<u>Data type in numpy</u>:

<u>Observe the difference</u>

```
import numpy as np

a=np.array([10,12,14])

print(a)
```

o/p:

```
[10 12 14]
```

Or

```
import numpy as np

dt=np.dtype([('age','f4')])

a=np.array([10,12,14],dtype=dt)

print(a)
```

o/p:

```
[(10.,) (12.,) (14.,)]
```

Or,

```
import numpy as np

dt=np.dtype([('age',np.float32)])

a=np.array([10,12,14],dtype=dt)

print(a)
```

o/p:

```
[(10.,) (12.,) (14.,)]
```

Or,

```
import numpy as np

dt=np.dtype([('age','i4')])

a=np.array([10,12,14],dtype=dt)
```

```
print(a)

o/p:

[(10,) (12,) (14,)]
```

Or

```
import numpy as np

dt=np.dtype([('age',np.int32)])

a=np.array([10,12,14],dtype=dt)

print(a)

o/p:

[(10,) (12,) (14,)]
```

Or,

```
import numpy as np
dt=np.dtype('i4')
print(dt)

o/p:

int32
```

or,

```
import numpy as np
dt=np.dtype('f4')
print(dt)

o/p:

float32
```

or,

```
import numpy as np
dt=np.dtype('S20')
print(dt)

o/p:

|S20
```

Or.

```
import numpy as np
student=np.dtype([('name','S20'),('age','i4'),('marks','f4')])
a=np.array([('bikas',20,25),('shubha',25,36)],dtype=student)
print(a)
```

o/p:

```
[(b'bikas', 20, 25.) (b'shubha', 25, 36.)]
```

Or,

```
import numpy as np
student=np.dtype([('name','S20'),('age','i4'),('marks','i4')])
a=np.array([('bikas',20,25),('shubha',25,36)],dtype=student)
print(a)
```

o/p:

```
[(b'bikas', 20, 25) (b'shubha', 25, 36)]
```

Or,

```
import numpy as np
a=np.array([('bikas',20,25),('shubha',25,36)])
print(a)
```

o/p:

```
[['bikas' '20' '25']
 ['shubha' '25' '36']]
```

Or,

```
import numpy as np
a=np.array([['bikas',20,25],['shubha',25,36]])
print(a)
```

o/p:

```
[['bikas' '20' '25']
 ['shubha' '25' '36']
```

See the difference:

```
import numpy as np
a=np.array([[1,2,3],[4,5,6]])
a.shape=(3,2)
print(a)
```

o/p:

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

Or,

```
import numpy as np
a=np.array([[1,2,3],[4,5,6]])
b=a.reshape(3,2)
print(b)
```

o/p:

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

Or,

```
import numpy as np
a=np.array([[[1,2,3],[4,5,6],[2,3,4],[5,6,7]]])
print(a)
```

o/p:

```
[[[1 2 3]
  [4 5 6]
  [2 3 4]
  [5 6 7]]]
```

<u>Itemsize</u>

```
import numpy as np
a=np.array([1,2,3,4],dtype=np.int16)
a.itemsize
```

o/p:

```
2
```

Or,

```
import numpy as np
a=np.array([1,2,3,4],dtype=np.int8)
a.itemsize
```

o/p:

```
1
```

Or,

```
import numpy as np
a=np.array([1,2,3,4],dtype=np.int32)
```

```
a.itemsize
```

o/p:

```
4
```

Or,

```
import numpy as np
a=np.array([1,2,3,4,5])
a.itemsize
```

o/p:

```
4
```

Or,

```
import numpy as np
a=np.array([1,2,3,4,5],dtype=np.float32)
a.itemsize
```

o/p:

```
4
```

Or,

```
import numpy as np
x=np.empty([5,4],dtype=int)
print(x)
```

o/p:

```
[[1699514400        230 1699520384        230]
 [1752658912        230 1699489504        230]
 [1752664096        230 1699489120        230]
 [1699493360        230 1699498160        230]
 [1699493968        230 1699518272        230]]
```

Or,

```
import numpy as np
x=np.empty([5,4],dtype=float)
print(x)
```

o/p:

```
[[1.29441743e-312 2.37663529e-312 2.05833592e-312 2.41907520e-312]
 [2.56761491e-312 1.93101617e-312 9.33678148e-313 9.33678148e-313]
 [9.33678148e-313 9.33678148e-313 1.97345609e-312 2.12199579e-312]
 [2.56761491e-312 2.14321575e-312 2.16443571e-312 2.35541533e-312]
 [2.46151512e-312 1.06099790e-312 8.07618130e-144 1.50008929e+248]]
```

Zeros

```
import numpy as np
a=np.zeros(6)
print(a)
```

o/p:

```
[0. 0. 0. 0. 0. 0.]
```

Or,

```
import numpy as np
a=np.zeros(6,dtype=float)
print(a)
```

op,

```
[0. 0. 0. 0. 0. 0.]
```

Or,

```
import numpy as np
a=np.zeros(6,dtype=int)
print(a)
```

o/p:

```
[0 0 0 0 0 0]
```

Or,

```
import numpy as np
a=np.zeros([2,2],dtype=float)
print(a)
```

o/p:

```
[[0. 0.]
 [0. 0.]]
```

Or,

```
import numpy as np
a=np.zeros([2,2],dtype=int)
print(a)
```

o/p:

```
[[0 0]
 [0 0]]
```

Or,

```
import numpy as np
a=np.zeros((2,2),dtype=[('x','i4'),('y','i4')])
print(a)

o/p:

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

Or,

```
import numpy as np
a=np.zeros([2,2],dtype=[('x','i4'),('y','i4')])
print(a)

o/p:

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

Or,

```
import numpy as np
a=np.zeros([2,2],dtype=[['x','i4'],['y','i4']])
print(a)

o/p:

---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call
last)
<ipython-input-57-9fe96feec492> in <module>
      1 import numpy as np
----> 2 a=np.zeros([2,2],dtype=[['x','i4'],['y','i4']])
      3 print(a)

TypeError: data type not understood
```

Or,

```
import numpy as np
a=np.zeros([2,2],dtype=[('x','f4'),('y','f4')])
print(a)

o/p:

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

Or,

```
import numpy as np
a=np.zeros([4,3],dtype=[('x','f4'),('y','f4')])
print(a)

o/p:

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

Or,

```
import numpy as np
a=np.zeros([4,3],dtype=[('x','i4'),('y','i4')])
print(a)

o/p:

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

Or,

```
import numpy as np
a=np.zeros([4,3],dtype=[('x','i4'),('y','i4'),('z','i4')])
print(a)

o/p:

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

<u>np.ones</u>

<u>or,</u>

```
import numpy as np
a=np.ones(6)
print(a)

o/p:

[1. 1. 1. 1. 1. 1.]
```

Or,

```
import numpy as np
```

```
a=np.ones([4,3])
print(a)
```

o/p:

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

Or,

```
import numpy as np
a=np.ones([4,3],dtype=int)
print(a)
```

o/p:

```
[[1 1 1]
 [1 1 1]
 [1 1 1]
 [1 1 1]]
```

o/p:

```
import numpy as np
a=np.ones([4,3],dtype=[('x','i4'),('y','i4')])
print(a)
```

o/p:

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

Or,

```
import numpy as np
a=np.ones([4,3],dtype=[('x','i4'),('y','i4'),('z','i4')])
print(a)
```

o/p:

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

np.full

```
import numpy as np
a=np.full([4,3],3.14,dtype=([('x','f4'),('y','f4'),('z','f4')]))
print(a)
```

o/p:

```
[[(3.14, 3.14, 3.14) (3.14, 3.14, 3.14) (3.14, 3.14, 3.14)]
 [(3.14, 3.14, 3.14) (3.14, 3.14, 3.14) (3.14, 3.14, 3.14)]
 [(3.14, 3.14, 3.14) (3.14, 3.14, 3.14) (3.14, 3.14, 3.14)]
 [(3.14, 3.14, 3.14) (3.14, 3.14, 3.14) (3.14, 3.14, 3.14)]]
```

Or,

```
import numpy as np
a=np.full([4,3],3.14)
print(a)
```

o/p:

```
[[3.14 3.14 3.14]
 [3.14 3.14 3.14]
 [3.14 3.14 3.14]
 [3.14 3.14 3.14]]
```

See the difference:

```
import numpy as np
x=[1,2,3]
b=np.asarray(x)
print(b)
```

o/p:

```
[1 2 3]
```

Or,

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

o/p:

```
[1 2 3]
```

np.asarray

```
import numpy as np
x=[1,2,3]
b=np.asarray(x)
```

```
print(b)
```

o/p:

```
[1 2 3]
```

Or,

```
import numpy as np
a=[1,2,3]
x=np.asarray(a,dtype=float)
print(x)
```

o/p:

```
[1. 2. 3.]
```

Or,

```
import numpy as np
a=[(1,2,3),(4,5)]
x=np.asarray(x)
print(x)
```

o/p:

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

Or,

```
import numpy as np
a=(1,2,3)
x=np.asarray(a)
print(x)
```

o/p:

```
[1 2 3]
```

np.copy

```
import numpy as np
s=[1,2,3,4]
a=np.asarray(s)
b=np.copy(a)
print('original array',a)
print('copy of original array',b)
```

o/p:

```
original array [1 2 3 4]
```

```
copy of original array [1 2 3 4]
```

<u>np.frombuffer</u>

```
import numpy as np
s = b'hello world'
a = np.frombuffer(s,dtype='S1')
print (a)
```

o/p:

```
[b'h' b'e' b'l' b'l' b'o' b' ' b'w' b'o' b'r' b'l' b'd']
```

<u>np.fromiter:</u>

<u>see the difference</u>

```
import numpy as np
iterable=range(10)
a=np.array(iterable)
print(a)
```

o/p:

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

Or,

```
import numpy as np
iterable=range(10)
a=np.fromiter(iterable,dtype='i4')
print(a)
```

o/p:

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

Or,

```
import numpy as np
iterable=range(10)
a=np.fromiter(iterable,dtype='f4')
print(a)
```

o/p:

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

NB: in ‘fromiter’ datatype is mandatory

Or,

```
import numpy as np
value=(x*x for x in range (10))
x=np.fromiter(value,dtype=int)
print(x)
```

o/p:

```
[ 0  1  4  9 16 25 36 49 64 81]
```

np.fromstring

```
import numpy as np
strval='1 2 3 4 5'
a=np.fromstring(strval,dtype=int,sep=' ')
print(a)
```

o/p:

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

Or,

```
import numpy as np
strval='1111 2222 3333 4444 5555'
np.fromstring(strval,dtype=int,sep=' ')
```

o/p:

```
array([1111, 2222, 3333, 4444, 5555])
```

np.arange

```
import numpy as np
x=np.arange(10,20,2,dtype=float)#start,stop,step,datatype
print(x)
```

o/p:

```
[10. 12. 14. 16. 18.]
```

np.linspace

```
import numpy as np
x=np.linspace(10,20,5)
print(x)
```

o/p:

```
[10.  12.5 15.  17.5 20. ]
```

Or,

```
import numpy as np
x=np.linspace(10,30,5,endpoint=False)
print(x)# it ensure that end value doesnot exceed stop value.
```

o/p:

```
[10. 14. 18. 22. 26.]
```

Or,

```
import numpy as np
x=np.linspace(10,20,5,endpoint=False)
print(x)# it ensure that end value doesnot exceed stop value.
```

o/p:

```
[10. 12. 14. 16. 18.]
```

Or,

```
import numpy as np
x=np.linspace(10,20,5,retstep=True)# return  step between two consequtive
number
print(x)
```

o/p:

```
(array([10. , 12.5, 15. , 17.5, 20. ]), 2.5)
```

Or,

```
import numpy as np
x=np.linspace(10,30,5,retstep=True)# return  step between two consequtive
number
print(x)
```

o/p:

```
(array([10., 15., 20., 25., 30.]), 5.0)
```

<u>np.logspace:</u>

```
import numpy as np
x=np.logspace(1,2,num=10)
print(x)
```

o/p:

```
[ 10.          12.91549665  16.68100537  21.5443469   27.82559402
```

```
  35.93813664  46.41588834  59.94842503  77.42636827 100.        ]
```

Or,

```
import numpy as np
x=np.logspace(1,2,num=10,base=2)
print(x)
```

o/p:

```
[2.         2.16011948 2.33305808 2.5198421  2.72158    2.93946898
 3.1748021  3.42897593 3.70349885 4.        ]
```

Or,

```
import numpy as np
x=np.logspace(1,10,num=10,base=2)
print(x)
```

o/p:

```
[   2.    4.    8.   16.   32.   64.  128.  256.  512. 1024.]
```

Slice of an array:

```
import numpy as np
a=np.arange(10)
s=slice(2,7,2)
print(a[s])
```

o/p:

```
[2 4 6]
```

Or,

```
import numpy as np
a=np.arange(10)
s=a[2:7:2]
print(s)
```

o/p:

```
[2 4 6]
```

Or,

```
#slice single element
import numpy as np
a=np.arange(10)
print(a[5])
```

```
o/p:

5

Or,

#slicing element between indices
import numpy as np
a=np.arange(10)
print(a[2:5])

o/p:

[2 3 4]


Or,

import numpy as np
a=np.array([[1,2,3],[3,4,5],[4,5,6]])
print(a)
print('the slicing array','\n',a[1:])

o/p:

[[1 2 3]
 [3 4 5]
 [4 5 6]]
the slicing array
 [[3 4 5]
 [4 5 6]]


Or,

import numpy as np
a=np.array([[1,2,3,4],[2,3,4,5],[4,5,6,7],[7,8,9,1]])
print('our is :','\n',a)
print('\n')
print('the element of second coloumn is:',a[...,1])
print('the element of first coloumn is:',a[...,0])
print('the element of third coloumn is:',a[...,2])
print('the element of second row is:',a[1,...])
print('the element of first row is:',a[0,...])
print('the element of third row is:',a[2,...])
print('the element of forth row is:',a[3,...])
print('the element of second coloumn to onward is:','\n',a[...,1:])
print('the element of third coloumn to forth column is:','\n',a[...,2:4])
print('the element of second row to onward is:','\n',a[1:,...])
print('the element of third row to forth row is:','\n',a[2:4,...])

o/p:
```

```
our is :
 [[1 2 3 4]
 [2 3 4 5]
 [4 5 6 7]
 [7 8 9 1]]


the element of second coloumn is: [2 3 5 8]
the element of first coloumn is: [1 2 4 7]
the element of third coloumn is: [3 4 6 9]
the element of second row is: [2 3 4 5]
the element of first row is: [1 2 3 4]
the element of third row is: [4 5 6 7]
the element of forth row is: [7 8 9 1]
the element of second coloumn to onward is:
 [[2 3 4]
 [3 4 5]
 [5 6 7]
 [8 9 1]]
the element of third coloumn to forth column is:
 [[3 4]
 [4 5]
 [6 7]
 [9 1]]
the element of second row to onward is:
 [[2 3 4 5]
 [4 5 6 7]
 [7 8 9 1]]
the element of third row to forth row is:
 [[4 5 6 7]
 [7 8 9 1]]
```

Advance slicing:

```
import numpy as np
a=np.array([[1,2],[3,4],[4,5]])
print(a)
print('\n')
print (a[[0,1,2],[1,0,1]])
```

o/p:

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



[2 3 5]
```

In [ ]:

Or,

```
import numpy as np
```

```
a=np.array([[1,2,3],[3,4,5],[6,7,8],[8,4,5]])
print(a)
print('\n')
row=np.array([[0,1],[3,2]])
print(row)
print('\n')
col=np.array([[1,2],[0,0]])
print(col)
print('\n')
print(a[row,col])
```

o/p:

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


[[0 1]
 [3 2]]


[[1 2]
 [0 0]]


[[2 5]
 [8 6]]
```

o/r:

```
import numpy as np
a=np.array([[1,2,3],[3,4,5],[6,7,8],[8,4,5]])
print(a)
print('\n')
row=np.array([[0,1],[2,2]])# now it is represent as column
print(row)
print('\n')
col=np.array([[1,2],[0,0]]) #now it is represent as row
print(col)
print('\n')
print(a[col,row])
```

o/p:

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


[[0 1]
 [2 2]]
```

```
[[1 2]
 [0 0]]


[[3 7]
 [3 3]]
```

Or,

```
import numpy as np
a=np.array([[1,2,3],[4,5,6],[7,8,9],[9,8,7]])
print(a)
print('\n')
print(a[2:4,0:2])
print('\n')
print(a[2:4,1:2])
```

o/p:

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


[[7 8]
 [9 8]]


[[8]
 [8]]
```

Or,

```
import numpy as np
a=np.array([[1,2,3],[4,5,6],[7,8,9],[9,8,7]])
print(a)
print('\n')
print(a[2:4,[1,2]])
print('\n')
print(a[2:4,1:2])
```

o/p:

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

```
[[8 9]
 [8 7]]


[[8]
 [8]]
```

Or,

```
import numpy as np
a=np.array([[1,2,3,3],[4,5,6,4],[7,8,9,5],[9,10,11,6]])
print(a)
print('\n')
print(a[2:4,1:3])
print('\n')
print(a[2:4,[1,3]])
```

o/p:

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


[[ 8  9]
 [10 11]]


[[ 8  5]
 [10  6]]
```

Or,

```
import numpy as np
a=np.array([[1,2,3,4],[4,5,6,7],[5,6,7,8],[6,7,8,9]])
print('the greater than five are:',a[a>5])
```

o/p:

```
the greater than five are: [6 7 6 7 8 6 7 8 9]
```

or,

```
import numpy as np
a=np.array([[np.nan,1,2,np.nan,4,5],[np.nan,7,9,np.nan,8,8]])
print(a[~np.isnan(a)])
```

o/p:

```
[1. 2. 4. 5. 7. 9. 8. 8.]
```

Or,

```
import numpy as np
a=np.array([2+3j,5,5+6j,3+6j,8])
print(a[~np.iscomplex(a)])
```

o/p:

```
[5.+0.j 8.+0.j]
```

Or,

```
import numpy as np
a=np.array([2+3j,5,5+6j,3+6j,8])
print(a[np.iscomplex(a)])
```

o/p:

```
[2.+3.j 5.+6.j 3.+6.j]
```

Broadcasting

```
import numpy as np
a=np.array([1,2,3,4])
print(a)
print('\n')
b=np.array([10,20,30,40])
print(b)
print('\n')
c=a*b
print(c)
```

o/p:

```
[1 2 3 4]


[10 20 30 40]


[ 10  40  90 160]
```

Or,

```
import numpy as np
a= np.array([[0,0,0],[10,10,10],[20,20,20],[30,30,30]])
print(a)
print('\n')
b=np.array([1,2,3])
print(b)
print('\n')
```

```
c=a+b
print(c)
```

o/p:

```
[[ 0  0  0]
 [10 10 10]
 [20 20 20]
 [30 30 30]]


[1 2 3]


[[ 1  2  3]
 [11 12 13]
 [21 22 23]
 [31 32 33]]
```

Or,

```
import numpy as np
a= np.array([[0,0,0],[10,10,10],[20,20,20],[30,30,30]])
print(a)
print('\n')
b=np.array([1])
print(b)
print('\n')
c=a+b
print(c)
```

o/p:

```
[[ 0  0  0]
 [10 10 10]
 [20 20 20]
 [30 30 30]]


[1]


[[ 1  1  1]
 [11 11 11]
 [21 21 21]
 [31 31 31]]
```

Broadcasting:

```
import numpy as np
my_3d_array=np.arange(70,dtype='i4')
my_3d_array.shape=(2,7,5)
print(my_3d_array)
print(my_3d_array.shape)
print(my_3d_array.ndim)
print(my_3d_array.size)
print(my_3d_array.itemsize)
print(my_3d_array.dtype)
print(my_3d_array-2)
print(5*my_3d_array-2)#np.broadcasting treat scaler having size 1
```

o/p:

```
[[[ 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]]]
(2, 7, 5)
3
70
4
int32
[[[-2 -1  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]]]
[[[ -2   3   8  13  18]
  [ 23  28  33  38  43]
  [ 48  53  58  63  68]
  [ 73  78  83  88  93]
  [ 98 103 108 113 118]
  [123 128 133 138 143]
  [148 153 158 163 168]]

 [[173 178 183 188 193]
  [198 203 208 213 218]
  [223 228 233 238 243]
  [248 253 258 263 268]
  [273 278 283 288 293]
  [298 303 308 313 318]
  [323 328 333 338 343]]]
```

```
Or,,
import numpy as np
left_mat=np.arange(6).reshape(2,3)
print(left_mat)
print('\n')
right_mat=np.arange(15).reshape(3,5)
print(right_mat)
print(np.dot(left_mat,right_mat))

o/p:

[[0 1 2]
 [3 4 5]]


[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
[[ 25  28  31  34  37]
 [ 70  82  94 106 118]]
```

```
Or,

import numpy as np
my_3d_array=np.arange(70,dtype='i4')
my_3d_array.shape=(2,7,5)
print(my_3d_array)
print('\n')
print(my_3d_array.sum())
print('\n')
print(my_3d_array.sum(axis=0))
print('\n')
print(my_3d_array.sum(axis=1))
print('\n')
print(my_3d_array.sum(axis=2))

o/p:

[[[ 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]]]


2415


[[ 35  37  39  41  43]
 [ 45  47  49  51  53]
 [ 55  57  59  61  63]
 [ 65  67  69  71  73]
 [ 75  77  79  81  83]
 [ 85  87  89  91  93]
 [ 95  97  99 101 103]]


[[105 112 119 126 133]
 [350 357 364 371 378]]


[[ 10  35  60  85 110 135 160]
 [185 210 235 260 285 310 335]]
```

NB. ** if we add across 0th axis then we eliminate 0th element that is (7,5) matrix

** if we add across 1th axis then we eliminate 1st element that is (2,5) matrix

** if we add across 0th axis then we eliminate 2nd element that is (2,7) matrix

Or,

```
array_a=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(array_a)
print('\n')
array_b=np.array([3,2,1])
print(array_b)
print('\n')
print(array_a+array_b)
print('\n')
array_c=np.array([[3],[2],[1]])
print(array_c)
print('\n')
print(array_a+array_c)
```

o/p:

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


[3 2 1]


[[ 4  4  4]
 [ 7  7  7]
 [10 10 10]]


[[3]
 [2]
 [1]]


[[ 4  5  6]
 [ 6  7  8]
 [ 8  9 10]]
```

Or,

```
array_d=np.ones(25,dtype=int).reshape(5,5)
print('array_d is:','\n',array_d)
array_e=np.array([1,2,3,4,5])
print('array_e is:','\n',array_e)
print('\n')
print('array_d*array_e is:','\n',array_d*array_e)
array_f=np.array([[1],[2],[3],[4],[5]])
print('array_f is:','\n',array_f)
print('\n')
print('array_e*array_f is:','\n',array_e*array_f)
print('\n')
print('array_d*array_f is:','\n',array_d*array_f)
print('\n')
print('array_f*array_d is:','\n',array_f*array_d)
```

o/p:

```
array_d is:
 [[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]]
array_e is:
 [1 2 3 4 5]


array_d*array_e is:
 [[1 2 3 4 5]
 [1 2 3 4 5]
 [1 2 3 4 5]
 [1 2 3 4 5]
 [1 2 3 4 5]]
array_f is:
 [[1]
 [2]
 [3]
 [4]
 [5]]


array_e*array_f is:
 [[ 1  2  3  4  5]
 [ 2  4  6  8 10]
 [ 3  6  9 12 15]
 [ 4  8 12 16 20]
 [ 5 10 15 20 25]]


array_d*array_f is:
 [[1 1 1 1 1]
 [2 2 2 2 2]
 [3 3 3 3 3]
 [4 4 4 4 4]
 [5 5 5 5 5]]


array_f*array_d is:
 [[1 1 1 1 1]
 [2 2 2 2 2]
 [3 3 3 3 3]
 [4 4 4 4 4]
 [5 5 5 5 5]]
```

o/r,

```
import numpy as np
a=np.array([[1,2,3],[2,3,4],[3,4,5]])
print(a)
b=np.array([0,1,2])
print(b)
print(a+b)
c=np.array([[0,1,2],[0,1,2],[0,1,2]])
```

```
print(c)
print(a+c)

o/p:

[[1 2 3]
 [2 3 4]
 [3 4 5]]
[0 1 2]
[[1 3 5]
 [2 4 6]
 [3 5 7]]
[[0 1 2]
 [0 1 2]
 [0 1 2]]
[[1 3 5]
 [2 4 6]
 [3 5 7]]
```

np.nditer

```
import numpy as np
x=np.arange(0,60,5).reshape(3,4)
print(x)
print('\n')
for i in np.nditer(x):
    print(i,end=' ')

o/p:

[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]


Transpose of this matrix
 [[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
 [15 35 55]]


0 5 10 15 20 25 30 35 40 45 50 55
```

Or,

```
import numpy as np
x=np.arange(0,60,5).reshape(3,4)
print(x)
print('\n')
b=x.T
print('Transpose of this matrix','\n',b)
print('\n')
for i in np.nditer(x):
    print(i,end=' ')
```

o/p:

```
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]


Transpose of this matrix
 [[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
 [15 35 55]]


0 5 10 15 20 25 30 35 40 45 50 55
```

Or,

```
import numpy as np
x=np.arange(0,48,2).reshape(4,6)
print(x)
y=x.T
print('\n')
print('Transpose of the matrix','\n',y)
print('\n')
for i in np.nditer(y):
    print(i,end=' ')
```

o/p:

```
[[ 0  2  4  6  8 10]
 [12 14 16 18 20 22]
 [24 26 28 30 32 34]
 [36 38 40 42 44 46]]


Transpose of the matrix
 [[ 0 12 24 36]
 [ 2 14 26 38]
 [ 4 16 28 40]
 [ 6 18 30 42]
 [ 8 20 32 44]
 [10 22 34 46]]


0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46
```

Or,

```
import numpy as np
x=np.arange(0,48,2).reshape(4,6)
print(x)
print('\n')
y=x.T
print('the transpose of the matrix','\n',y)
print('\n')
z=y.copy(order='C')
print(z)
print('\n')
for i in np.nditer(z):
    print(i,end=' ')
print('\n')
a=y.copy(order='F')
print(z)
print('\n')
for i in np.nditer(a):
    print(i,end=' ')
print('\n')
```

o/p:

```
[[ 0  2  4  6  8 10]
 [12 14 16 18 20 22]
 [24 26 28 30 32 34]
 [36 38 40 42 44 46]]


the transpose of the matrix
 [[ 0 12 24 36]
 [ 2 14 26 38]
 [ 4 16 28 40]
 [ 6 18 30 42]
 [ 8 20 32 44]
 [10 22 34 46]]


[[ 0 12 24 36]
 [ 2 14 26 38]
 [ 4 16 28 40]
 [ 6 18 30 42]
 [ 8 20 32 44]
 [10 22 34 46]]


0 12 24 36 2 14 26 38 4 16 28 40 6 18 30 42 8 20 32 44 10 22 34 46

[[ 0 12 24 36]
 [ 2 14 26 38]
 [ 4 16 28 40]
 [ 6 18 30 42]
 [ 8 20 32 44]
 [10 22 34 46]]
```

```
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46
```

Or,

```
import numpy as np
a=np.arange(1,13).reshape(4,3)
print(a)
print('iterate in C type order','\n')
for i in np.nditer(a,order='C'):
    print(i,end=' ')
print('\n','iterate in F type order','\n')
for i in np.nditer(a,order='F'):
    print(i,end=' ')
```

o/p:

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
iterate in C type order

1 2 3 4 5 6 7 8 9 10 11 12
 iterate in F type order

1 4 7 10 2 5 8 11 3 6 9 12
```

Or,

```
import numpy as np
a=np.arange(0,72,3).reshape(6,4)
print(a)
print('\n')
for i in np.nditer(a,op_flags=['readwrite']):
    i[...]=i*5
print('the modified array is:','\n',a)
```

o/p:

```
[[ 0  3  6  9]
 [12 15 18 21]
 [24 27 30 33]
 [36 39 42 45]
 [48 51 54 57]
 [60 63 66 69]]


the modified array is:
 [[  0  15  30  45]
 [ 60  75  90 105]
 [120 135 150 165]
 [180 195 210 225]
 [240 255 270 285]
 [300 315 330 345]]
```

Or,

```
import numpy as np
x=np.arange(0,70,2).reshape(5,7)
print(x)
print('\n')
for i in np.nditer(x,flags=['external_loop'],order='C'):
    print(i,end=' ')
print('\n')
for i in np.nditer(x,flags=['external_loop'],order='F'):
    print(i,end=' ')
```

o/p:

```
[[ 0  2  4  6  8 10 12]
 [14 16 18 20 22 24 26]
 [28 30 32 34 36 38 40]
 [42 44 46 48 50 52 54]
 [56 58 60 62 64 66 68]]


[ 0  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46
 48 50 52 54 56 58 60 62 64 66 68]

[ 0 14 28 42 56] [ 2 16 30 44 58] [ 4 18 32 46 60] [ 6 20 34 48 62] [ 8 22
36 50 64] [10 24 38 52 66] [12 26 40 54 68]
```

Or,

```
import numpy as np
x=np.arange(0,70,2).reshape(5,7)
print(x)
print('\n')
for i in np.nditer(x,flags=['c_index'],order='C'):
    print(i,end=' ')
print('\n')
for i in np.nditer(x,flags=['c_index'],order='F'):
    print(i,end=' ')
```

o/p:

```
[[ 0  2  4  6  8 10 12]
 [14 16 18 20 22 24 26]
 [28 30 32 34 36 38 40]
 [42 44 46 48 50 52 54]
 [56 58 60 62 64 66 68]]
```

```
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
52 54 56 58 60 62 64 66 68

0 14 28 42 56 2 16 30 44 58 4 18 32 46 60 6 20 34 48 62 8 22 36 50 64 10
24 38 52 66 12 26 40 54 68
```

Or,

```
import numpy as np
x=np.arange(0,70,2).reshape(5,7)
print(x)
print('\n')
for i in np.nditer(x,order='C'):
    print(i,end=' ')
print('\n')
for i in np.nditer(x,order='F'):
    print(i,end=' ')
```

o/p:

```
[[ 0  2  4  6  8 10 12]
 [14 16 18 20 22 24 26]
 [28 30 32 34 36 38 40]
 [42 44 46 48 50 52 54]
 [56 58 60 62 64 66 68]]


0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
52 54 56 58 60 62 64 66 68

0 14 28 42 56 2 16 30 44 58 4 18 32 46 60 6 20 34 48 62 8 22 36 50 64 10
24 38 52 66 12 26 40 54 68
```

Or,

```
import numpy as np
x=np.arange(0,70,2).reshape(5,7)
print(x)
print('\n')
for i in np.nditer(x,flags=['f_index'],order='C'):
    print(i,end=' ')
print('\n')
for i in np.nditer(x,flags=['f_index'],order='F'):
    print(i,end=' ')
```

o/p:

```
[[ 0  2  4  6  8 10 12]
 [14 16 18 20 22 24 26]
 [28 30 32 34 36 38 40]
 [42 44 46 48 50 52 54]
 [56 58 60 62 64 66 68]]
```

```
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
52 54 56 58 60 62 64 66 68

0 14 28 42 56 2 16 30 44 58 4 18 32 46 60 6 20 34 48 62 8 22 36 50 64 10
24 38 52 66 12 26 40 54 68
```

Or,

```
import numpy as np
x=np.arange(0,60,2).reshape(5,6)
print(x)
print('\n')
y=np.array([1,2,3,4,5,6])
print(y)
print('\n')
for i,j in np.nditer([x,y]):
    print('{}:{}'. format (i,j),end=' ')
```

o/p:

```
[[ 0  2  4  6  8 10]
 [12 14 16 18 20 22]
 [24 26 28 30 32 34]
 [36 38 40 42 44 46]
 [48 50 52 54 56 58]]


[1 2 3 4 5 6]

0:1 2:2 4:3 6:4 8:5 10:6 12:1 14:2 16:3 18:4 20:5 22:6 24:1 26:2 28:3 30:4
32:5 34:6 36:1 38:2 40:3 42:4 44:5 46:6 48:1 50:2 52:3 54:4 56:5 58:6
```

Or,

```
import numpy as np
x=np.arange(0,60,2).reshape(5,6)
print(x)
print('\n')
y=np.array([[1],[2],[3],[4],[5]])
print(y)
print('\n')
for i,j in np.nditer([x,y]):
    print('{}:{}'. format (i,j),end=' ')
```

o/p:

```
[[ 0  2  4  6  8 10]
 [12 14 16 18 20 22]
 [24 26 28 30 32 34]
 [36 38 40 42 44 46]
 [48 50 52 54 56 58]]
```

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


0:1 2:1 4:1 6:1 8:1 10:1 12:2 14:2 16:2 18:2 20:2 22:2 24:3 26:3 28:3 30:3
32:3 34:3 36:4 38:4 40:4 42:4 44:4 46:4 48:5 50:5 52:5 54:5 56:5 58:5
```

Flat:

```
import numpy as np
a=np.arange(8).reshape(2,4)
print(a)
print('\n')
print(a.flat[5])
```

o/p:

```
[[0 1 2 3]
 [4 5 6 7]]


5
```

Flatten:

```
import numpy as np
a=np.arange(12).reshape(4,3)
print(a)
print('\n')
print(a.flatten())
```

o/p:

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


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

Or,

```
import numpy as np
a=np.arange(12).reshape(4,3)
print(a)
print('\n')
print('flatten in C order','\n',a.flatten(order='C'))
print('\n')
print('flatten in F order','\n',a.flatten(order='F'))
```

o/p:

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


flatten in C order
 [ 0  1  2  3  4  5  6  7  8  9 10 11]


flatten in F order
 [ 0  3  6  9  1  4  7 10  2  5  8 11]
```

<u>ravel()</u>

```
import numpy as np
a=np.arange(0,60,5).reshape(3,4)
print(a)
print('\n')
print(a.ravel())
```

o/p:

```
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]


[ 0  5 10 15 20 25 30 35 40 45 50 55]
```

Or

```
import numpy as np
a=np.arange(0,60,5).reshape(3,4)
print(a)
print('\n')
print(a.ravel(order='C'))
print('\n')
print(a.ravel(order='F'))
```

o/p:

```
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]
```

```
[ 0  5 10 15 20 25 30 35 40 45 50 55]


[ 0 20 40  5 25 45 10 30 50 15 35 55]
```

np.transpose:

```
import numpy as np
a=np.arange(0,70,2).reshape(5,7)
print(a)
print('\n')
print('Transpose of the matrix','\n',np.transpose(a))
```

o/p:

```
[[ 0  2  4  6  8 10 12]
 [14 16 18 20 22 24 26]
 [28 30 32 34 36 38 40]
 [42 44 46 48 50 52 54]
 [56 58 60 62 64 66 68]]


Transpose of the matrix
 [[ 0 14 28 42 56]
 [ 2 16 30 44 58]
 [ 4 18 32 46 60]
 [ 6 20 34 48 62]
 [ 8 22 36 50 64]
 [10 24 38 52 66]
 [12 26 40 54 68]]
```

Or,

```
import numpy as np
a=np.arange(0,60,5).reshape(3,4)
print(a)
print('\n')
print('Transpose of the matrix','\n',a.T)
```

o/p:

```
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]


Transpose of the matrix
 [[ 0 20 40]
 [ 5 25 45]
 [10 30 50]
 [15 35 55]]
```

Collapsing of matrix,

```
import numpy as np
a=np.arange(24).reshape(2,3,4)
print(a)
print('\n')
print(a[0])
print('\n')
print(a[1])
print('\n')
print('a[0]+a[1]','\n',a[0]+a[1])
print('\n')
print('np.sum(a,axis=0)','\n',np.sum(a,axis=0))
```

o/p:

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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]


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


[[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]


a[0]+a[1]
 [[12 14 16 18]
 [20 22 24 26]
 [28 30 32 34]]


np.sum(a,axis=0)
 [[12 14 16 18]
 [20 22 24 26]
 [28 30 32 34]]
```

Here a[0]+a[1] = np.sum(a,axis=0)
It is called collapsing

o/p:

```
import numpy as np
a=np.arange(24).reshape(2,3,4)
print(a)
print('\n')
print('a[0][0]','\n',a[0][0])
print('\n')
print('a[0][1]','\n',a[0][1])
print('\n')
print('a[0][2]','\n',a[0][2])
print('\n')
print('a[1][0]','\n',a[1][0])
print('\n')
print('a[1][1]','\n',a[1][1])
print('\n')
print('a[1][2]','\n',a[1][2])
print('\n')
print('a[0][0]+a[0][1]+a[0][2]','\n',a[0][0]+a[0][1]+a[0][2])
print('\n')
print('a[1][0]+a[1][1]+a[1][2]','\n',a[1][0]+a[1][1]+a[1][2])
print('\n')
print('np.sum(a,axis=1)','\n',np.sum(a,axis=1))
```

o/p:

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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]


a[0][0]
 [0 1 2 3]


a[0][1]
 [4 5 6 7]


a[0][2]
 [ 8  9 10 11]


a[1][0]
 [12 13 14 15]


a[1][1]
 [16 17 18 19]
```

```
a[1][2]
 [20 21 22 23]


a[0][0]+a[0][1]+a[0][2] -----(1)
 [12 15 18 21]


a[1][0]+a[1][1]+a[1][2] ------(2)
 [48 51 54 57]


np.sum(a,axis=1)
 [[12 15 18 21]
 [48 51 54 57]]
```

Combined (1) and(2) we get np.sum(a,axis=1)

Or,

```
import numpy as np
a=np.arange(24).reshape(2,3,4)
print(a)
print('\n')
print('a[0][0][0]','\n',a[0][0][0])
print('\n')
print('a[0][0][1]','\n',a[0][0][1])
print('\n')
print('a[0][0][2]','\n',a[0][0][2])
print('\n')
print('a[0][0][3]','\n',a[0][0][3])
print('\n')
print('a[0][1][0]','\n',a[0][1][0])
print('\n')
print('a[0][1][1]','\n',a[0][1][1])
print('\n')
print('a[0][1][2]','\n',a[0][1][2])
print('\n')
print('a[0][1][3]','\n',a[0][1][3])
print('\n')
print('a[0][2][0]','\n',a[0][2][0])
print('\n')
print('a[0][2][1]','\n',a[0][2][1])
print('\n')
print('a[0][2][2]','\n',a[0][2][2])
print('\n')
print('a[0][2][3]','\n',a[0][2][3])
print('\n')
print('a[1][0][0]','\n',a[1][0][0])
print('\n')
print('a[1][0][1]','\n',a[1][0][1])
print('\n')
print('a[1][0][2]','\n',a[1][0][2])
print('\n')
print('a[1][0][3]','\n',a[1][0][3])
print('\n')
print('a[1][1][0]','\n',a[1][1][0])
print('\n')
print('a[1][1][1]','\n',a[1][1][1])
print('\n')
print('a[1][1][2]','\n',a[1][1][2])
print('\n')
print('a[1][1][3]','\n',a[1][1][3])
print('\n')
print('a[1][2][0]','\n',a[1][2][0])
print('\n')
print('a[1][2][1]','\n',a[1][2][1])
print('\n')
print('a[1][2][2]','\n',a[1][2][2])
print('\n')
print('a[1][2][3]','\n',a[1][0][3])
print('\n')
print('a[0][0][0]+a[0][0][1]+a[0][0][2]+a[0][0][3]','\n',a[0][0][0]+a[0][0
][1]+a[0][0][2]+a[0][0][3])
print('\n')
print('a[0][1][0]+a[0][1][1]+a[0][1][2]+a[0][1][3]','\n',a[0][1][0]+a[0][1
][1]+a[0][1][2]+a[0][1][3])
print('\n')
print('a[0][2][0]+a[0][2][1]+a[0][2][2]+a[0][2][3]','\n',a[0][2][0]+a[0][2
][1]+a[0][2][2]+a[0][2][3])
print('\n')
print('a[1][0][0]+a[1][0][1]+a[1][0][2]+a[1][0][3]','\n',a[1][0][0]+a[1][0
][1]+a[1][0][2]+a[1][0][3])
print('\n')
print('a[1][1][0]+a[1][1][1]+a[1][1][2]+a[1][1][3]','\n',a[1][1][0]+a[1][1
][1]+a[1][1][2]+a[1][1][3])
print('\n')
print('a[1][2][0]+a[1][2][1]+a[1][2][2]+a[1][2][3]','\n',a[1][2][0]+a[1][2
][1]+a[1][2][2]+a[1][2][3])
print('\n')
print('np.sum(a,axis=2)','\n',np.sum(a,axis=2))
```

o/p:

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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

```
a[0][0][0]
 0


a[0][0][1]
 1


a[0][0][2]
 2


a[0][0][3]
 3


a[0][1][0]
 4


a[0][1][1]
 5


a[0][1][2]
 6


a[0][1][3]
 7


a[0][2][0]
 8


a[0][2][1]
 9


a[0][2][2]
 10
```

```
a[0][2][3]
 11


a[1][0][0]
 12


a[1][0][1]
 13


a[1][0][2]
 14


a[1][0][3]
 15


a[1][1][0]
 16


a[1][1][1]
 17


a[1][1][2]
 18


a[1][1][3]
 19


a[1][2][0]
 20


a[1][2][1]
 21
```

```
a[1][2][2]
 22


a[1][2][3]
 15


a[0][0][0]+a[0][0][1]+a[0][0][2]+a[0][0][3]
 6


a[0][1][0]+a[0][1][1]+a[0][1][2]+a[0][1][3]
 22


a[0][2][0]+a[0][2][1]+a[0][2][2]+a[0][2][3]
 38


a[1][0][0]+a[1][0][1]+a[1][0][2]+a[1][0][3]
 54


a[1][1][0]+a[1][1][1]+a[1][1][2]+a[1][1][3]
 70


a[1][2][0]+a[1][2][1]+a[1][2][2]+a[1][2][3]
 86


np.sum(a,axis=2)
 [[ 6 22 38]
 [54 70 86]]
```

np.transpose()

```
a=np.arange(24).reshape(2,3,4)

print(a)

print('\n')

print(np.transpose(a))

print(np.transpose(a).shape)
```

o/p:

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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]


[[[ 0 12]
  [ 4 16]
  [ 8 20]]

 [[ 1 13]
  [ 5 17]
  [ 9 21]]

 [[ 2 14]
  [ 6 18]
  [10 22]]

 [[ 3 15]
  [ 7 19]
  [11 23]]]
(4, 3, 2)
```

Or,

```
a=np.arange(1,11).reshape(5,2)

print(a)

print('\n')

print(np.transpose(a,axes=(1,0)))
```

o/p:

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


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

In [23]:

Or,

```
a=np.arange(1,11).reshape(5,2)

print(a)

print('\n')

print(np.transpose(a,axes=(0,1)))
```

o/p:

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


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

Or,

```
a=np.arange(24).reshape(2,3,4)
print(a)
print('\n')
print(np.transpose(a,axes=(0,1,2)))
print(np.transpose(a,axes=(0,1,2)).shape)
```

o/p:

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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]


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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
(2, 3, 4)
```

Or,

```
a=np.arange(24).reshape(2,3,4)
print(a)
print('\n')
print(np.transpose(a,axes=(2,1,0)))
print(np.transpose(a,axes=(2,1,0)).shape)
```

o/p:

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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]


[[[ 0 12]
  [ 4 16]
  [ 8 20]]

 [[ 1 13]
```

```
  [ 5 17]
  [ 9 21]]

 [[ 2 14]
  [ 6 18]
  [10 22]]

 [[ 3 15]
  [ 7 19]
  [11 23]]]
(4, 3, 2)
```

Or,,

```
a=np.arange(24).reshape(2,3,4)
print(a)
print('\n')
print(np.transpose(a,axes=(2,0,1)))
print(np.transpose(a,axes=(2,0,1)).shape)
```

o/p:

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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]


[[[ 0  4  8]
  [12 16 20]]

 [[ 1  5  9]
  [13 17 21]]

 [[ 2  6 10]
  [14 18 22]]

 [[ 3  7 11]
  [15 19 23]]]
(4, 2, 3)
```

Or,

```
a=np.arange(24).reshape(2,3,4)
print(a)
print('\n')
print(np.transpose(a,axes=(1,0,2)))
print(np.transpose(a,axes=(1,0,2)).shape)
```

o/p:

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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]


[[[ 0  1  2  3]
  [12 13 14 15]]

 [[ 4  5  6  7]
  [16 17 18 19]]

 [[ 8  9 10 11]
  [20 21 22 23]]]
(3, 2, 4)
```

np.swapaxes:

```
a=np.arange(1,5).reshape(2,2)
print(a)
print('\n')
print(np.swapaxes(a,1,0))
```

o/p:

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


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

Or,

```
a=np.arange(1,5).reshape(2,2)
print(a)
print('\n')
print(np.swapaxes(a,0,1))
```

o/p:

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


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

Or,

```
a=np.array([[1,2,3]])
print(a)
print('\n')
print(np.swapaxes(a,1,0))
print('\n')
print(a.T)
```

o/p:

```
[[1 2 3]]


[[1]
 [2]
 [3]]


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

Or,,

```
a=np.arange(0,40).reshape(2,4,5)#here axis 0=2,axis 1=4 and axis 2=5
print(a)
print('\n')
print('interchage axis 1 and axis 2 by swapaxes','\n',np.swapaxes(a,1,2))
print(np.swapaxes(a,1,2).shape)
print('\n')
print('interchage axis 0 and axis 2 by swap axes','\n',np.swapaxes(a,0,2))
print(np.swapaxes(a,0,2).shape)
print('\n')
print('interchage axis 0 and axis 2 by
transpose','\n',np.transpose(a,axes=(2,1,0)))
print(np.transpose(a,axes=(2,1,0)).shape)
```

o/p:

```
[[[ 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]]]


interchage axis 1 and axis 2 by swapaxes
 [[[ 0  5 10 15]
```

```
  [ 1  6 11 16]
  [ 2  7 12 17]
  [ 3  8 13 18]
  [ 4  9 14 19]]

 [[20 25 30 35]
  [21 26 31 36]
  [22 27 32 37]
  [23 28 33 38]
  [24 29 34 39]]]
(2, 5, 4)


interchage axis 0 and axis 2 by swap axes
 [[[ 0 20]
  [ 5 25]
  [10 30]
  [15 35]]

 [[ 1 21]
  [ 6 26]
  [11 31]
  [16 36]]

 [[ 2 22]
  [ 7 27]
  [12 32]
  [17 37]]

 [[ 3 23]
  [ 8 28]
  [13 33]
  [18 38]]

 [[ 4 24]
  [ 9 29]
  [14 34]
  [19 39]]]
(5, 4, 2)


interchage axis 0 and axis 2 by transpose
 [[[ 0 20]
  [ 5 25]
  [10 30]
  [15 35]]

 [[ 1 21]
  [ 6 26]
  [11 31]
  [16 36]]

 [[ 2 22]
  [ 7 27]
  [12 32]
  [17 37]]

 [[ 3 23]
  [ 8 28]
  [13 33]
  [18 38]]

 [[ 4 24]
  [ 9 29]
  [14 34]
  [19 39]]]
(5, 4, 2)
```

The difference between transpose and swapaxes lies in swapaxes we can interchange only two axis but in transpose we can interchange many axisr

Or,

```
import numpy as np
a=np.arange(8).reshape(2,2,2)
print(a)
print('\n')
print(np.swapaxes(a,2,0))
print('\n')
print(np.swapaxes(a,1,2))
print('\n')
print(np.swapaxes(a,0,1))
```

o/p:

```
[[[0 1]
  [2 3]]

 [[4 5]
  [6 7]]]


[[[0 4]
  [2 6]]

 [[1 5]
  [3 7]]]


[[[0 2]
  [1 3]]

 [[4 6]
  [5 7]]]
```

```
[[[0 1]
  [4 5]]

 [[2 3]
  [6 7]]]
```

Or,

```
import numpy as np
a=np.arange(24).reshape(2,3,4)
print(a)
print('\n')
print(np.swapaxes(a,2,0))
print('\n')
print(np.swapaxes(a,1,2))
print('\n')
print(np.swapaxes(a,0,1))
```

o/p:

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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]


[[[ 0 12]
  [ 4 16]
  [ 8 20]]

 [[ 1 13]
  [ 5 17]
  [ 9 21]]

 [[ 2 14]
  [ 6 18]
  [10 22]]

 [[ 3 15]
  [ 7 19]
  [11 23]]]


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

 [[12 16 20]
  [13 17 21]
  [14 18 22]
  [15 19 23]]]


[[[ 0  1  2  3]
  [12 13 14 15]]

 [[ 4  5  6  7]
  [16 17 18 19]]

 [[ 8  9 10 11]
  [20 21 22 23]]]
```

<u>np.rollaxis:</u>

```
import numpy as np
a=np.arange(60).reshape(3,4,5)
print(a)
print('\n')
print(np.rollaxis(a,2))
print(np.rollaxis(a,2).shape)
print('\n')
print(np.rollaxis(a,1))
print(np.rollaxis(a,1).shape)
```

or,,

```
[[[ 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]]]


[[[ 0  5 10 15]
  [20 25 30 35]
  [40 45 50 55]]

 [[ 1  6 11 16]
  [21 26 31 36]
  [41 46 51 56]]

 [[ 2  7 12 17]
  [22 27 32 37]
  [42 47 52 57]]

 [[ 3  8 13 18]
  [23 28 33 38]
  [43 48 53 58]]

 [[ 4  9 14 19]
  [24 29 34 39]
  [44 49 54 59]]]
(5, 3, 4)


[[[ 0  1  2  3  4]
  [20 21 22 23 24]
  [40 41 42 43 44]]

 [[ 5  6  7  8  9]
  [25 26 27 28 29]
  [45 46 47 48 49]]

 [[10 11 12 13 14]
  [30 31 32 33 34]
  [50 51 52 53 54]]

 [[15 16 17 18 19]
  [35 36 37 38 39]
  [55 56 57 58 59]]]
(4, 3, 5)
```

Or,

```
import numpy as np
a=np.arange(60).reshape(3,4,5)
print(a)
print('\n')
print('roll axis(a,2,1)','\n',np.rollaxis(a,2,1))
print('swap axis(a,2,1)','\n',np.swapaxes(a,2,1))
print('roll axis(a,2,1).shape','\n',np.rollaxis(a,2,1).shape)
print('swap axis(a,2,1).shape','\n',np.swapaxes(a,2,1).shape)
print('\n')
print('roll axis(a,1,2)','\n',np.rollaxis(a,1,2))
print('swap axis(a,1,2)','\n',np.swapaxes(a,1,2))
print('roll axis(a,1,2).shape','\n',np.rollaxis(a,1,2).shape)
print('swap axis(a,1,2).shape','\n',np.swapaxes(a,1,2).shape)
```

o/p:

```
[[[ 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]]]


roll axis(a,2,1)
 [[[ 0  5 10 15]
  [ 1  6 11 16]
  [ 2  7 12 17]
  [ 3  8 13 18]
  [ 4  9 14 19]]

 [[20 25 30 35]
  [21 26 31 36]
  [22 27 32 37]
  [23 28 33 38]
  [24 29 34 39]]

 [[40 45 50 55]
  [41 46 51 56]
  [42 47 52 57]
  [43 48 53 58]
  [44 49 54 59]]]
swap axis(a,2,1)
 [[[ 0  5 10 15]
  [ 1  6 11 16]
  [ 2  7 12 17]
  [ 3  8 13 18]
  [ 4  9 14 19]]

 [[20 25 30 35]
  [21 26 31 36]
  [22 27 32 37]
  [23 28 33 38]
  [24 29 34 39]]

 [[40 45 50 55]
  [41 46 51 56]
  [42 47 52 57]
  [43 48 53 58]
  [44 49 54 59]]]
roll axis(a,2,1).shape
 (3, 5, 4)
swap axis(a,2,1).shape
 (3, 5, 4)


roll axis(a,1,2)
 [[[ 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]]]
swap axis(a,1,2)
 [[[ 0  5 10 15]
  [ 1  6 11 16]
  [ 2  7 12 17]
  [ 3  8 13 18]
  [ 4  9 14 19]]

 [[20 25 30 35]
  [21 26 31 36]
  [22 27 32 37]
  [23 28 33 38]
  [24 29 34 39]]

 [[40 45 50 55]
  [41 46 51 56]
  [42 47 52 57]
  [43 48 53 58]
  [44 49 54 59]]]
roll axis(a,1,2).shape
 (3, 4, 5)
swap axis(a,1,2).shape
 (3, 5, 4)
```

Or,

```
import numpy as np
a=np.arange(60).reshape(3,4,5)
print(a)
print('\n')
print(np.rollaxis(a,2,0))
print(np.rollaxis(a,2,0).shape)
print('\n')
print(np.rollaxis(a,1,0))
print(np.rollaxis(a,1,0).shape)
```

o/p:

```
[[[ 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]]]


[[[ 0  5 10 15]
  [20 25 30 35]
  [40 45 50 55]]

 [[ 1  6 11 16]
  [21 26 31 36]
  [41 46 51 56]]

 [[ 2  7 12 17]
  [22 27 32 37]
  [42 47 52 57]]

 [[ 3  8 13 18]
  [23 28 33 38]
  [43 48 53 58]]

 [[ 4  9 14 19]
  [24 29 34 39]
  [44 49 54 59]]]
(5, 3, 4)


[[[ 0  1  2  3  4]
  [20 21 22 23 24]
  [40 41 42 43 44]]

 [[ 5  6  7  8  9]
  [25 26 27 28 29]
  [45 46 47 48 49]]

 [[10 11 12 13 14]
  [30 31 32 33 34]
  [50 51 52 53 54]]

 [[15 16 17 18 19]
  [35 36 37 38 39]
  [55 56 57 58 59]]]
(4, 3, 5)
```

