Welcome to NumpyTutorial

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

In [2]:
```
myarr = np.array([[3,6,37,7]], np.int32)
```

In [3]:
```
myarr[0,1]
```

Out[3]:
```
6
```

In [4]:
```
myarr.shape
```

Out[4]:
```
(1, 4)
```

In [5]:
```
myarr.dtype
```

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

In [6]:
```
myarr[0,1]
```

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

In [7]:
```
myarr[0,1] = 45
```

In [8]:
```
myarr
```

Out[8]:
```
array([[ 3, 45, 37,  7]])
```

Array creation: 1 Conversion from other Python structures (i.e. lists and tuples)

In [9]:
```
listarray = np.array([[1,2,3],[5,8,5],[0,3,1]])
```

In [10]:
```
listarray
```

Out[10]:
```
array([[1, 2, 3],
       [5, 8, 5],
       [0, 3, 1]])
```

In [11]:
```
listarray.dtype
```

Out[11]:
```
dtype('int32')
```

In [12]:
```
listarray.shape
```

Out[12]:
```
(3, 3)
```

In [13]:
```
listarray.size
```

Out[13]:
```
9
```

Array creation: 2 Intrinsic NumPy array creation functions (e.g. arange, ones, zeros, etc.)

Zeros

In [14]:
```
zeros = np.zeros((2, 5))
```

In [15]:
```
zeros
```

Out[15]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

In [16]:
```
zeros.dtype
```

Out[16]:
```
dtype('float64')
```

In [17]:
```
zeros.shape
```

Out[17]:
```
(2, 5)
```

arange

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])
```

linspace

ye mujhe 1 se lekar 5 tak equally linearly spaced 12 elements dega

In [20]:
```
lspace = np.linspace(1,5,12)
```

In [21]:
```
lspace
```

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]:
```
lspace = np.linspace(1,50,10)
```

In [23]:
```
lspace
```

Out[23]:
```
array([ 1.        ,  6.44444444, 11.88888889, 17.33333333, 22.77777778,
       28.22222222, 33.66666667, 39.11111111, 44.55555556, 50.        ])
```

In [24]:
```
lspace = np.linspace(1,4,4)
```

In [25]:
```
lspace
```

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

empty

to np.empty kya karega aaplog ko khali array de dega (4,6) ka or sare ke sare jo elements honge wo random hoga, or aaplogo ko uske bad jo bhi value chahiye aaplog isko assigne kara sakte hai

In [26]:
```
emp =np.empty((4,6))
```

In [27]:
```
emp
```

Out[27]:
```
array([[6.23042070e-307, 1.42417221e-306, 1.37961641e-306,
        6.23040033e-307, 6.23053954e-307, 9.34609790e-307],
       [8.45593934e-307, 9.34600963e-307, 1.86921143e-306,
        6.23061763e-307, 8.90104239e-307, 6.89804132e-307],
       [1.33512376e-306, 6.89806849e-307, 9.34609790e-307,
        1.69121096e-306, 1.05700515e-307, 1.11261774e-306],
       [1.29060871e-306, 8.34424766e-308, 8.34445138e-308,
        1.37959129e-306, 1.02360528e-306, 0.00000000e+000]])
```

empty like

to basically ye karta kya hai. apne koi purana array banaya hoga, uss array ka jo bhi size tha usko copy karke ek empty array bana deta hai.

In [28]:
```
emp_like = np.empty_like(lspace)
```

In [29]:
```
emp_like
```

Out[29]:
```
array([1., 2., 3., 4.])
```

Identity

*Ye hume ek identity matrics deta hai (45*45)*

In [30]:
```
ide = np.identity(45)
```

In [31]:
```
ide
```

Out[31]:
```
array([[1., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 0., 1.]])
```

In [32]:
```
ide.shape
```

Out[32]:
```
(45, 45)
```

reshape

In [33]:
```
arr = np.arange(99)
```

In [34]:
```
arr
```

Out[34]:
```
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, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98])
```

Man lo agar me chahta hun ise reshape karna. to yaha kya hai 99 means (1,9) yaniki 1D array, to me isko (3,33) yaniki 3D array bhi bana sakta hum

In [35]:
```
arr.reshape(3,33)
```

Out[35]:
```
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, 60, 61, 62, 63, 64,
        65],
       [66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
        82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
        98]])
```

lekin agar me arr.reshape ko (3,31) karu to ye error dega, kyuki (3,31) me kitne element hai 93 or humara array kitne element ka hia 99 to isiliye error dega, means aap isko jis bhi shape me dena chahte hai unko multilpy karne se total element aana cahiye.

Abb arr change nahi hua hai usko change karne ke liye usko varible me dalna padega....eg.

In [37]:
```
arr
```

Out[37]:
```
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, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98])
```

In [ ]:
```
arr = arr.reshape(3,33)
```

In [ ]:
```
arr
```

ravel

isse hum array ko sidha kar dete hai jo reshape se humne change kiya hai usko sidha karne ke liye hum ravel ka use karte hai, 3D ko wapas 1D banane ke liye

In [ ]:
```
arr = arr.ravel()
```

In [ ]:
```
arr
```

In [ ]:
```
arr.shape
```

Numpy Axis

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

In [ ]:
```
ar = np.array(x)
```

In [ ]:
```
ar
```

Abb hum agar axis 0 ka sum nikalenge to

In [ ]:
```
ar.sum(axis=0)
```

Or agar hum axis 1 ka sum nikale to

In [ ]:
```
ar.sum(axis=1)
```

Atributes and Methods of NumPy

1.Transpose Array

Transpose matlab jo row tha wo column or jo column tha wo row

In [ ]:
```
ar.T
```

2.Flatiter

Ye ek iterator deta hai, or isko me for loop laga kar iterate kar sakta hun

ar.flat

In [ ]:
```
for item in ar.flat:
    print(item)
```

to ye mujhe iske sare items de dega, kyuki humne ar ko change nahi kiya to jo humra purana wala ar hai usi ke sare items dega ye

3.No. of Dimensions

In [ ]:
```
ar.ndim
```

4.Size

In [ ]:
```
ar.size
```

5.Total byte consume

Ye aapke array ke elements kitni jagah kha raha hai wo batata hai, kitni bytes le raha hai.

In [ ]:
```
ar.nbytes
```

Methods

1.argmax

argmax humara index deta hai hai jahan par maximum element hai, ab yahan par hum ek 1D array banayenge.

In [ ]:
```
one = np.array([1,3,4,634,2])
```

In [ ]:
```
one.argmax()
```

'or humne jo yahan parenthesis lagaya hai ek parenthesis open kiya or ek band kiya iska matlb mene isko (argmax) ko as an "Function" use kiya hai

2.argmin

argmin hume index deta hia jahan par minimum element hai

In [ ]:
```
one.argmin()
```

3.argshort

In [ ]:
```
one.argsort()
```

arhsort mujhe ek array dega or ye mujhe array ki wo indices dega jo ki iss array ko sort kar dega. wo indices, jis order me agar ye array ke element hote to wo sort ho jata, yaniki 0 par uska sabse chota element,fir uske ek kadam par 4 index par hai

to yahan aap logo ko badhte kram me array ke indicess ka array mill jata hai

These methods in 2-D array

In [ ]:
```
ar
```

In [ ]:
```
ar.argmin()
```

In [ ]:
```
ar.argmax()
```

In [ ]:
```
ar.argmax(axis=0)
```

