Numpy Tutorial

In [1]:

```
import numpy as np
```

In [2]:

```
myarr = np.array([3,6,32,7],np.int64)
```

In [3]:

```
myarr
```

Out[3]:

```
array([ 3,  6, 32,  7])
```

In [4]:

```
myarr[0]
```

Out[4]:

```
3
```

In [5]:

```
myarr = np.array([[3,6,32,7]],np.int64)
```

In [6]:

```
myarr[0,1]
```

Out[6]:

```
6
```

In [7]:

```
myarr.shape
```

Out[7]:

```
(1, 4)
```

In [8]:

```
myarr.dtype
```

Out[8]:

```
dtype('int64')
```

In [9]:

```
myarr[0,1] = 45
```

In [10]:

```
myarr
```

Out[10]:

```
array([[ 3, 45, 32,  7]])
```

Array Creation : Conversion from other Python Structures

In [11]:

```
  listarray = np.array([[1,2,3],[5,8,5],[0,3,1]])
```

In [12]:

```
 listarray
```

Out[12]:

```
array([[1, 2, 3],
       [5, 8, 5],
       [0, 3, 1]])
```

In [13]:

```
listarray.dtype
```

Out[13]:

```
dtype('int64')
```

In [14]:

```
listarray.shape
```

Out[14]:

```
(3, 3)
```

In [15]:

```
listarray.size
```

Out[15]:

```
9
```

In [16]:

```
zeros = np.zeros((2,5))
```

In [17]:

```
zeros
```

Out[17]:

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

In [18]:

```
rng = np.arange(15)
```

In [19]:

```
rng
```

Out[19]:

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

In [20]:

```
lin_space = np.linspace(1,5,12)
```

In [21]:

```
lin_space
```

Out[21]:

```
array([1.        , 1.36363636, 1.72727273, 2.09090909, 2.45454545,
       2.81818182, 3.18181818, 3.54545455, 3.90909091, 4.27272727,
       4.63636364, 5.        ])
```

In [22]:

```
emp = np.empty((4,6)) #A Random array
```

In [23]:

```
emp
```

Out[23]:

```
array([[ 0.00000000e+000,  1.00937611e-320,  0.00000000e+000,
         4.78344603e-001,  8.50390406e-001,  1.20419856e+000],
       [-5.73695579e-001, -2.62086512e-001,  2.24225511e-001,
        -2.54675889e-001,  3.27915049e-001,  4.25118474e-001],
       [ 2.00133715e-001,  4.70330436e-001,  4.70601113e-001,
         1.41111658e+001,  5.47722558e+000,  2.23606798e+000],
       [ 1.11022302e-016, -6.66133815e-016,  1.63437886e+000,
         5.59500345e-001,  1.25559236e-001,  0.00000000e+000]])
```

In [24]:

```
ide = np.identity(10)
```

In [25]:

```
ide
```

Out[25]:

```
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]])
```

In [26]:

```
arr = np.arange(45)
```

In [27]:

```
arr
```

Out[27]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 1
5, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 3
2, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44])
```

In [28]:

```
arr.reshape(3,15)
```

Out[28]:

```
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]])
```

In [29]:

```
arr.ravel()
```

Out[29]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 1
5, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 3
2, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44])
```

Axis in NumPy

In [30]:

```
x = [[1,2,3],[4,5,6],[7,1,0]]
```

In [31]:

```
ar = np.array(x)
```

In [32]:

```
ar
```

Out[32]:

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

In [33]:

```
ar.sum(axis = 0)
```

Out[33]:

```
array([12,  8,  9])
```

In [34]:

```
ar.sum(axis = 1)
```

Out[34]:

```
array([ 6, 15,  8])
```

In [35]:

```
ar.T # Transpose
```

Out[35]:

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

In [36]:

```
ar.flat # Gives an iterator
```

Out[36]:

```
<numpy.flatiter at 0x7f8497482400>
```

In [37]:

```
for item in ar.flat:
    print(item)
```

```
1
2
3
4
5
6
7
1
0
```

In [38]:

```
 ar.ndim # Gives Dimension
```

Out[38]:

```
2
```

In [39]:

```
oneD = np.array([1,3,4,50,6])
```

In [40]:

```
oneD.argmax()  # This gives index at which maximum value is present.

# Jisme me bhi brackets lage hote h wo Methods hote hai. Means we are using this a
# Agar bracket nhi lagayenge then it's called attributes.
```

Out[40]:

```
3
```

In [41]:

```
oneD.argmin() # This gives index at which minimu value is present.
```

Out[41]:

```
0
```

In [42]:

```
oneD.argsort() # Gives array of indexes for if we use in that order then elements
```

Out[42]:

```
array([0, 1, 2, 4, 3])
```

In [43]:

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

In [44]:

```
arr
```

Out[44]:

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

In [45]:

```
arr.argmax()
```

Out[45]:

```
6
```

In [46]:

```
arr.argmax(axis = 0)
```

Out[46]:

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

In [47]:

```
arr.ravel()
```

Out[47]:

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

In [48]:

```
arr.reshape(9,1)
```

Out[48]:

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

Mathematical Operations in NumPy

In [50]:

```
arr
```

Out[50]:

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

In [49]:

```
arr2 = np.array([[1,2,2],[1,0,3],[1,1,0]])
```

In [52]:

```
arr2
```

Out[52]:

```
array([[1, 2, 2],
       [1, 0, 3],
       [1, 1, 0]])
```

In [51]:

```
arr+arr2
```

Out[51]:

```
array([[2, 4, 5],
       [5, 5, 9],
       [8, 2, 0]])
```

In [54]:

```
naya = arr*arr2
```

In [55]:

```
naya
```

Out[55]:

```
array([[ 1,  4,  6],
       [ 4,  0, 18],
       [ 7,  1,  0]])
```

In [56]:

```
np.sqrt(naya)
```

Out[56]:

```
array([[1.        , 2.        , 2.44948974],
       [2.        , 0.        , 4.24264069],
       [2.64575131, 1.        , 0.        ]])
```

In [57]:

```
naya.sum()
```

Out[57]:

```
41
```

In [58]:

```
naya.max()
```

Out[58]:

```
18
```

In [59]:

```
naya.min()
```

Out[59]:

```
0
```

In [60]:

```
np.where(naya>6)
```

Out[60]:

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

In [61]:

```
type(np.where(naya>6))
```

Out[61]:

```
tuple
```

In [62]:

```
np.count_nonzero(naya)
```

Out[62]:

```
7
```

In [63]:

```
np.nonzero(naya)
```

Out[63]:

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

In [64]:

```
import sys
```

In [69]:

```
py_ar = [0,4,55,2] # A python array
```

In [66]:

```
np_ar = np.array([0,4,55,2]) # A numPy Array
```

In [73]:

```
sys.getsizeof(1)*len(py_ar)
```

Out[73]:

```
112
```

In [74]:

```
np_ar.itemsize*np_ar.size # Hence it takes less memory
```

Out[74]:

```
32
```