Numpy

Installing numpy

```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\uttam thakur\anaconda3\lib\site-packages (1.20.3)
Note: you may need to restart the kernel to use updated packages.
```

Note:-

1. Numpy library array me kaam karti hai. Array bhi list ki tarah multiple element store karta hai par array fast hota as compare to list.

using numpy

```
import numpy as np
```

Note:-

1. Jab bhi numpy ko use karte hai pehle use import karna padta hai

```
my_list=[10,20,30,40,50,60,70,10.4,'60.4']   # hetorgeneous
```

```
# Let's create a numpy array from the list "my_list"
x = np.array(my_list)          # homogeneous
x
```

```
array(['10', '20', '30', '40', '50', '60', '70', '10.4', '60.4'],
      dtype='<U32')
```

```
type(x)
```

```
numpy.ndarray
```

#numpy libray me se array naam k function ko call kiya hai or (l) naam k list ko array me change kiya

note:-

1. Array single data type pe kaam karta hai Is liye ise homogeneous hota hai
2. Agar koi ek bhi data type string hua to sare data type ko string me change kar deta hai.
3. List multipe data type pe kaam karta hai is liye use hetrogeneus kehte hain.

4.(60.4) ko integer me change karta toh (0.4) ka loss hota, is liye numpy data loss ko prevent karta hai.

Ndim

Ndim dimensions check karne kaam aata hai

```
l=[1,2,3,4]
x=np.array(l)
x.ndim
```

output:- 1

Creating multi_dimensional array

```
l=[[1,2,3],[4,5,6],[7,8,9]]
a=np.array(l)
print(a)
print(type(a))
print("dimensions of array:- ",a.ndim)  #dimensions check karne k liyr last k brackets count kar lo
print("size of array:- ",a.size)
print("shape of matrix:- ",a.shape)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
<class 'numpy.ndarray'>
dimensions of array:-  2
size of array:-  9
shape of matrix:-  (3, 3)

3
```

Note:-

1. Size function array k elements batata hai.
2. Shape function array k shape batata hai ki kitne by kitne ki matrix hai.
3. Size elements ko count karta hai.
4. Length list ko count karta hai.

BUILT-IN METHODS AND FUNCTIONS

- **rand( )**

```
# "rand()" uniform distribution between 0 and 1
x =np.random.rand(5)
x
```

```
array([0.96536219, 0.98125757, 0.35981617, 0.34491806, 0.85875857])
```

Note:-

1. Numpy ki library me se random naam k module ko call kiya or random module random values dega rand() function ki help se.
2. Or ye 5 uniform distributed values dega 0 se 1 k beech me.

Creating a matrix using rand () function with random values

```
# you can create a matrix of random number as well
x = np.random.rand(3,4)
x
```

```
array([[0.82996883, 0.50826092, 0.10180974, 0.63434323],
       [0.65740801, 0.06077867, 0.24345894, 0.01794433],
       [0.10886852, 0.42512479, 0.59872937, 0.31578915]])
```

```
x.ndim
```

```
2
```

Here we get random data in 3 rows and 4 column

- **X = np.random.rand (no. of rows , no. of column)**

- **randn( )**

1. it gives normal distributed values or isme mostly values -2 se 2 k beech mostly 0 k aas pass hongi or bohot hi rare -3 se 3 aati hai.

```
# "randn()" normal distribution
x = np.random.randn(15)
x
```

```
array([ 0.67895596,  2.41126037,  0.00763651, -0.98593767, -1.10049447,
       -0.54478071, -0.02732841, -1.20112615, -0.77572644,  1.04218302,
        0.38650065,  0.67449692, -0.52693278, -0.62244399,  0.34282627])
```

Creating a matrix using randn () function with random values

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

```
array([[-0.08307465, -0.69612123, -0.0214274 ],
       [-0.43783242,  0.34972269,  0.51506286],
       [-1.11558049, -1.29269116, -0.51410614],
       [-0.7077529 ,  0.08240517,  1.07863018]])
```

- **randint( )**

randint() function koi bhi random integer value nikal k dega given arguments k beech me.

```
# "randint" is used to generate random integers between upper and lower bounds
x = np.random.randint(10,500)
print(x)
```

```
53
```

```
# "randint" can be used to generate a certain number of random itegers as follows
x = np.random.randint(5, 100, 12)
x
```

```
array([36, 22, 22, 30, 27, 48, 23,  5, 51, 98, 63, 99])
```

- **X = np.random.rand (kaha se value chiye ,kaha tak value chaiye , or kitni values chiye) ye random values dega**

- **arange( )**

```
# np.arange creates an evenly spaced values within a given interval
x = np.arange(1, 11)
x
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

- **X = np.random.rand (kaha se value chiye ,kaha tak value chaiye) ye line wise / evenly spaced values dega**

Stepping in arange function

```
# Create an evenly spaced values with a step of 5
x = np.arange(0, 101,5)
x
```

```
array([  0,   5,  10,  15,  20,  25,  30,  35,  40,  45,  50,  55,  60,
        65,  70,  75,  80,  85,  90,  95, 100])
```

- **X = np.random.rand (kaha se value chiye ,kaha tak value chaiye , itne no. ki stepping chiye)**

- **eye( )**

used to create identity matrix 3by3 ,4by4, etc matrics. Isme diagonal elements 1 hota hai or baki k elements zero honge.

Isme values float me hoti hai kyu ki numpy float me kaam karta hai taki data loss nah o paye.

