NumPy

What is numpy?

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on array including mathematical, logical, shape manipulation, sorting, selecting, I/0, discrete Fourier transforms, basic linear algebra, basi statistical operations, random simulation and much more.

At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types

1.Creating NumPy Array

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

a = np.array([1,2,3])
print(a)
```

```
[1 2 3]
```

In [2]:
```
print(type(a))
```

```
<class 'numpy.ndarray'>
```

In [3]:
```
# 2D and 3D
b = np.array([[1,2,3],[4,5,6]])
print(b)
```

```
[[1 2 3]
 [4 5 6]]
```

In [4]:
```
c = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
print(c)
```

```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

2.Creating NumPy Array with Data Type

In [5]:
```
# dtype
# let say mujhe float datatype ka array banana hai
np.array([1,2,3],dtype=float)
```

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

In [6]:
```
np.array([1,2,3],dtype=bool)
```

Out[6]:
```
array([ True,  True,  True])
```

In [7]:
```
np.array([1,2,3],dtype=complex)
```

Out[7]:
```
array([1.+0.j, 2.+0.j, 3.+0.j])
```

3.Creating NumPy Array using np.Arange() and .reshape()

In [8]:
```
# np.arange
np.arange(1,11)
```

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

In [9]:
```
np.arange(1,11,2)
```

Out[9]:
```
array([1, 3, 5, 7, 9])
```

In [10]:
```
# with reshape
np.arange(1,11).reshape(5,2)
```

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

4.Creating NumPy Array with np.ones and np.zeros

In [11]:
```
np.ones((3,4))
```

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

In [12]:
```
np.zeros((3,4))
```

Out[12]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

5.Creating NumPy Array with np.random

In [13]:
```
np.random.random((3,4))
```

Out[13]:
```
array([[0.22017328, 0.09314581, 0.5138173 , 0.15559058],
       [0.49418057, 0.04875155, 0.83287926, 0.90993352],
       [0.64177404, 0.62074344, 0.81311745, 0.40473122]])
```

6.Creating NumPy Array with linearly spaced array- np.linspace()

In [14]:
```
np.linspace(-10,10,10)
```

Out[14]:
```
array([-10.        ,  -7.77777778,  -5.55555556,  -3.33333333,
       -1.11111111,   1.11111111,   3.33333333,   5.55555556,
        7.77777778,  10.        ])
```

7.Creating NumPy Array with Identity Matrix Creation - np.identity() or np.eye()

In [15]:
```
np.identity(3)
```

Out[15]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

8.NumPy Array Attributes

In [16]:
```
a1 = np.arange(10)
a2 = np.arange(12,dtype=float).reshape(3,4)
a3 = np.arange(8).reshape(2,2,2)

a3
```

Out[16]:
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [17]:
```
# ndim(no.of dimentions)
# isko use karke aap pata kar sakte ho iska demention kya hai kitne dimensio

a2.ndim
```

Out[17]:
```
2
```

In [18]:
```
# shape
# ye basically bataiga ki har dimension me row aur column kitne hai

a1.shape
```

Out[18]:
```
(10,)
```

In [19]:
```
a2.shape
```

Out[19]:
```
(3, 4)
```

In [20]:
```
# size
a3.size
```

Out[20]:
```
8
```

In [21]:
```
print(a2.size)
a2
```

```
12
```

Out[21]:
```
array([[ 0.,  1.,  2.,  3.],
       [ 4.,  5.,  6.,  7.],
       [ 8.,  9., 10., 11.]])
```

In [22]:
```
# itemsize
a3.itemsize
```

Out[22]:
```
4
```

In [23]:
```
# dtype
print(a1.dtype)
print(a2.dtype)
print(a3.dtype)
```

```
int32
float64
int32
```

9.Changing Datatype

In [24]:
```
# astype
a3.astype(np.int32)
```

Out[24]:
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

10.Array Operations

In [25]:
```
a1 = np.arange(12).reshape(3,4)
a2 = np.arange(12,24).reshape(3,4)
```

In [26]:
```
# Scaler operations

# arithmetic(u can use all the arithmetic operator on like *,+,-,/,//,**,etc
a1 * 2
```

Out[26]:
```
array([[ 0,  2,  4,  6],
       [ 8, 10, 12, 14],
       [16, 18, 20, 22]])
```

