Numpy

It is a module or library

Numpy means numerical python

It contain multi dimensional array objects.

Numpy provide lot of functions to work in a domain of linear algebra, fourier tranform and matrices.

It provides functions related to arrays.

It creates an array called nd(n-dimensional) array.

**INSTALLATON OF NUMPY**

COMMAND: pip install numpy

**CHEK THE VERSION**

Print(numpy.—version--)

**import and creating an alias name as**

import numpy

import numpy as np \\creating alias name\\

**TYPES OF ARRAYS**

* Single Dimensional Array

Ex: import numpy as np

a=np.array((1,2,3))

print(a)

**output**: array(1 2 3)

* Two Dimensional Array

import numpy as np

b=np.array([[,2,3],[4,5,6]])

print(b)

**output:** [[1 2 3]

[4 5 6]]

* 3-D array

import numpy as np

b=np.array([[[,2,3],[4,5,6]],[[1,2,3],[4,5,6]]])

print(b)

**output: [** [[1 2 3]

[4 5 6]]

[[1 2 3]

[4 5 6]]]

**To know the dimensions**

* For single dimensional array

import numpy as np

a=np.array((1,2,3))

a.ndim

o/p:1

* For 2-D array

import numpy as np

b=np.array([[,2,3],[4,5,6]])

b.ndim

o/p: 2

**To know the shape**

* For single dimensional array

import numpy as np

a=np.array((1,2,3))

a.shape

o/p:(3,)

* For 2-D array

import numpy as np

b=np.array([[1,2,3],[4,5,6]])

b.shape

o/p: (3,3)

**To know the data type**

* For single dimensional array

import numpy as np

a=np.array((1,2,3))

a.dtype

o/p:dtype(‘int32’)

* For 2-D array

import numpy as np

b=np.array([1.1,2.1,3.1])

b.dtype

o/p: dtype(‘float64’)

**To know the item size**

* For single dimensional array

import numpy as np

a=np.array((1,2,3))

a.size

o/p:3

* For 2-D array

import numpy as np

b=np.array([[1,2,3],[4,5,6]])

b.size

o/p: 6

**To know the number of bytes**

* For single dimensional array

import numpy as np

a=np.array((1,2,3))

a.nbytes

o/p:12

* For 2-D array

import numpy as np

b=np.array([[1,2,3],[4,5,6]])

b.nbytes

o/p: 24

**For Example**

Import numpy as np

c=np.array([1,2,3,4,5,6],[1,2,3,4,5,6])

c[1,3] //only [] is accessible//

**o/p:4**

c[0,:]

**o/p:array[1 2 3 4 5 6]**

c[:,2]

**o/p:array(3,3)**

* np.arrange:

**This function returns array with values in range and required data type**

**Syntax:** np.arange(start index , stop index , step size , dtype)

**Example**: **1**. import numpy a np

a=np.arange(0,10,2,dtype=”int”)

print(a)

**o/p: [0 2 4 6 8]**

**Ex:2**  import numpy a np

a=np.arrange(0,10,2,dtype=”float”)

print(a)

**o/p: [0. 2. 4. 6. 8.]**

**Ex:3**  import numpy a np

a=np.arrange(0,10,3)

print(a)

**o/p: [0 3 6 9]**

* **Linspace:**

**syntax:** np.linspace(start index ,stop index ,num ,end point ,retstep , dtype)

**num=number of required values in array (default value=50 //optio nal//)**

**endpoint=true-> stop index included**

**false->stop index excluded**

**(default=true //optional//)(//in general the starting point is included and ending point is excluded//)**

**Retstep=it gives the difference between two values in output**

**(by default it is false)**

**Example**:1 import numpy a np

a=np.linspace(0,10,5,endpoint=True)

print(a)

**o/p: [0. 2.5 5. 7.5 10.]**

**Ex:2** import numpy a np

a=np. linspace (0,10,5,endpoint=False)

print(a)

**o/p:[0. 2. 4. 6. 8.]**

**Ex:3** import numpy a np

a=np. linspace (0,10,5,endpoint=True,retstep=True)

print(a)

**o/p:(array([0. 2.5 5. 7.5 10. ]), 2.5)**

**Ex:4** import numpy a np

a=np. linspace (0,10,5,endpoint=False,retstep=False)

print(a)

**o/p: [0. 2. 4. 6. 8.]**

* **np.shape()**

This function will return a tuple specified indices and number of elements.