For a matrix

A=np.arange(1,25).reshape(2,3,4) ,

Depth/axis0=2

Height/axis1=3

Depth/axis2=4

Or,

```
import numpy as np

a=np.arange(60).reshape(3,4,5)

print(a)

print('\n')

print('roll axis(a,2,1)','\n',np.rollaxis(a,2,1))

print('swap axis(a,2,1)','\n',np.swapaxes(a,2,1))

print('roll axis(a,2,1).shape','\n',np.rollaxis(a,2,1).shape)

print('swap axis(a,2,1).shape','\n',np.swapaxes(a,2,1).shape)

print('\n')

print('roll axis(a,1,2)','\n',np.rollaxis(a,1,2))

print('swap axis(a,1,2)','\n',np.swapaxes(a,1,2))

print('roll axis(a,1,2).shape','\n',np.rollaxis(a,1,2).shape)

print('swap axis(a,1,2).shape','\n',np.swapaxes(a,1,2).shape)

print('roll axis(a,0,1)','\n',np.rollaxis(a,0,1))

print('swap axis(a,0,1)','\n',np.swapaxes(a,0,1))
```

o/p:

```
[[[ 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]]]


roll axis(a,2,1)
 [[[ 0  5 10 15]
  [ 1  6 11 16]
  [ 2  7 12 17]
  [ 3  8 13 18]
  [ 4  9 14 19]]

 [[20 25 30 35]
  [21 26 31 36]
  [22 27 32 37]
  [23 28 33 38]
  [24 29 34 39]]

 [[40 45 50 55]
  [41 46 51 56]
  [42 47 52 57]
  [43 48 53 58]
  [44 49 54 59]]]
swap axis(a,2,1)
 [[[ 0  5 10 15]
  [ 1  6 11 16]
  [ 2  7 12 17]
  [ 3  8 13 18]
  [ 4  9 14 19]]

 [[20 25 30 35]
  [21 26 31 36]
  [22 27 32 37]
  [23 28 33 38]
  [24 29 34 39]]

 [[40 45 50 55]
  [41 46 51 56]
  [42 47 52 57]
  [43 48 53 58]
  [44 49 54 59]]]
roll axis(a,2,1).shape
 (3, 5, 4)
swap axis(a,2,1).shape
 (3, 5, 4)
```

```
roll axis(a,1,2)
 [[[ 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]]]
swap axis(a,1,2)
 [[[ 0  5 10 15]
  [ 1  6 11 16]
  [ 2  7 12 17]
  [ 3  8 13 18]
  [ 4  9 14 19]]

 [[20 25 30 35]
  [21 26 31 36]
  [22 27 32 37]
  [23 28 33 38]
  [24 29 34 39]]

 [[40 45 50 55]
  [41 46 51 56]
  [42 47 52 57]
  [43 48 53 58]
  [44 49 54 59]]]
roll axis(a,1,2).shape
 (3, 4, 5)
swap axis(a,1,2).shape
 (3, 5, 4)
roll axis(a,0,1)
 [[[ 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]]]
swap axis(a,0,1)
 [[[ 0  1  2  3  4]
```

```
  [20 21 22 23 24]
  [40 41 42 43 44]]

 [[ 5  6  7  8  9]
  [25 26 27 28 29]
  [45 46 47 48 49]]

 [[10 11 12 13 14]
  [30 31 32 33 34]
  [50 51 52 53 54]]

 [[15 16 17 18 19]
  [35 36 37 38 39]
  [55 56 57 58 59]]]
```

Or,

```
import numpy as np

a=np.arange(8).reshape(2,2,2)

print(a)

print('\n')

print(np.rollaxis(a,2))

print('\n')

print(np.rollaxis(a,2,1))
```

o/p:

```
[[[0 1]
  [2 3]]

 [[4 5]
  [6 7]]]


[[[0 2]
  [4 6]]

 [[1 3]
  [5 7]]]
```

```
[[[0 2]
  [1 3]]

 [[4 6]
  [5 7]]]
```

np.broadcast to:

```
import numpy as np
x=np.arange(4).reshape(1,4)
print(x)
print('\n')
print(np.broadcast_to(x,(4,4)))
```

o/p:

```
[[0 1 2 3]]


[[0 1 2 3]
 [0 1 2 3]
 [0 1 2 3]
 [0 1 2 3]]
```

Or,

```
import numpy as np

x=np.arange(4).reshape(1,4)

print(x)

print('\n')

print(np.broadcast_to(x,(3,4)))
```

o/p:

```
[[0 1 2 3]]


[[0 1 2 3]
 [0 1 2 3]
 [0 1 2 3]]
```

Or,

```
import numpy as np

x=np.arange(4).reshape(1,4)

print(x)

print('\n')

print(np.broadcast_to(x,(4,3)))
```

o/p,

```
value error:

operands could not be broadcast together with remapped shapes [original-
>remapped]: (1,4) and requested shape (4,3)
```

or,

```
import numpy as np
x=np.arange(4).reshape(4,1)
print(x)
print('\n')
print(np.broadcast_to(x,(4,3)))
```

o/p:

```
[[0]
 [1]
 [2]
 [3]]


[[0 0 0]
 [1 1 1]
 [2 2 2]
 [3 3 3]]
```

Or,

```
import numpy as np
x=np.arange(6).reshape(3,2)
print(x)
print('\n')
print(np.broadcast_to(x,(4,2)))
```

o/p:

```
ValueError: operands could not be broadcast together with remapped shapes
[original->remapped]: (3,2) and requested shape (4,2)
```

np.expand dims:

```
import numpy as np
x=np.array([[1,2],[3,4]])
print(x)
print('\n')
y=np.expand_dims(x,axis=0)
print(y)
print('\n')
print('shape of x array',x.shape)
print('shape of y array',y.shape)
print('dim of x array',x.ndim)
print('dim of y array',y.ndim)
```

o/p:

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


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


shape of x array (2, 2)
shape of y array (1, 2, 2)
dim of x array 2
dim of y array 3
```

or,

```
import numpy as np
x=np.array([[1,2],[3,4]])
print(x)
print('\n')
y=np.expand_dims(x,axis=1)
print(y)
print('\n')
print('shape of x array',x.shape)
print('shape of y array',y.shape)
print('dim of x array',x.ndim)
print('dim of y array',y.ndim)
```

o/p:

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


[[[1 2]]
```

```
 [[3 4]]]


shape of x array (2, 2)
shape of y array (2, 1, 2)
dim of x array 2
dim of y array 3
```

or,

```
import numpy as np
x=np.array([[1,2],[3,4]])
print(x)
print('\n')
y=np.expand_dims(x,axis=2)
print(y)
print('\n')
print('shape of x array',x.shape)
print('shape of y array',y.shape)
print('dim of x array',x.ndim)
print('dim of y array',y.ndim)
```

o/p:

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


[[[1]
  [2]]

 [[3]
  [4]]]


shape of x array (2, 2)
shape of y array (2, 2, 1)
dim of x array 2
dim of y array 3
```

or,

```
import numpy as np
x=np.arange(60).reshape(3,4,5)
print(x)
print('\n')
y=np.expand_dims(x,axis=2)
print(y)
print('\n')
print('shape of x array',x.shape)
print('shape of y array',y.shape)
print('dim of x array',x.ndim)
print('dim of y array',y.ndim)
```

o/p:

```
[[[ 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]]]


[[[[ 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]]]]


shape of x array (3, 4, 5)
shape of y array (3, 4, 1, 5)
dim of x array 3
dim of y array 4
```

or,

```
import numpy as np

x=np.arange(60).reshape(3,4,5)

print(x)

print('\n')

y=np.expand_dims(x,axis=1)

print(y)

print('\n')

print('shape of x array',x.shape)

print('shape of y array',y.shape)

print('dim of x array',x.ndim)

print('dim of y array',y.ndim)
```

o/p:

```
[[[ 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]]]


[[[[ 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]]]]


shape of x array (3, 4, 5)
shape of y array (3, 1, 4, 5)
dim of x array 3
dim of y array 4
```

or,

```
import numpy as np

x=np.arange(60).reshape(3,4,5)

print(x)

print('\n')

y=np.expand_dims(x,axis=0)

print(y)

print('\n')

print('shape of x array',x.shape)

print('shape of y array',y.shape)

print('dim of x array',x.ndim)

print('dim of y array',y.ndim)
```

o/p:

```
[[[ 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]]]


[[[[ 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]]]]


shape of x array (3, 4, 5)
shape of y array (1, 3, 4, 5)
dim of x array 3
dim of y array 4
```

or,

```
import numpy as np

x=np.arange(60).reshape(3,4,5)

print(x)

print('\n')

y=np.expand_dims(x,axis=3)

print(y)

print('\n')

print('shape of x array',x.shape)
print('shape of y array',y.shape)

print('dim of x array',x.ndim)

print('dim of y array',y.ndim)
```

o/p

```
[[[ 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]]]


[[[[ 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]]]]


shape of x array (3, 4, 5)
shape of y array (3, 4, 5, 1)
dim of x array 3
dim of y array 4
```

np.squeeze:

```
import numpy as np
x=np.arange(9).reshape(1,3,3)
print(x)
print('\n')
y=np.squeeze(x,axis=0)
print( 'after squeeze array','\n',y)
```

o/p:

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


after squeeze array
 [[0 1 2]
 [3 4 5]
 [6 7 8]]
```

