, axis=0)
c
```

Out[43]:
```
array([[0, 0, 1, 1, 1, 1, 0],
       [1, 1, 0, 0, 0, 0, 1]], dtype=int64)
```

In [44]:
```
c = np.argsort(c, axis=1)
c
```

Out[44]:
```
array([[0, 1, 6, 2, 3, 4, 5],
       [2, 3, 4, 5, 0, 1, 6]], dtype=int64)
```

In [49]:
```
c = np.array([1,3,27,43,55,66,4,3,7,8,9,10,23,34])
np.sort(c, axis=0)
c
```

Out[49]:
```
array([ 1,  3, 27, 43, 55, 66,  4,  3,  7,  8,  9, 10, 23, 34])
```

In [ ]: