Theory

NumPy

is fundamental package for scientifc computing in Python.

short form of NUmerical python

It is Python library that provides a multidimensional array object, various derived objects (masked arrays and matrices),

supports fast operations on arrays,

including

mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation andmore.

NumPy supports Object Oriented Approach

Open source , community built functions

Why is NumPy Fast?

Vectorization means absence of explicit looping, indexing, etc.,

in code - these things are taking place, of course, just “behind the scenes” in optimized, pre-compiled C code.

Vectorized code : advantages

• more concise & easier to read

• fewer lines of code means fewer bugs

• code resembles standard mathematical notation

• more “Pythonic” code.

- no typecheck unlike in python list
- DROM burst leveraged in NumPy

Without vectorization

code would be littered with inefficient and

difficult to read for loop

NumPy+

easy file handling

can be accesible by other packages

```
import numpy as np
```

```
%%time
#list to multiple 10000 numbers
#list comprehension --
#lambda function
#np multiplication

#time reduces
```

Creating an array

```
ar = np.array(1,2,3) #note one more braces is needed inside
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-1dc4f885b91d> in <module>()
----> 1 ar = np.array(1,2,3)

TypeError: array() takes from 1 to 2 positional arguments but 3 were given
```

SEARCH STACK OVERFLOW

```
ar1 = np.array((1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0))
ar1
```

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

```
#or use list []
ar1 = np.array([1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0,10.0])
ar1
```

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

```
ar2 = np.array([1,2],[3,4],[5,6]) #braces
ar2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-33-66c775d27180> in <module>()
----> 1 ar2 = np.array([1,2],[3,4],[5,6])
      2 ar2

TypeError: array() takes from 1 to 2 positional arguments but 3 were given
```

SEARCH STACK OVERFLOW

```
ar2 = np.array(
                (
                [1,2],
                [3,4],
                [5,6]
                    )
                        ) #braces
ar2
#no of () 1d array , ((2d array)) , (((3d array)))
```

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

```
ar3 = np.array(
                (
                    ([1,2],[3,4],[5,6])
                    ,
                    ([1,2],[3,4],[5,6])
                                                )
                                                    ) #observe braces
ar3 #3d array
```

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

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

```
#The type of the array can also be explicitly specifed at creation time
c = np.array([[1, 2], [3, 4]], dtype=float)
c
```

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

```
np.zeros(3,3,3) #braces error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-94-87e28da24389> in <module>()
----> 1 np.zeros(3,3,3)

TypeError: Cannot interpret '3' as a data type
```

SEARCH STACK OVERFLOW

```
np.zeros((3,3,3))
```

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

       [[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]]])
```

```
np.zeros((0,3,3))
```

```
array([], shape=(0, 3, 3), dtype=float64)
```

```
np.zeros(10) #creates null vector of size 10
```

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

```
np.ones((0,3,3))
```

```
array([], shape=(0, 3, 3), dtype=float64)
```

```
print(np.ones((0,3,3)))
```

```
[]
```

```
np.ones((3,3))
```

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

```
2022*np.ones((3,3))
```

```
array([[2022., 2022., 2022.],
       [2022., 2022., 2022.],
       [2022., 2022., 2022.]])
```

```
np.ones((3,3,3))
```

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

       [[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]],

       [[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]]])
```

```
np.empty((3,3)) #Return new array of given shape and type, without initializing entrie
```

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

```
np.empty((2,2))
```

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

```
np.empty([2,2])
```

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

```
np.arange(10)
```

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

```
np.arange(10,30) #note 30 wont be printed
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29])
```

```
np.arange(10,30,5)
```

```
array([10, 15, 20, 25])
```

```
np.linspace(10,30,5) #5 - total numbers of entries i want
```

```
array([10., 15., 20., 25., 30.])
```

```
np.array((True,False))
```

```
array([ True, False])
```

```
np.array(["0.1","False"])
```

```
array(['0.1', 'False'], dtype='<U5')
```

```
np.repeat(1,10)
```

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

```
np.eye(3,3) #identity matrix
```

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

Create random arrays

```
np.random.random(3,3) #braces error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-264-15adae4bb9f6> in <module>()
----> 1 np.random.random(3,3)

mtrand.pyx in numpy.random.mtrand.RandomState.random()

TypeError: random() takes at most 1 positional argument (2 given)
```

SEARCH STACK OVERFLOW

```
np.random.random((3,3))
```

```
array([[0.18239202, 0.96163452, 0.13060079],
       [0.26402307, 0.26142463, 0.42977447],
       [0.82869696, 0.60825607, 0.55931551]])
```

```
np.random.rand(2,3) #0-1
```

```
array([[0.12846604, 0.93787524, 0.33340694],
       [0.79302825, 0.05842871, 0.82087029]])
```

```
np.random.randn(2,3) #-1 to +1 sir said
```

```
array([[1.04111123, 0.28110752, 1.90942781],
       [0.7773684 , 0.98799932, 1.51758331]])
```

```
np.random.randint(0,100,(2,3))
```

```
array([[57,  3, 38],
       [59, 97, 25]])
```

```
np.random.uniform(-100,+100,10)
```

```
array([-51.5989774 ,  -6.45928098, -46.95957337,  51.33980491,
        -2.80115962, -49.66099706, -96.23401238,  49.81837101,
        39.4549169 , -88.99665817])
```

Basic Attributes

```
ar1.shape
```

```
(10,)
```

```
ar2.shape
```

```
(3, 2)
```

```
ar3.shape
```

```
(2, 3, 2)
```

```
ar1.ndim
```

```
1
```

```
ar2.ndim
```

```
2
```

```
ar3.ndim
```

```
3
```

```
ar3.size #total elements in array
```

```
12
```

```
type(ar3)
```

```
numpy.ndarray
```

```
ar3.dtype
```

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

```
ar3.itemsize # size in bytes of each element of the array.
```

```
8
```

```
ar3.data
```

```
<memory at 0x7f5e08fa9350>
```

TypeCasting an array

```
ar1.astype(dtype=int)
```

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

```
#alternate
np.array(ar1,dtype="float")
```

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

Indexing

```
#(table ind ,row ind ,column ind )
# : if want all table / row /col
#start , stop , stop value not includes
```

Slicing

```
cric = np.array([167.,  31.,  54., 314.],[168.,   2.,  52., 419.],[169.,  44.,   9., 22
cric
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-aaf8e004657c> in <module>()
----> 1 cric = np.array([167.,  31.,  54., 314.],[168.,   2.,  52., 419.],[169.,
44.,   9., 229.])
      2 cric

TypeError: array() takes from 1 to 2 positional arguments but 3 were given
```

SEARCH STACK OVERFLOW

```
cric = np.array(([167.,  31.,  54., 314.],[168.,   2.,  52., 419.],[169.,  44.,   9., 2
cric
```

```
array([[167.,  31.,  54., 314.],
       [168.,   2.,  52., 419.],
       [169.,  44.,   9., 229.]])
```

```
#to ignore 1st column
cric[:,1:]
```

```
array([[ 31.,  54., 314.],
       [  2.,  52., 419.],
       [ 44.,   9., 229.]])
```

```
cric[-1] # last row just like  cric[-1, :]
```

```
array([169.,  44.,   9., 229.])
```

Reshape

```
np.arange(9)
```

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

```
np.arange(9).reshape(3,3)
```

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

```
np.arange(9).reshape((3,3))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-197-54bf90d58ebc> in <module>()
----> 1 np.concatenate(a1,a2)

<__array_function__ internals> in concatenate(*args, **kwargs)

TypeError: only integer scalar arrays can be converted to a scalar index
```

SEARCH STACK OVERFLOW

```
#reshape function returns its argument with a modifed shape,
# whereas the ndarray.resize method modifes the array itself:
#both are same but resize permanently
```

```
ar2
```

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

```
ar2.reshape(2,-1) # -1 means "whatever is needed"
```

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

```
ar2
```

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

```
ar2.T #Transpose
```

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

```
#hsplit, you can split an array along its horizontal axis, eithe
np.hsplit(ar2,2) #2 parition of array
```

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

```
np.vsplit(ar2,3) #3 parition of array
```

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

```
ar1
```

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

```
ar1[::-1] #reverse 1d array
```

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

Broadcasting

```
#2 arrays should be same size
# for addition also
#2 table+1 table = good where same row and col
np.random.rand((2,3,3))#no need of inner brace
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-161-9bd5e0af0e24> in <module>()
      2 # for addition also
      3 #2 table+1 table = good where same row and col
----> 4 np.random.rand((2,3,3))

mtrand.pyx in numpy.random.mtrand.RandomState.rand()

mtrand.pyx in numpy.random.mtrand.RandomState.random_sample()

_common.pyx in numpy.random._common.double_fill()

TypeError: 'tuple' object cannot be interpreted as an integer
```

SEARCH STACK OVERFLOW

```
np.random.rand(2,3,4)+np.random.rand(3,1) #works #not index but col row tab num
```

```
array([[[1.38337503, 0.92104383, 1.30026019, 1.74554921],
        [1.34742629, 1.61639976, 1.31633756, 1.57476553],
        [0.96192882, 1.16692681, 1.22429914, 1.66438088]],
       [[1.72842948, 1.87069918, 1.11385654, 1.61821935],
        [1.84129349, 1.66125648, 1.93846892, 1.16821229],
        [1.42208846, 1.01222946, 1.02517282, 1.12147675]]])
```

```
np.random.rand(2,3,4)+np.random.rand(1,4)
```

```
array([[[0.91610125, 1.5909772 , 1.14970065, 0.7455476 ],
        [1.48363786, 1.58779516, 1.38226467, 0.93291797],
        [1.63775082, 1.49375592, 1.3469581 , 1.49020003]],

       [[1.11339847, 1.63749186, 1.87015756, 0.68456399],
        [0.81343381, 0.99967744, 1.97249002, 1.19952342],
        [1.42990593, 1.4879084 , 1.13149066, 0.80363712]]])
```

```
np.random.rand(2,3,4)+np.random.rand(1)
```

```
array([[[1.06297095, 0.90597033, 0.75585172, 0.17905719],
        [0.1308205 , 0.23535766, 0.40914715, 1.0614186 ],
        [0.96394111, 0.31318313, 0.59386724, 0.31467717]],

       [[0.27936853, 1.01181072, 0.34629117, 1.00331491],
        [0.66012042, 0.22278927, 0.6453232 , 0.77615768],
        [0.93959176, 0.37389353, 0.41305393, 0.60552345]]])
```

```
np.random.rand(1,4)+np.random.rand(1,5)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-174-3f15a5d43b94> in <module>()
----> 1 np.random.rand(1,4)+np.random.rand(1,5)

ValueError: operands could not be broadcast together with shapes (1,4) (1,5)
```

SEARCH STACK OVERFLOW

```
np.random.rand(1,4)+np.random.rand(5,1)
```

```
array([[1.368971  , 1.78149733, 1.74188825, 1.69866021],
       [1.23142145, 1.64394778, 1.6043387 , 1.56111065],
       [1.35287451, 1.76540083, 1.72579175, 1.68256371],
       [0.92488686, 1.33741318, 1.2978041 , 1.25457606],
       [0.52655167, 0.93907799, 0.89946892, 0.85624087]])
```

```
np.random.rand(4,1)+np.random.rand(5,1)
```

```
---------------------------------------------------------------------------
```

```
np.random.rand(4,1)+np.arange(16)
```

```
array([[ 0.79521136,  1.79521136,  2.79521136,  3.79521136,  4.79521136,
         5.79521136,  6.79521136,  7.79521136,  8.79521136,  9.79521136,
        10.79521136, 11.79521136, 12.79521136, 13.79521136, 14.79521136,
        15.79521136],
       [ 0.1296321 ,  1.1296321 ,  2.1296321 ,  3.1296321 ,  4.1296321 ,
         5.1296321 ,  6.1296321 ,  7.1296321 ,  8.1296321 ,  9.1296321 ,
        10.1296321 , 11.1296321 , 12.1296321 , 13.1296321 , 14.1296321 ,
        15.1296321 ],
       [ 0.06760031,  1.06760031,  2.06760031,  3.06760031,  4.06760031,
         5.06760031,  6.06760031,  7.06760031,  8.06760031,  9.06760031,
        10.06760031, 11.06760031, 12.06760031, 13.06760031, 14.06760031,
        15.06760031],
       [ 0.53000106,  1.53000106,  2.53000106,  3.53000106,  4.53000106,
         5.53000106,  6.53000106,  7.53000106,  8.53000106,  9.53000106,
        10.53000106, 11.53000106, 12.53000106, 13.53000106, 14.53000106,
        15.53000106]])
```

```
np.random.rand(4,1)+(np.arange(16).reshape(4,4))
```

```
array([[ 0.93228619,  1.93228619,  2.93228619,  3.93228619],
       [ 4.33074554,  5.33074554,  6.33074554,  7.33074554],
       [ 8.14580235,  9.14580235, 10.14580235, 11.14580235],
       [12.44067822, 13.44067822, 14.44067822, 15.44067822]])
```

Accessing elements selectively

```
cric
```

```
array([[167.,  31.,  54., 314.],
       [168.,   2.,  52., 419.],
       [169.,  44.,   9., 229.]])
```

```
cric[cric>150]
```

```
array([167., 314., 168., 419., 169., 229.])
```

Comparing 2 arrays and accesing

```
a1=np.array[1,2,3,4,5,6,7,8]
a2=np.array[8,7,6,5,4,3,2,1] #BRACES error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-43-db5aab7e80ad> in <module>()
----> 1 a1=np.array[1,2,3,4,5,6,7,8]
```

```
a1=np.array([1,2,3,4,5,6,7,8])
a2=np.array([8,7,6,5,4,3,2,1])
```

```
a2<5
```

```
array([False, False, False, False,  True,  True,  True,  True])
```

Fancy indexing

```
#means passing an array of indices to access multiple array elements at once
a2[a2%2==0]
```

```
array([8, 6, 4, 2])
```

```
cric[(cric%2==0)&(cric>150)]
```

```
array([314., 168.])
```

Mathematical funcions

```
a2
```

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

```
#inverse
1/a2
```

```
array([0.125     , 0.14285714, 0.16666667, 0.2       , 0.25      ,
       0.33333333, 0.5       , 1.        ])
```

```
np.isinf(ar2)#checks if has infinite element
```

```
array([[False, False],
       [False, False],
       [False, False]])
```

```
#common elements bw 2 array
np.intersect1d(a1,a2)
```

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

```
a2+np.array([1,2,3]) #both should be similar size or possible for broadcasting reshape
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-151-8c5be801b576> in <module>()
----> 1 a2+np.array([1,2,3])

ValueError: operands could not be broadcast together with shapes (8,) (3,)
```

SEARCH STACK OVERFLOW

```
a2+np.array([1,2]) #both should be similar size or possible for broadcasting reshape
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-152-ad966d205f19> in <module>()
----> 1 a2+np.array([1,2]) #both should be similar size or possible for broadcasting
reshape

ValueError: operands could not be broadcast together with shapes (8,) (2,)
```

SEARCH STACK OVERFLOW

```
np.sort(a2)
```

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

```
np.cumsum(a2)
```

```
array([ 8, 15, 21, 26, 30, 33, 35, 36])
```

```
np.sqrt(a2)
```

```
array([2.82842712, 2.64575131, 2.44948974, 2.23606798, 2.        ,
       1.73205081, 1.41421356, 1.        ])
```

```
np.sin(a2)
```

```
array([ 0.98935825,  0.6569866 , -0.2794155 , -0.95892427, -0.7568025 ,
        0.14112001,  0.90929743,  0.84147098])
```

```
np.exp(a2)
```

```
array([2.98095799e+03, 1.09663316e+03, 4.03428793e+02, 1.48413159e+02,
       5.45981500e+01, 2.00855369e+01, 7.38905610e+00, 2.71828183e+00])
```

```
np.log(a2)
```

```
array([2.07944154, 1.94591015, 1.79175947, 1.60943791, 1.38629436,
       1.09861229, 0.69314718, 0.        ])
```

```
A = np.array([[1, 1],[0, 1]])
```

```
B = np.array([[2, 0],[3, 4]])
A
```

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

```
B
```

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

```
#B = A changes original A / value also but
# not copy , B change in A
np.copy(B)
```

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

```
A+B
```

```
array([[3, 1],
       [3, 5]])
```

```
#or
np.add(A,B)
```

```
array([[3, 1],
       [3, 5]])
```

```
A*B #elementwise one - one product
```

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

```
A@B #matrix product
```

```
array([[5, 4],
       [3, 4]])
```

```
A.dot(B) #matrix product
```

```
array([[5, 4],
       [3, 4]])
```

```
np.arange(10)**3 #CUBE OF NUMBERS FROM 0-9
```

```
array([  0,   1,   8,  27,  64, 125, 216, 343, 512, 729])
```

```
#CUBE ROOT
#a1**(1 / 3))
```

Statistical Functions

```
ar1
```

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

```
np.min(ar1)
```

```
1.0
```

```
np.amin(ar1)
```

```
1.0
```

```
np.max(ar1)
```

```
10.0
```

```
np.sum(ar1)
```

```
55.0
```

```
np.mean(ar1) #mean for 1D array
```

```
5.5
```

```
cric
```

```
array([[167.,  31.,  54., 314.],
       [168.,   2.,  52., 419.],
       [169.,  44.,   9., 229.]])
```

```
np.mean(cric,axis=0) #Mean COLUMNwise
```

```
array([168.        ,  25.66666667,  38.33333333, 320.66666667])
```

```
np.mean(cric,axis=1) #Mean ROWwise
```

```
array([141.5 , 160.25, 112.75])
```

```
np.median(cric)
```

```
110.5
```

```
ar1
```

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

```
np.percentile(ar1,75)
```

```
7.75
```

```
np.percentile(ar1,[25,50,75]) #multiple percentile values
```

```
array([3.25, 5.5 , 7.75])
```

```
np.histogram(ar1,bins=5) #can be used to , find frequency table
```

```
(array([2, 2, 2, 2, 2]), array([ 1. ,  2.8,  4.6,  6.4,  8.2, 10. ]))
```

```
np.histogram(ar1,bins=[0,3,6,9])
```

```
(array([2, 3, 4]), array([0, 3, 6, 9]))
```

```
np.histogram(ar1,bins=range(0,10,4))
```

```
(array([3, 5]), array([0, 4, 8]))
```

```
ar1
```