**EX:1** import numpy as np

a=([10,20,30],[40,50,60])

np.shape(a)

**o/p:** (2,3)

EX:2 import numpy as np

a=([10,20,30],[40,50,60],[70,80,90])

np.shape(a)

**o/p:** (3,3)

* **np.reshape()**

Here it accept array name to change he shape dimensions

EX:1 import numpy as np

a=([10,20,30],[40,50,60])

np.shape(a)

**o/p:** (2,3)

Now np**.**reshape(a,(3,2))

**o/p:** array([[10,20]

[30,40]

[50,60]])

* **np.zeros()**

**syntax:**np.zeros(shape,dtype,order)

shape=no. of rows,columns

dtype=data type//default float//

order= row wise=(”F”)

column wise=(“C”)//default column wise//

**Example:**

**for single dimensional array**

**Ex:1** Import numpy as np

a=np.zeros(5,dtype=”int”)

print(a)

**o/p:** array([0 0 0 0 0])

// for two dimensional array

Import numpy as np

a=np.zeros([4,5])

print(a)

**o/p:** array([0 0 0 0 0

0 0 0 0 0

0 0 0 0 0

0 0 0 0 0])

* **np.ones()**

**syntax:**np.ones(shape,dtype,order)

shape=no. of rows,columns

dtype=data type//default float//

order= row wise=(”F”)

column wise=(“C”)//default column wise//

**Example:** for single dimensional array

Import numpy as np

a=np.ones(5,dtype=”int”)

print(a)

**o/p:** array([1 1 1 1 1])

// for two dimensional array

Import numpy as np

a=np.ones([4,5])

print(a)

**o/p:** array([1 1 1 1 1

1 1 1 1 1

1 1 1 1 1

1 1 1 1 1])

* **np.random.random()**

It returns random sample of type float between 0 and 1

Ex:1 import numy as np

a=np.random.random()

print(a)

**o/p:** 0.837904988708216

if we want more than one sample

import numy as np

a=np.random.random(5)

print(a)

**o/p:** 0.8379, 0.8469, 0.6384, 0.7389, 0.110234

**Now if we want to create a matrix**

import numy as np

a=np.random.random([5,5])

print(a)

**o/p:** [0.6359 0.5846 0.5879 0.5741 0.2415

0.8524 0.5478 0.2541 0.2548 0.3547

0.2654 0.3654 0.5412 0.5874 0.6987

0.1586 0.6587 0.6987 0.2545 0.5426

0.2574 0.2548 0.1587 0.2475 0.9845]

* **np.full()**

**syntax:** np.full(shape, default value)

**EX:1** import numpy as np

a=np.full((2,3) ,5)

print(a)

**o/p:** array([5,5,5, [5,5,5])

* **np.eye()**

It conside square matrix.

It initializes 1’s on all diagnol positions and initializes o’s on all Non- diagnol positions

**syntax:** np.eye(shape, dtype)

**EX:1** import numpy as np

a=np.eye(3)

print(a)

**o/p:** array([1 0 0

0 1 0

0 0 1])

* **np.empty()**

**syntax:**np.empty(shape,dtype,order)multi dimensional data in row major(C-style) or column major(F-style) //default C//

**order:** It is used to store

**Ex:1** import numpy as np

a=np.empty(2)

print(a)

**o/p:** array([1.505e,0.00565e])

**Ex:**2 import numpy as np

a=np.empty([3,3],dtype=int)

print(a)

**o/p:** array([511e2 65e4 1445e

254e2 8243 0254e

254e 2554 185e])