to axis=0 ka mtlb hai ye pehle column1 pa dekhega kon se index me sabse bada no hai jo index no 2 par hai, fir ye column2 par dekhega ko ki index no 1 par hai or column3 me bhi index no 1 par hi hai to isiliye isne [2,1,1] diya

In [ ]:
```
ar.argmax(axis=1)
```

Same axix=0 ke jaisa hi yahan par bhi hua

In [ ]:
```
ar.argsort(axis=1)
```

yahan par axis=1 matlb ye row1 me dekhega element ko fir elements ki increasing order ke index ko print karega like R1=[1,2,3] ka sort hoga [0,1,2], R2=[4,5,6] ka bhi wohi [0,1,2] or R3=[7,1,0] ka hoga [2,1,0] ye same kam karega 1-D ke jaisa hi

In [ ]:
```
ar.argsort(axis=0)
```

same ye bhi waise hi kam karta hai

In [ ]:
```
ar.ravel()
```

ravel se ye sidha ho jata hai same 1-D ke tarah hi

In [ ]:
```
ar.reshape((9,1))
```

reshape in 9 cross 1

In [ ]:
```
ar.reshape((9,))
```

reshape in 9 cross 0

Mathematical operations in array

Matrices Operations

In [ ]:
```
ar
```

Aab hum yahan par ek dusra 2-D array banayenge

In [ ]:
```
ar2 = np.array([[1,2,1],
               [4,0,6],
               [8,1,0]])
```

In [ ]:
```
ar+ar2
```

To aap dekh sakte hai isne dono array ko jod diya hai badi asani se

Abb yahi kam agar hum python list ke sath karte to nahi hota

In [ ]:
```
[324,34]+[34,546]
```

to aap dekh sakte hai humara list yahan add nahi hua ye extend ho gaya

In [ ]:
```
ar
```

In [ ]:
```
ar2
```

In [ ]:
```
ar*ar2
```

to aap dekh sakte hai yahan isne element wise multiply kar diya hai

In [ ]:
```
ar
```

In [ ]:
```
ar2
```

In [ ]:
```
ar - ar2
```

to aap dekh sakte yahan isne element wise substract bhi kar diya

In [ ]:
```
np.sqrt(ar)
```

isse ye element wise square root le lega

In [ ]:
```
ar.sum()
```

sare element ka sum de diya

In [ ]:
```
ar.max()
```

ye ar ka sabse bada no de raha hai

In [ ]:
```
ar.min()
```

ye ar ka sabse chota no de raha hai

Finding elements in array

1.Where

In [ ]:
```
ar
```

In [ ]:
```
np.where(ar>5)
```

to isne mujhe arrays ka ek tuple return kar diya ki yaha-yahan par apke pass elements hai jo ki 5 se bade hai, or hum iska type bhi check kar sakte hai

In [ ]:
```
type(np.where(ar>5))
```

2.count_nonzero

In [ ]:
```
np.count_nonzero(ar)
```

ye count karke de dega kitne nonzero elements hai humare pass

3.nonezero

In [ ]:
```
np.nonzero(ar)
```

to ye basically har dimension ke liye ek tuple dega mujhe

aab mene kaha [1,2] ko = 0 kar do

In [ ]:
```
ar[1,2]=0
```

In [ ]:
```
np.nonzero(ar)
```

aap dekh sakte hai aab yahan par [1,2] yahan se gayab ho gaya hai

abb hum dekhenge kis tarah numpy kam space leta hai

In [ ]:
```
import sys
```

In [ ]:
```
py_ar = [0,4,55,2]
```

ye humara python array hai

In [ ]:
```
np_ar = np.array(py_ar)
```

or ye humara numpy array hai

In [ ]:
```
sys.getsizeof(1) * len(py_ar)
```

sys.getsizeof(1) ye mujhe python array ke ek element ka size batata hai or fir hum usko pure python array ke length ke sath multiply kar diya

In [ ]:
```
np_ar.itemsize * np_ar.size
```

np_ar.itemsize ye mujhe numpy array ke ek element ka size batata hai or fir hum usko pure numpy array ke size ke sath multiply kar diya

To aap dekh sakte hai python array humara 112 bytes le raha hai or humara numpy array sirf 16 bites le raha hai, to aap dekh sakte hai hume kitna fayda hota hai

In [ ]:
```

```