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

```
np.digitize(ar1,bins=[0,2,4,6,8])
```

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

```
ar2a=np.array([[1, 2],[6,3],[4,5]])
ar2a
```

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

```
a1
```

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

```
np.argmax(a1) #index of max value in a1 array
```

```
7
```

```
#if else , if value 1 , then One , else Zero
np.where(a1==1,"One","Zero")
```

```
array(['One', 'Zero', 'Zero', 'Zero', 'Zero', 'Zero', 'Zero', 'Zero'],
      dtype='<U4')
```

```
#compare row wise
np.argmax(ar2a,axis=0)
#in 0th index col , 1st index row has max value
#in 1th index col , 2nd index row has max value
```

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

```
#compare columnwise
np.argmax(ar2a,axis=1)
#in 0th index row , 1st index col has max value
#in 1st index row , 0st index col has max value
#in 2nd index row , 1st index col has max value
```

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

```
a1a=np.array([0,1,0,2,3,0,0,4])
a1a
```

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

```
np.count_nonzero(a1a) #counts other than 0
```

```
4
```

```
np.nonzero(a1a) #return argument / indeces of non zero
```

```
(array([1, 3, 4, 7]),)
```

```
a1=np.array([1,2,3,4,5,6,7,8])
a2=np.array([8,7,6,5,4,3,2,1])
```

```
np.concatenate((a1,a2))
```

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

```
np.hstack((a1,a2))
#hstack same as columnstack
```

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

```
ar33=np.vstack((a1,a2))
ar33
#hstack same as rowstack
```

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

```
ar3 = np.random.randint(5,100,7)
```

```
ar3
```

```
array([23, 36, 79, 70, 96, 89, 75])
```

```
np.amin(ar33,axis=0) #columnwise
```

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

```
np.amin(ar33,axis=1) #row wise
```

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

```
a1
```

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

```
np.var(a1)
```

```
5.25
```

```
np.std(a1)
```

```
2.29128784747792
```

```
zscore = ((a1-np.mean(a1))
                             /
                                  (np.std(a1)))
zscore
```

```
array([-1.52752523, -1.09108945, -0.65465367, -0.21821789,  0.21821789,
        0.65465367,  1.09108945,  1.52752523])
```

```
np.append(a1,[1000,2000])
```

```
array([   1,    2,    3,    4,    5,    6,    7,    8, 1000, 2000])
```

```
# ix_ function can be used to
# combine different vectors so as to obtain the result for each n-uplet.
```