* **Len(array):It displays number of rows**

**Ex1:** import numpy as np

a=len(np.array([[1,2,3],[4,5,6]])

print(a)

**o/p: 2**

**Ex1:** import numpy as np

a=len(np.array([[1,2,3,7],[4,5,6,8]].T)

print(a)

**o/p: 4**

* **array.astype(type): This method is used to change the data type**

**Ex1:** import numpy as np

a=np.array([1.2,2.3,3.4])

b=a.astype(int)

print(a)

print(b)

**o/p: array([1.2,2.3,3.4])**

**array([1,2,3])**

* **np.append():**//if we want to add elements to the existing array we are using append() method either column wise or row wise//

**syntax:** numpy.append(array, values, axis = None)

array=input array

values=values to be added to array

axis if axis= 0

if axis=1

Ex1: for single dimensional array

Import numpy as np

a=np.arange(0,10)

print(a)

**o/p:[0 1 2 3 4 5 6 7 8 9]**

**if I want to append single value to the existing array then**

**np.append(a,[20])**

**o/p array([10, 11, 12, 13, 14, 15, 16, 17, 18, 20])**

**if I want to append more than one values to the existing**

**array then**

**np.append(a,[30,40,50])**

**o/p array([10, 11, 12, 13, 14, 15, 16, 17, 18, 30, 40, 50])**

**Ex2:** import numpy as np

b=np.array([[1,2,3],[4,5,6]])

print(b)

**o/p: [[1 2 3]**

**4 5 6]]**

**np.append(b,[[50,60,70]])**

**o/p: array ([ 1, 2, 3, 4, 5, 6, 50, 60, 70])**

**np.append(b,[[50,51,52]],axis=0)**

**o/p: array ([[ 1, 2, 3],**

**[ 4, 5, 6],**

**[50, 51, 52]])**

**np.append(b,[[50],[51]],axis=1)**

**o/p: array ([[ 1, 2, 3, 50],**

**[ 4, 5, 6, 51]])**

* **np.insert()**

**This method is used to insert any element**

**either in row wise or column wise**

**syntax:** **np.insert(arr,obj,value,axis=none)**

**arr=input array**

**obj=position of element to insert**

**value=which value to be inserted**

**axis=0 row wise**

**=1 column wise**

**Ex1:** for single dimensional array

**import numpy as np**

**a=np.array([1,2,3,4,5])**

**print(a)**

**o/p: [1 2 3 4 5]**

**np.insert(a,2,7)**

**o/p: array([1, 2, 7, 3, 4, 5])**

**np.insert(a,4,8)**

**o/p: array([1, 2, 3, 4, 8, 5])**

**Ex2:** for 2-D array

**import numpy as np**

**b=np.array([[1,2,3,4],[5,6,7,8]])**

**print(b)**

**o/p:** [[1 2 3 4]

[5 6 7 8]]

**np.insert(b,1,100,axis=0)**

**o/p:**

array([[ 1, 2, 3, 4],

[100, 100, 100, 100],

[ 5, 6, 7, 8]])

**np.insert(b,1,100,axis=1)**

**o/p:** array ([[ 1, 100, 2, 3, 4],

[ 5, 100, 6, 7, 8]])

* **np.delete()**

**syntax:** np.delete(arr,obj,value,axis=none)

**Ex1: for single dimensional array**

**import numpy as np**

**a=np.array([1,2,3,4,5])**

**print(a)**

**o/p:** [1 2 3 4 5]

**np.delete(a,3)**

**o/p:** array([1, 2, 3, 5])

**EX:2 for 2-D array**

**import numpy as np**

**b=np.array([[1,2,3,4],[5,6,7,8]])**

**print(b)**

**o/p:** [[1 2 3 4]

[5 6 7 8]]

**np.delete(b,6)**

**o/p:** array([1, 2, 3, 4, 5, 6, 8])

**np.delete(b,[1,4])**

**o/p:** array([1, 3, 4, 6, 7, 8])

* **np.resize()**

This method is used to resize the array permanently

**Ex:1** import numpy as np

a=np.arange(0,18)

print(a)