```
# create a diagonal of ones and zeros everywhere else
x = np.eye(4)   # identity matrix
x
```

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

X=np.eye(**kitne by kitne ki matrix chiye**)

- **ones( )**

ones function (1) wali matrx bana k deta hai

```
# Array of ones
x = np.ones(5)
x
```

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

For array:- x=np.ones(kitne element chiye array me)

```
# Matrices of ones
x = np.ones((5,3))
x
```

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

For matrix:- x=np.ones(no. of rows , no. of columns)

- **Data type change in ones matrix**

For matrix:- x=np.ones([no. of rows , no. of columns],kon sa data type chiye)

```
x = np.ones([5,3],str)
x
```

```
array([['1', '1', '1'],
       ['1', '1', '1'],
       ['1', '1', '1'],
       ['1', '1', '1'],
       ['1', '1', '1']], dtype='<U1')
```

- **zeros( )**
  same as ones

```
# Array of zeros
x = np.zeros([3,3])
x
```

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

```
x = np.zeros([2,3],int)
x
```

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

Creating custom matrix with ones

For matrix:- x=np.ones([no. of rows , no. of columns])* koi sab no. jiski matrix chiye

SHAPE, LENGTH, TYPE, RESHAPE, AND MAX/MIN VALUES

```
# Let's define a one-dimensional array
my_list = [-30, 4, 50, 60, 29, 15, 22]
x = np.array(my_list)
x
```

```
array([-30,   4,  50,  60,  29,  15,  22])
```

```
# Get Length of a numpy array
len(x)
```

```
7
```

ye 7 rows hain

```
# Get shape
x.shape
```

```
(7,)
```

```
# Obtain the datatype
x.dtype
```

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

```
my_list = [30, 4, 50, 60, 29, 15, -22, 90,10,20,14,55]
my_list
x = np.array(my_list)
x
```

```
array([ 30,   4,  50,  60,  29,  15, -22,  90,  10,  20,  14,  55])
```

```
# Reshape 1D array into a matrix
z = x.reshape(4,1,3)
print(z)
```

```
[[[ 30   4  50]]

 [[ 60  29  15]]

 [[-22  90  10]]

 [[ 20  14  55]]]
```

- **dtype memory space deta hai ki element kitne bytes space lere hain.**
- **Reshape 1d array ko matrix k form me change karta hai**

`Z=x.reshape(kitne row banana hai , kitne column banana hain)`

Argu,ments ka multiply total no. of 1D array k barabar honi chiye . jaise 1D array me 12 elements hain toh matrix banana k time `Z=x.reshape(a,b)` a or b ka multiply 12 hona hi chiye.

```
# Obtain the maximum element (value)
z.max()
```

```
90
```

```
# Obtain the minimum element (value)
z.min()
```

```
-22
```

Max or min value find karne k liye array me se

SLICING AND INDEXING

```
x = np.array([20, 40, 50, 21, 15])
x
```

```
array([20, 40, 50, 21, 15])
```

```
x[1:4]
```

```
array([40, 50, 21])
```

```
# Access specific index from the numpy array
x[2]
```

```
50
```

```
x[0] = -15
```

```
x
```

```
array([-15,  40,  50,  21,  15])
```

0 index pe value change kar k -15 kara hai

```
# Let's define a two dimensional numpy array
matrix = np.random.randint(1,20, (5, 5))
matrix
```

```
array([[10, 16, 18, 14,  4],
       [18,  9, 10, 19, 12],
       [15,  9, 10, 10, 16],
       [10,  1, 10,  8, 15],
       [18, 13,  6,  7, 14]])
```

```
# Get a row from a mtrix
matrix[2][4]
```

```
16
```

Matrix[indexing of row ,jo no. chiye uski indexing]

Changing row in matrix

```
matrix[2] = [5,8,4,3,2]
```

```
matrix
```

```
array([[10, 16, 18, 14,  4],
       [18,  9, 10, 19, 12],
       [ 5,  8,  4,  3,  2],
       [10,  1, 10,  8, 15],
       [18, 13,  6,  7, 14]])
```

Slicing se row access

```
matrix
```

```
array([[10, 16, 18, 14,  4],
       [18,  9, 10, 19, 12],
       [ 5,  8,  4,  3,  2],
       [10,  1, 10,  8, 15],
       [18, 13,  6,  7, 14]])
```

```
mini_matrix = matrix[:3]
mini_matrix
```

```
array([[10, 16, 18, 14,  4],
       [18,  9, 10, 19, 12],
       [ 5,  8,  4,  3,  2]])
```

```
mini_matrix = matrix[2:]
mini_matrix
```

```
array([[ 5,  8,  4,  3,  2],
       [10,  1, 10,  8, 15],
       [18, 13,  6,  7, 14]])
```