File Handling in NumPy

```
#upload file to folder , can run this before loadtxt
!head cric_data.tsv
```

```
          Sachin Tendulkar        Rahul Dravid    India
0         100     78      342
1         11      62      191
2         8       85      252
```

```
3       71      24      307
4       104     17      229
5       18      104     246
6       8       76      226
7       86      74      288
8       12      60      216
```

```
cric_data = np.loadtxt("cric_data.tsv",skiprows=1)
cric_data
```

```
array([[  0., 100.,  78., 342.],
       [  1.,  11.,  62., 191.],
       [  2.,   8.,  85., 252.],
       [  3.,  71.,  24., 307.],
       [  4., 104.,  17., 229.],
       [  5.,  18., 104., 246.],
       [  6.,   8.,  76., 226.],
       [  7.,  86.,  74., 288.],
       [  8.,  12.,  60., 216.],
       [  9.,  85.,  12., 224.],
       [ 10.,  18.,  63., 161.],
       [ 11.,   4., 107., 276.],
       [ 12.,   7.,  76., 283.],
       [ 13.,  37.,   4., 297.],
       [ 14.,  14.,   5., 139.],
       [ 15.,   0.,  33., 224.],
       [ 16.,   4.,   7., 178.],
       [ 17.,   0.,   0.,   0.],
       [ 18.,  21.,  36., 193.],
       [ 19.,   1.,  66., 231.],
       [ 20.,  62.,   0., 134.],
       [ 21.,   0., 123., 246.],
       [ 22., 138.,  39., 299.],
       [ 23.,  38.,   9., 242.],
       [ 24.,   2.,  11., 214.],
       [ 25.,  46.,  14., 152.],
       [ 26.,  65.,   0., 104.],
       [ 27.,   0.,   0.,   4.],
       [ 28.,  39.,  26., 155.],
       [ 29.,  48.,   4., 168.],
       [ 30., 141.,  48., 282.],
       [ 31.,  62.,   7., 228.],
       [ 32.,  12.,  73., 231.],
       [ 33.,   1.,  86., 238.],
       [ 34.,  41.,  32., 255.],
       [ 35.,  11.,  82., 273.],
       [ 36.,   3.,  25., 143.],
       [ 37., 186., 153., 345.],
       [ 38.,  11.,  26., 134.],
       [ 39.,  27.,   1., 292.],
       [ 40.,  27.,   6., 299.],
       [ 41.,  51.,   3., 233.],
       [ 42.,  18.,   1., 332.],
       [ 43.,  32.,  39., 276.],
       [ 44., 146.,  30., 264.],
       [ 45.,   5.,  32., 213.],
       [ 46.,  45.,  84., 224.],
       [ 47., 141.,  36., 306.],
       [ 48.,  12.,  31., 259.],
       [ 49.,  65.,   0., 141.],
       [ 50.,  27.,  47., 155.],
       [ 51.,   7.,  13., 183.],
       [ 52.,  16.,  49., 309.],
       [ 53.,   2.,  28., 208.],
       [ 54.,  28.,   0., 124.],
       [ 55.,   6.,  28., 208.],
       [ 56., 123.,  19., 305.],
       [ 57., 120.,  13., 273.],
```

```
cric_data = np.loadtxt("cric_data.tsv",skiprows=1,usecols=[1,2,3])
```

```
cric_data
```

```
    array([[100.,  78., 342.],
           [ 11.,  62., 191.],
           [  8.,  85., 252.],
           [ 71.,  24., 307.],
           [104.,  17., 229.],
           [ 18., 104., 246.],
           [  8.,  76., 226.],
           [ 86.,  74., 288.],
           [ 12.,  60., 216.],
           [ 85.,  12., 224.],
           [ 18.,  63., 161.],
           [  4., 107., 276.],
           [  7.,  76., 283.],
           [ 37.,   4., 297.],
           [ 14.,   5., 139.],
           [  0.,  33., 224.],
           [  4.,   7., 178.],
           [  0.,   0.,   0.],
           [ 21.,  36., 193.],
           [  1.,  66., 231.],
           [ 62.,   0., 134.],
           [  0., 123., 246.],
           [138.,  39., 299.],
           [ 38.,   9., 242.],
           [  2.,  11., 214.],
           [ 46.,  14., 152.],
           [ 65.,   0., 104.],
           [  0.,   0.,   4.],
           [ 39.,  26., 155.],
           [ 48.,   4., 168.],
           [141.,  48., 282.],
           [ 62.,   7., 228.],
           [ 12.,  73., 231.],
           [  1.,  86., 238.],
           [ 41.,  32., 255.],
           [ 11.,  82., 273.],
           [  3.,  25., 143.],
           [186., 153., 345.],
           [ 11.,  26., 134.],
           [ 27.,   1., 292.],
           [ 27.,   6., 299.],
           [ 51.,   3., 233.],
           [ 18.,   1., 332.],
           [ 32.,  39., 276.],
           [146.,  30., 264.],
           [  5.,  32., 213.],
           [ 45.,  84., 224.],
           [141.,  36., 306.],
           [ 12.,  31., 259.],
           [ 65.,   0., 141.],
           [ 27.,  47., 155.],
           [  7.,  13., 183.],
           [ 16.,  49., 309.],
           [  2.,  28., 208.],
           [ 28.,   0., 124.],
           [  6.,  28., 208.],
           [123.,  19., 305.],
```

```
np.genfromtxt("cric_data.tsv",skip_header=1,usecols=[1,2,3])
#to overcome any string issue in
```

```
    array([[100.,  78., 342.],
           [ 11.,  62., 191.],
           [  8.,  85., 252.],
           [ 71.,  24., 307.],
           [104.,  17., 229.],
           [ 18., 104., 246.],
           [  8.,  76., 226.],
           [ 86.,  74., 288.],
           [ 12.,  60., 216.],
           [ 85.,  12., 224.],
           [ 18.,  63., 161.],
           [  4., 107., 276.],
           [  7.,  76., 283.],
           [ 37.,   4., 297.],
           [ 14.,   5., 139.],
           [  0.,  33., 224.],
           [  4.,   7., 178.],
           [  0.,   0.,   0.],
           [ 21.,  36., 193.],
           [  1.,  66., 231.],
           [ 62.,   0., 134.],
           [  0., 123., 246.],
           [138.,  39., 299.],
           [ 38.,   9., 242.],
           [  2.,  11., 214.],
           [ 46.,  14., 152.],
           [ 65.,   0., 104.],
           [  0.,   0.,   4.],
           [ 39.,  26., 155.],
           [ 48.,   4., 168.],
           [141.,  48., 282.],
           [ 62.,   7., 228.],
           [ 12.,  73., 231.],
           [  1.,  86., 238.],
           [ 41.,  32., 255.],
           [ 11.,  82., 273.],
           [  3.,  25., 143.],
           [186., 153., 345.],
           [ 11.,  26., 134.],
           [ 27.,   1., 292.],
           [ 27.,   6., 299.],
           [ 51.,   3., 233.],
           [ 18.,   1., 332.],
           [ 32.,  39., 276.],
           [146.,  30., 264.],
           [  5.,  32., 213.],
           [ 45.,  84., 224.],
           [141.,  36., 306.],
           [ 12.,  31., 259.],
           [ 65.,   0., 141.],
           [ 27.,  47., 155.],
           [  7.,  13., 183.],
           [ 16.,  49., 309.],
           [  2.,  28., 208.],
           [ 28.,   0., 124.],
           [  6.,  28., 208.],
           [123.,  19., 305.],
           [120.,  13., 273.],
```

```
np.nan_to_num(cric_data,nan=1)
```

```
    array([[100.,  78., 342.],
           [ 11.,  62., 191.],
           [  8.,  85., 252.],
           [ 71.,  24., 307.],
           [104.,  17., 229.],
           [ 18., 104., 246.],
           [  8.,  76., 226.],
           [ 86.,  74., 288.],
           [ 12.,  60., 216.],
           [ 85.,  12., 224.],
           [ 18.,  63., 161.],
           [  4., 107., 276.],
           [  7.,  76., 283.],
           [ 37.,   4., 297.],
           [ 14.,   5., 139.],
           [  0.,  33., 224.],
           [  4.,   7., 178.],
           [  0.,   0.,   0.],
           [ 21.,  36., 193.],
           [  1.,  66., 231.],
           [ 62.,   0., 134.],
           [  0., 123., 246.],
           [138.,  39., 299.],
           [ 38.,   9., 242.],
           [  2.,  11., 214.],
           [ 46.,  14., 152.],
           [ 65.,   0., 104.],
           [  0.,   0.,   4.],
           [ 39.,  26., 155.],
           [ 48.,   4., 168.],
           [141.,  48., 282.],
           [ 62.,   7., 228.],
           [ 12.,  73., 231.],
           [  1.,  86., 238.],
           [ 41.,  32., 255.],
           [ 11.,  82., 273.],
           [  3.,  25., 143.],
           [186., 153., 345.],
           [ 11.,  26., 134.],
           [ 27.,   1., 292.],
           [ 27.,   6., 299.],
           [ 51.,   3., 233.],
           [ 18.,   1., 332.],
           [ 32.,  39., 276.],
           [146.,  30., 264.],
           [  5.,  32., 213.],
           [ 45.,  84., 224.],
           [141.,  36., 306.],
           [ 12.,  31., 259.],
           [ 65.,   0., 141.],
           [ 27.,  47., 155.],
           [  7.,  13., 183.],
           [ 16.,  49., 309.],
           [  2.,  28., 208.],
           [ 28.,   0., 124.],
           [  6.,  28., 208.],
           [123.,  19., 305.],
           [120.,  13., 273.],
```

```
np.save("Planets_new",cric)
```

```
np.savetxt("Planets_new",cric)
```

```
np.savez("Planets_new",a1,a2)
```

```
np.savez_compressed("Planets_new",a1,a2)
```

```