**o/p:** [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17]

**a.resize(6,3)**

**print(a)**

**o/p**: [[ 0 1 2]

[ 3 4 5]

[ 6 7 8]

[ 9 10 11]

[12 13 14]

[15 16 17]]

**a.resize(3,6)**

**a**

**o/p:** array([[ 0, 1, 2, 3, 4, 5],

[ 6, 7, 8, 9, 10, 11],

[12, 13, 14, 15, 16, 17]])

**a.resize(3,2,3)**

**a**

o/p: array([[[ 0, 1, 2],

[ 3, 4, 5]],

[[ 6, 7, 8],

[ 9, 10, 11]],

[[12, 13, 14],

[15, 16, 17]]])

**Joining functions:concatenate,stack,vstack,hstack**

**np.conctenate():**join two arrays with respect to given axis

**syntax:** np.concatenate((array names),axis)

**Ex1**:**import numpy as np**

**a=np.arange(0,6).reshape(2,3)**

**print(a)**

**o/p:** [[0 1 2]

[3 4 5]]

**b=np.arange(7,13).reshape(2,3)**

**print(b)**

**o/p:** [[ 7 8 9]

[10 11 12]]

**np.concatenate((a,b))**

**o/p:** array([[ 0, 1, 2],

[ 3, 4, 5],

[ 7, 8, 9],

[10, 11, 12]])

**np.concatenate((a,b),axis=1)**

**o/p:** array([[ 0, 1, 2, 7, 8, 9],

[ 3, 4, 5, 10, 11, 12]])

**np.stack((a,b))**

**o/p:** array([[[ 0, 1, 2],

[ 3, 4, 5]],

[[ 7, 8, 9],

[10, 11, 12]]])

**np.stack((a,b),axis=1)**

**o/p:** array([[[ 0, 1, 2],

[ 7, 8, 9]],

[[ 3, 4, 5],

[10, 11, 12]]])

**np.vstack((a,b))**

**o/p:** array([[ 0, 1, 2],

[ 3, 4, 5],

[ 7, 8, 9],

[10, 11, 12]])

**np.hstack((a,b))**

**o/p:** array([[ 0, 1, 2, 7, 8, 9],

[ 3, 4, 5, 10, 11, 12]])

**np.dstack((a,b))**

**o/p:** array([[[ 0, 7],

[ 1, 8],

[ 2, 9]],

[[ 3, 10],

[ 4, 11],

[ 5, 12]]])

**Arithematic operations in numpy:**

**Addition subtraction, ,multiplication,division,modulo power,reciprocal,complex numbers**

**Rules for implementing arithematic operations**

* **shapes of two arrays must be equal(np.shape(a))**

Ex:a=[1,2],[3,4] ->(2,2)

b=[5,6],[7,8] ->(2,2) so we can add

* **second array should have atleast one dimension and the number of elements dimension should b equal to first array**

**Ex:** a=[1,2],[3,4] ->(2,2)

b=[5,6] ->(2,) then the interpreter take it as

b=[5,6],[5,6] then they will add

* **Second array having single elements**

Ex: a=[1,2],[3,4] ->(2,2)

b=10 -> then the interpreter take it as

b=[10,10],[10,10] then they will add

**Addition: syntax: np.add(first-array,second array)**

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1,2,3,4,5])

c=a+b

print(c)

o/p:

[11 22 33 44 55]

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1.3,3.2,4.6,5.8,9.4])

c=a+b

print(c)

o/p:

[11.3 23.2 34.6 45.8 59.4]

import numpy as np

a=np.array([10.6,20.1,30.5,40.7,50.3])

b=np.array([1.4,2.6,3.8,4.9,5.0])

c=a+b

print(c)

o/p:

[12. 22.7 34.3 45.6 55.3]

import numpy as np

a=np.array([10,20,30,40])

b=np.array([2])

c=a+b

print(c)

o/p:

[12 22 32 42]

import numpy as np

a=np.array([[10,20],[30,40]])

b=np.array([50,30])

c=a+b

print(c)

o/p:

[[60 50]

[80 70]]

**Subtraction: Syntax: np.subtract(first-array,second array)**

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1,2,3,4,5])

c=a-b

print(c)

o/p:

[ 9 18 27 36 45]

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1.3,3.2,4.6,5.8,9.4])

c=a-b

print(c)

o/p:

[ 8.7 16.8 25.4 34.2 40.6]

import numpy as np

a=np.array([10.6,20.1,30.5,40.7,50.3])

b=np.array([1.4,2.6,3.8,4.9,5.0])

c=a-b

print(c)

o/p:

[ 9.2 17.5 26.7 35.8 45.3]

import numpy as np

a=np.array([10,20,30,40])

b=np.array([2])

c=a-b

print(c)

o/p:

[ 8 18 28 38]

import numpy as np

a=np.array([[10,20],[30,40]])

b=np.array([50,30])

c=a-b

print(c)

o/p:

[[-40 -10]

[-20 10]]

**Multiply: Syntax: np.multiply(first-array,second array)**

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1,2,3,4,5])

c=a\*b

print(c)

o/p:

[ 10 40 90 160 250]

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1.3,3.2,4.6,5.8,9.4])

c=a\*b

print(c)

o/p:

[ 13. 64. 138. 232. 470.]

import numpy as np

a=np.array([10.6,20.1,30.5,40.7,50.3])

b=np.array([1.4,2.6,3.8,4.9,5.0])

c=a\*b

print(c)

o/p:

[ 14.84 52.26 115.9 199.43 251.5 ]

import numpy as np

a=np.array([10,20,30,40])

b=np.array([2])

c=a\*b

print(c)

o/p:

[20 40 60 80]

import numpy as np

a=np.array([[10,20],[30,40]])

b=np.array([50,30])

c=a\*b

print(c)

o/p:

[[ 500 600]

[1500 1200]]

**Division: Syntax: np.divide(first-array,second array)**

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1,2,3,4,5])

c=a/b

print(c)

o/p:

[10. 10. 10. 10. 10.]

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1.3,3.2,4.6,5.8,9.4])

c=a/b

print(c)

o/p:

[7.69230769 6.25 6.52173913 6.89655172 5.31914894]

import numpy as np

a=np.array([10.6,20.1,30.5,40.7,50.3])

b=np.array([1.4,2.6,3.8,4.9,5.0])

c=a/b

print(c)

o/p:

[ 7.57142857 7.73076923 8.02631579 8.30612245 10.06 ]

import numpy as np

a=np.array([10,20,30,40])

b=np.array([2])

c=a/b

print(c)

o/p:

[ 5. 10. 15. 20.]

import numpy as np

a=np.array([[10,20],[30,40]])

b=np.array([50,30])

c=a/b

print(c)

o/p:

[[0.2 0.66666667]

[0.6 1.33333333]]

**Modulo: Syntax: np.mod(first-array,second array)**

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1,2,3,4,5])

c=np.mod(a,b)

print(c)

o/p:

[0 0 0 0 0]

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1.3,3.2,4.6,5.8,9.4])

c=np.mod(a,b)

print(c)

o/p:

[0.9 0.8 2.4 5.2 3. ]

import numpy as np

a=np.array([10.6,20.1,30.5,40.7,50.3])

b=np.array([1.4,2.6,3.8,4.9,5.0])

c=np.mod(a,b)

print(c)

o/p:

[0.8 1.9 0.1 1.5 0.3]

import numpy as np

a=np.array([10,20,30,40])

b=np.array([2])

c=np.mod(a,b)

print(c)

o/p:

[0 0 0 0]

import numpy as np

a=np.array([[10,20],[30,40]])

b=np.array([50,30])

c=np.mod(a,b)

print(c)

o/p:

[[10 20]