In [27]:
```
# relational (it will provide boolean operation)
a2 > 15 # u can use >,<,=,!=,etc
```

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

In [28]:
```
# Vector operations(you can do arithmetic operation on 2 arrays)

# arithmetic operator(*,+,-,/,//,**,etc)
# a1 * a2
# a1 + a2
# a1 - a2
# a1 ** a2
# a1 // a2
a1 / a2
```

Out[28]:
```
array([[0.        , 0.07692308, 0.14285714, 0.2       ],
       [0.25      , 0.29411765, 0.33333333, 0.36842105],
       [0.4       , 0.42857143, 0.45454545, 0.47826087]])
```

11.Array Function

In [29]:
```
a1 = np.random.random((3,3))
a1 = np.round(a1*100)

a1
```

Out[29]:
```
array([[ 9., 76., 25.],
       [27., 41., 97.],
       [85., 99., 23.]])
```

In [30]:
```
# max/min/sum/prod

np.max(a1) # maximum no nikal kar dega 91
```

Out[30]:
```
99.0
```

In [31]:
```
np.min(a1) # minimum no nikal k dega 0
```

Out[31]:
```
9.0
```

In [32]:
```
np.sum(a1) # Total ka sum kar dega
```

Out[32]:
```
482.0
```

In [33]:
```
np.prod(a1) # multiplication of all number
```

Out[33]:
```
355383632290500.0
```

In [34]:
```
# mujhe har row ka min chaiye yaha axis use hoga
# 0 -> column and 1 -> row
np.min(a1,axis=1)
```

Out[34]:
```
array([ 9., 27., 23.])
```

In [35]:
```
# mean/median/std/var

np.mean(a1)
```

Out[35]:
```
53.55555555555556
```

In [36]:
```
np.median(a1)
```

Out[36]:
```
41.0
```

In [37]:
```
np.std(a1,axis=0)
```

Out[37]:
```
array([32.42769735, 23.84673283, 34.42221505])
```

In [38]:
```
np.var(a1,axis=1)
```

Out[38]:
```
array([ 816.22222222,  914.66666667, 1090.66666667])
```

In [39]:
```
# trigonometric function

np.sin(a1)
```

Out[39]:
```
array([[ 0.41211849,  0.56610764, -0.13235175],
       [ 0.95637593, -0.15862267,  0.37960774],
       [-0.17607562, -0.99920683, -0.8462204 ]])
```

In [40]:
```
# dot product
# dot prod me kya karte ho basically 2 matrices ka bich ka dot prod nikalte

a2 = np.arange(12).reshape(3,4)
a3 = np.arange(12,24).reshape(4,3)

a2
```

Out[40]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [41]:
```
a3
```

Out[41]:
```
array([[12, 13, 14],
       [15, 16, 17],
       [18, 19, 20],
       [21, 22, 23]])
```

In [42]:
```
np.dot(a2,a3)
```

Out[42]:
```
array([[114, 120, 126],
       [378, 400, 422],
       [642, 680, 718]])
```

In [43]:
```
# log and exponent
np.exp(a1)
```

Out[43]:
```
array([[8.10308393e+03, 1.01480039e+33, 7.20048993e+10],
       [5.32048241e+11, 6.39843494e+17, 1.33833472e+42],
       [8.22301271e+36, 9.88903032e+42, 9.74480345e+09]])
```

In [44]:
```
# round/floor/ceil
```

12.Indexing and Slicing

In [45]:
```
# Indexing

a1 = np.arange(10)
a2 = np.arange(12).reshape(3,4)
a3 = np.arange(8).reshape(2,2,2)

a1
```

Out[45]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [46]:
```
a1[0]
```

Out[46]:
```
0
```

In [47]:
```
a2
```

Out[47]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [48]:
```
a2[-2][2]
```