```

Miscellaneous

```
Ndim=2
Npoints=100000
Points = np.random.rand(Npoints,Ndim)
dfo = np.zeros(Npoints,1) #distancr from zeros , #brace error here
OutsidePoints=0
for i in range(Npoints) :
    for j in range(Ndim) :
        dfo[i] = dfo[i] + Points[i,j]**2
        dfo[i] = np.sqrt( dfo[i])
    if dfo[i]>1 :
        OutsidePoints=OutsidePoints+1
OutsidePoints/Npoints
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-282-740e535c7938> in <module>()
      2 Npoints=100000
      3 Points = np.random.rand(Npoints,Ndim)
```

```
Ndim=2
Npoints=100000
Points = np.random.rand(Npoints,Ndim)
dfo = np.zeros((Npoints,1)) #distancr from zeros , #brace error here
OutsidePoints=0
for i in range(Npoints) :
    for j in range(Ndim) :
        dfo[i] = np.sqrt(
                        dfo[i] + Points[i,j]**2
                        )
    if dfo[i]>1 :
        OutsidePoints=OutsidePoints+1
OutsidePoints/Npoints
```

```
0.333466
```

```
range(0,10,2)
```

```
range(0, 10, 2)
```

```
print(range(0,100,2))
#actual op is 0,2,4,6,8,10,12,14,16,.......
```

```
range(0, 100, 2)
```

```
for i in range(0,10,2) :
    print(i)
```

```
0
2
4
6
8
```

```
a=4
b=5
```

```
a+=b
a
#similarly - * /