[30 10]]

​

**Power: Syntax: np.power(first-array,second array)**

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1,2,3,4,5])

c=np.power(a,b)

print(c)

o/p:

[ 10 400 27000 2560000 312500000]

import numpy as np

a=np.array([10,20,30,40,50])

b=np.array([1.3,3.2,4.6,5.8,9.4])

c=np.power(a,b)

print(c)

o/p:

[1.99526231e+01 1.45645136e+04 6.23387044e+06 1.95860992e+09

9.33937988e+15]

import numpy as np

a=np.array([10.6,20.1,30.5,40.7,50.3])

b=np.array([1.4,2.6,3.8,4.9,5.0])

c=np.power(a,b)

print(c)

o/p:

[2.72538720e+01 2.44517389e+03 4.36857091e+05 7.70925845e+07

3.21988177e+08]

import numpy as np

a=np.array([10,20,30,40])

b=np.array([2])

c=np.power(a,b)

print(c)

o/p:

[ 100 400 900 1600]

**STATISTICAL OPERATIONS:** np.min,np.max,np.mean,np.median,np.var,np.std

**np.min:**to find the minimum element in the array

Ex: import numpy as np

a=([[10,30,60],[3,4,5],[6,7,8]])

print(a)

o/p: [[10, 30, 60], [3, 4, 5], [6, 7, 8]]

np.amin(a)

o/p: 3

np.amin(a,axis=0)

o/p: array([3, 4, 5])

np.amin(a,axis=1)

o/p: array([10, 3, 6])

np.amax(a)

o/p: 60

np.amax(a,axis=0)

o/p array([10, 30, 60])

np.amax(a,axis=1)

o/p: array([60, 5, 8])

**np.mean**: It is calculated by adding all elements in the array and then dividing the result by the total number of elements in the array.

We use the np.mean() function to calculate the mean value

Ex: import numpy as np

a=([[10,30,60],[3,4,5],[6,7,8]])

np.mean(a)

o/p: 14.777777777777779

np.mean(a,axis=0)

o/p: array([ 6.33333333, 13.66666667, 24.33333333])

np.mean(a,axis=1)

o/p: array([33.33333333, 4. , 7. ])

**np.median**: To find the middle element

Ex: import numpy as np

a=([[10,30,60],[3,4,5],[6,7,8]])

np.median(a)

o/p: 7.0

np.median(a,axis=0)

o/p: array([6., 7., 8.])

np.median(a,axis=1)

o/p: array([30., 4., 7.])

**np.std:** standard deviation can be calculated either along the horizontal or the vertical axis individually, or across the entire array.

use the axis parameter inside np.std()

\\Ex: we are taking the array elements as

a=([[10,30,60],[3,4,5],[6,7,8]])

now we have to find mean to these elements

**o/p:** 7.0

now calculate the variance[variance can be calculated as value-mean value square (10-7) divided by total number of values]

np.var(a)

std=sruareroot(variance)\\

**string operations in numpy:lower,split,join**

import numpy as np

a=np.char.upper([“this is NRI”])

print(a)

o/p:

['THIS IS NRI']

import numpy as np

a=np.char.lower(['THIS IS NRI'])

print(a)

o/p:

['this is nri']

import numpy as np

a=np.char.split(['this is NRI'])

print(a)

o/p:

[list(['this', 'is', 'NRI'])]

import numpy as np

a = np.array(['this','is','NRI'])

b = np.array(['\*\*\*\*\*'])

c = np.char.join(a,b)

print(c)

o/p:

['\*this\*this\*this\*this\*' '\*is\*is\*is\*is\*' '\*NRI\*NRI\*NRI\*NRI\*']

import numpy as np

d = np.array(['Apple', 'Python', 'NumPy','StudyTonight'])

e = np.array(['^^^'])

output= np.char.join(d, e)

print(output)

o/p:

['^Apple^Apple^' '^Python^Python^' '^NumPy^NumPy^'

'^StudyTonight^StudyTonight^']