Out[48]:
```
6
```

In [49]:
```
a2[1,2]
```

Out[49]:
```
6
```

In [50]:
```
a2[1,0]
```

Out[50]:
```
4
```

In [51]:
```
a3
```

Out[51]:
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]])
```

In [52]:
```
 a3[0,1,0]
```

Out[52]:
```
2
```

In [53]:
```
a3[0,0,0]
```

Out[53]:
```
0
```

In [56]:
```
a3[1,1,0]
```

Out[56]:
```
6
```

In [57]:
```
# Slicing

a1 = np.arange(10)
a2 = np.arange(12).reshape(3,4)
a3 = np.arange(8).reshape(2,2,2)

a1
```

Out[57]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [58]:
```
# a1 k ander se nikalna hai 2,3,4

a1[2:5]
```

Out[58]:
```
array([2, 3, 4])
```

In [59]:
```
a1[2:5:2]
```

Out[59]:
```
array([2, 4])
```

In [60]:
```
a2
```

Out[60]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [62]:
```
a2[0,:]
```

Out[62]:
```
array([0, 1, 2, 3])
```

In [63]:
```
a2[:,2]
```

Out[63]:
```
array([ 2,  6, 10])
```

In [68]:
```
a2[1:,1:3]
```

Out[68]:
```
array([[ 5,  6],
       [ 9, 10]])
```

In [72]:
```
a2[::2,::3]
```

Out[72]:
```
array([[ 0,  3],
       [ 8, 11]])
```

In [84]:
```
a2[::2]
```

Out[84]:
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11]])
```

In [103]:
```
a2[1,0::3]
```

Out[103]:
```
array([4, 7])
```

In [109]:
```
a2[:2,1:]
```

Out[109]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [111]:
```
a3 = np.arange(27).reshape(3,3,3)
```

In [112]:
```
a3
```

Out[112]:
```
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]]])
```

In [120]:
```
a3[0,1]
```

Out[120]:
```
array([3, 4, 5])
```

In [122]:
```
a3[1,:,1]
```

Out[122]:
```
array([10, 13, 16])
```

In [128]:
```
a3[2,1:,1:]
```

Out[128]:
```
array([[22, 23],
       [25, 26]])
```

In [148]:
```
a3[::2,0,::2]
```

Out[148]:
```
array([[ 0,  2],
       [18, 20]])
```

13.Iterating

In [152]:
```
a1

for i in a1:
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

In [150]:
```
a2
```

Out[150]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [153]:
```
for i in a2:
    print(i)
```

```
[0 1 2 3]
[4 5 6 7]
[ 8  9 10 11]
```

In [151]:
```
a3
```

Out[151]:
```
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]]])
```

In [154]:
```
for i in a3:
    print(i)
```

```
[[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]]
```

In [155]:
```
for i in np.nditer(a3):
    print(i)
```

```
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
```

15.Reshaping

In [ ]:
```
# reshape
```

In [157]:
```
# Transpose(transpose row ko column aur column ko row kar deta hai)

np.transpose(a2)
```

Out[157]:
```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

In [158]:
```
# Another syntax
a2.T
```

Out[158]:
```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

In [159]:
```
# ravel(ravel Kitne v dimension k array ko 1-d me convert kar deta hai)

a3.ravel()
```

Out[159]:
```
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])
```

In [160]:
```
a2.ravel()
```

Out[160]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

16.Stacking

2 numpy array ko stack(zor) kar sakte ho ek horizontal aur dusra vertical [ ] [ ] --> [ ]

In [162]:
```
# horizontal stacking
a4 = np.arange(12).reshape(3,4)
a5 = np.arange(12,24).reshape(3,4)
```

In [163]:
```
a4
```

Out[163]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [164]:
```
a5
```

Out[164]:
```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [165]:
```
np.hstack((a4,a5))
```

Out[165]:
```
array([[ 0,  1,  2,  3, 12, 13, 14, 15],
       [ 4,  5,  6,  7, 16, 17, 18, 19],
       [ 8,  9, 10, 11, 20, 21, 22, 23]])
```

In [166]:
```
# vertical stacking
np.vstack((a4,a5))
```

Out[166]:
```
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]])
```

17.Splitting

In [168]:
```
# horizontal splitting

a4
```

Out[168]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [169]:
```
np.hsplit(a4, 2)
```

Out[169]:
```
[array([[0, 1],
        [4, 5],
        [8, 9]]),
 array([[ 2,  3],
        [ 6,  7],
        [10, 11]])]
```

In [170]:
```
# vertical splitting

a5
```

Out[170]:
```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [171]:
```
np.vsplit(a5,3)
```

Out[171]:
```
[array([[12, 13, 14, 15]]),
 array([[16, 17, 18, 19]]),
 array([[20, 21, 22, 23]])]
```

In [ ]:
```

```