```

```
14
```

```
# int = int + float operation doesnt holds good
```

```
#printing row wise
for row in a1:
    print(row)
```

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

```
a2.flat
```

```
<numpy.flatiter at 0x5615188be100>
```

```
np.__version__
```

```
'1.21.5'
```

```
np.show_config()
```

```
blas_mkl_info:
  NOT AVAILABLE
blis_info:
  NOT AVAILABLE
openblas_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
    runtime_library_dirs = ['/usr/local/lib']
blas_opt_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
    runtime_library_dirs = ['/usr/local/lib']
lapack_mkl_info:
  NOT AVAILABLE
openblas_lapack_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
    runtime_library_dirs = ['/usr/local/lib']
lapack_opt_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
    runtime_library_dirs = ['/usr/local/lib']
Supported SIMD extensions in this NumPy install:
    baseline = SSE,SSE2,SSE3
    found = SSSE3,SSE41,POPCNT,SSE42,AVX,F16C,FMA3,AVX2
    not found = AVX512F,AVX512CD,AVX512_KNL,AVX512_KNM,AVX512_SKX,AVX512_CLX,AVX512_(
```

```
np.info(np.add)
```

```
add(x1, x2, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, s

Add arguments element-wise.

Parameters
----------
x1, x2 : array_like
    The arrays to be added.
    If ``x1.shape != x2.shape``, they must be broadcastable to a common
    shape (which becomes the shape of the output).
out : ndarray, None, or tuple of ndarray and None, optional
    A location into which the result is stored. If provided, it must have
    a shape that the inputs broadcast to. If not provided or None,
    a freshly-allocated array is returned. A tuple (possible only as a
    keyword argument) must have length equal to the number of outputs.
where : array_like, optional
    This condition is broadcast over the input. At locations where the
    condition is True, the `out` array will be set to the ufunc result.
    Elsewhere, the `out` array will retain its original value.
    Note that if an uninitialized `out` array is created via the default
    ``out=None``, locations within it where the condition is False will
    remain uninitialized.
**kwargs
    For other keyword-only arguments, see the
    :ref:`ufunc docs <ufuncs.kwargs>`.

Returns
-------
add : ndarray or scalar
    The sum of `x1` and `x2`, element-wise.
    This is a scalar if both `x1` and `x2` are scalars.

Notes
-----
Equivalent to `x1` + `x2` in terms of array broadcasting.

Examples
--------
>>> np.add(1.0, 4.0)
5.0
>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> np.add(x1, x2)
array([[  0.,   2.,   4.],
       [  3.,   5.,   7.],
       [  6.,   8.,  10.]])

The ``+`` operator can be used as a shorthand for ``np.add`` on ndarrays.

>>> x1 = np.arange(9.0).reshape((3, 3))
>>> x2 = np.arange(3.0)
>>> x1 + x2
array([[ 0.,  2.,  4.],
       [ 3.,  5.,  7.],
       [ 6.,  8., 10.]])
```

```
ar6=np.ones(4,4)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-271-55a6c6c8e50b> in <module>()
----> 1 ar6=np.ones(4,4)

/usr/local/lib/python3.7/dist-packages/numpy/core/numeric.py in ones(shape, dtype,
order, like)
    202         return _ones_with_like(shape, dtype=dtype, order=order, like=like)
    203 
--> 204     a = empty(shape, dtype, order)
    205     multiarray.copyto(a, 1, casting='unsafe')
    206     return a

TypeError: Cannot interpret '4' as a data type
```

SEARCH STACK OVERFLOW

```
ar6=np.ones((4,4))
ar6
```

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

```
np.pad(ar6,pad_width=1,mode='constant',constant_values=0)
```

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

```
  Z=np.diag(np.arange(1,6))
  Z
```

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

```
  np.tile(np.array([0,1],[1,0]),(4,4)) #braces error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-277-a805fb05dfa7> in <module>()
----> 1 np.tile(np.array([0,1],[1,0]),(4,4))
```

```
np.tile(np.array([[0,1],[1,0]]),(4,4))
```

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

```
#normalise means () xi - xbar )/std
```

```
np.datetime64('today')
```

```
numpy.datetime64('2022-04-15')
```