Or,

```
import numpy as np
x=np.arange(9).reshape(1,3,3)
print(x)
print('\n')
y=np.squeeze(x,axis=1)
print( 'after squeeze array','\n',y)
```

o/p:

```
ValueError: cannot select an axis to squeeze out which has size not equal
to one
```

Or,

```
import numpy as np
x=np.arange(9).reshape(3,1,3)
print(x)
print('\n')
y=np.squeeze(x,axis=1)
print( 'after squeeze array','\n',y)
```

o/p:

```
[[[0 1 2]]

 [[3 4 5]]

 [[6 7 8]]]
```

```
after squeeze array
 [[0 1 2]
 [3 4 5]
 [6 7 8]]
```

Or,

```
import numpy as np
x=np.arange(18).reshape(3,2,3)
print(x)
print('\n')
y=np.squeeze(x,axis=1)
print( 'after squeeze array','\n',y)
```

o/p:

```
[[[ 0  1  2]
  [ 3  4  5]]

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

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


---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-17-97f567f3e861> in <module>
      3 print(x)
      4 print('\n')
----> 5 y=np.squeeze(x,axis=1)
      6 print( 'after squeeze array','\n',y)

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

~\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py in squeeze(a, axis)
   1481         return squeeze()
   1482     else:
-> 1483         return squeeze(axis=axis)
   1484
   1485

ValueError: cannot select an axis to squeeze out which has size not equal to
one
```

np.concatenate:

```
import numpy as np
a=np.arange(6).reshape(2,3)
print(a)
print('\n')
b=np.arange(6).reshape(2,3)
print(b)
print('\n')
c=np.concatenate((a,b),axis=0)
```

O/P:

```
[[0 1 2]
 [3 4 5]]


[[0 1 2]
 [3 4 5]]


[[0 1 2]
 [3 4 5]
 [0 1 2]
 [3 4 5]]
```

OR:

```
import numpy as np
a=np.arange(6).reshape(2,3)
print(a)
print('\n')
b=np.arange(6).reshape(2,3)
print(b)
print('\n')
c=np.concatenate((a,b),axis=1)
print(c)
```

o/p:

```
[[0 1 2]
 [3 4 5]]


[[0 1 2]
 [3 4 5]]


[[0 1 2 0 1 2]
 [3 4 5 3 4 5]]
```

Or,

```
import numpy as np
```

```
a=np.arange(6).reshape(2,3)
print(a)
print('\n')
b=np.arange(12).reshape(4,3)
print(b)
print('\n')
c=np.concatenate((a,b),axis=0)
print(c)
```

o/p:

```
[[0 1 2]
 [3 4 5]]


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


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

Or,

```
import numpy as np
a=np.arange(6).reshape(2,3)
print(a)
print('\n')
b=np.arange(12).reshape(4,3)
print(b)
print('\n')
c=np.concatenate((a,b),axis=1)
print(c)
```

o/p:

```
ValueError: all the input array dimensions for the concatenation axis must
match exactly, but along dimension 0, the array at index 0 has size 2 and
the array at index 1 has size 4
```

Or,,

```
import numpy as np
a=np.arange(6).reshape(2,3)
print(a)
print('\n')
```

```
b=np.arange(8).reshape(2,4)
print(b)
print('\n')
c=np.concatenate((a,b),axis=1)
print(c)

o/p:

[[0 1 2]
 [3 4 5]]


[[0 1 2 3]
 [4 5 6 7]]


[[0 1 2 0 1 2 3]
 [3 4 5 4 5 6 7]]


Or,

import numpy as np
a=np.arange(60).reshape(3,4,5)
print(a)
print('\n')
b=np.arange(72).reshape(3,4,6)
print(b)
print('\n')
c=np.concatenate((a,b),axis=0)
print(c)

o/p:

ValueError: all the input array dimensions for the concatenation axis must
match exactly, but along dimension 2, the array at index 0 has size 5 and
the array at index 1 has size 6


Or,,

import numpy as np
a=np.arange(60).reshape(3,4,5)
print(a)
print('\n')
b=np.arange(20,80).reshape(3,4,5)
print(b)
print('\n')
c=np.concatenate((a,b),axis=0)
print(c)

o/p:

[[[ 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]]]


[[[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]]]


[[[ 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]]

 [[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]]]
```

Or,

```
import numpy as np
a=np.arange(60).reshape(3,4,5)
print(a)
print('\n')
b=np.arange(20,80).reshape(3,4,5)
print(b)
print('\n')
c=np.concatenate((a,b),axis=1)
print(c)
```

o/p:

```
[[[ 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]]]


[[[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]]]
```

```
[[[ 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]]

 [[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]]

 [[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]]]
```

Or,

```
import numpy as np
a=np.arange(60).reshape(3,4,5)
print(a)
print('\n')
b=np.arange(20,80).reshape(3,4,5)
print(b)
print('\n')
c=np.concatenate((a,b),axis=2)
print(c)
```

o/p:

```
[[[ 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]]]


[[[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]]]


[[[ 0  1  2  3  4 20 21 22 23 24]
  [ 5  6  7  8  9 25 26 27 28 29]
  [10 11 12 13 14 30 31 32 33 34]
  [15 16 17 18 19 35 36 37 38 39]]

 [[20 21 22 23 24 40 41 42 43 44]
  [25 26 27 28 29 45 46 47 48 49]
  [30 31 32 33 34 50 51 52 53 54]
  [35 36 37 38 39 55 56 57 58 59]]

 [[40 41 42 43 44 60 61 62 63 64]
  [45 46 47 48 49 65 66 67 68 69]
  [50 51 52 53 54 70 71 72 73 74]
  [55 56 57 58 59 75 76 77 78 79]]]
```

np.stack:

```
import numpy as np
a=np.arange(12).reshape(3,4)
print(a)
print('\n')
b=np.arange(12,24).reshape(3,4)
print(b)
print('\n')
print(np.stack((a,b),0))
print(np.stack((a,b),0).shape)
```

o/p:

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

```
[[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]


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

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
(2, 3, 4)
```

Or,

```
import numpy as np
a=np.arange(12).reshape(3,4)
print(a)
print('\n')
b=np.arange(12,24).reshape(3,4)
print(b)
print('\n')
print(np.stack((a,b),1))
print(np.stack((a,b),1).shape)
```

o/p:

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


[[12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]


[[[ 0  1  2  3]
  [12 13 14 15]]

 [[ 4  5  6  7]
  [16 17 18 19]]

 [[ 8  9 10 11]
  [20 21 22 23]]]
(3, 2, 4)
```

Or,,

```
import numpy as np
a=np.arange(6).reshape(2,3)
```

```
print(a)
print('\n')
b=np.arange(12).reshape(4,3)
print(b)
print('\n')
print(np.stack((a,b),0))
```

```
o/p:

[[0 1 2]
 [3 4 5]]



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



---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-36-1389ce410f0a> in <module>
      6 print(b)
      7 print('\n')
----> 8 print(np.stack((a,b),0))

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

~\Anaconda3\lib\site-packages\numpy\core\shape_base.py in stack(arrays, axis,
out)
    424     shapes = {arr.shape for arr in arrays}
    425     if len(shapes) != 1:
--> 426         raise ValueError('all input arrays must have the same shape')
    427 
    428     result_ndim = arrays[0].ndim + 1

ValueError: all input arrays must have the same shape
```

Or,

```
import numpy as np
x=np.arange(60).reshape(3,4,5)
print(x)
print('\n')
y=np.arange(60,120).reshape(3,4,5)
```

```
print(y)
print('\n')
print('after stack','\n',np.stack((x,y),0))
print('ahter stack',np.stack((x,y),0).shape)

o/p:

[[[ 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  99]]

 [[100 101 102 103 104]
  [105 106 107 108 109]
  [110 111 112 113 114]
  [115 116 117 118 119]]]


after stack
 [[[[  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  99]]

  [[100 101 102 103 104]
   [105 106 107 108 109]
   [110 111 112 113 114]
   [115 116 117 118 119]]]]
ahter stack (2, 3, 4, 5)
```

or,

```
import numpy as np

x=np.arange(60).reshape(3,4,5)

print(x)

print('\n')

y=np.arange(60,120).reshape(3,4,5)

print(y)

print('\n')

print('after stack','\n',np.stack((x,y),1))

print('ahter stack',np.stack((x,y),1).shape)
```

o/p:

```
[[[ 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  99]]

 [[100 101 102 103 104]
  [105 106 107 108 109]
  [110 111 112 113 114]
  [115 116 117 118 119]]]


after stack
 [[[[  0   1   2   3   4]
   [  5   6   7   8   9]
   [ 10  11  12  13  14]
   [ 15  16  17  18  19]]

  [[ 60  61  62  63  64]
   [ 65  66  67  68  69]
   [ 70  71  72  73  74]
   [ 75  76  77  78  79]]]


 [[[ 20  21  22  23  24]
   [ 25  26  27  28  29]
   [ 30  31  32  33  34]
   [ 35  36  37  38  39]]

  [[ 80  81  82  83  84]
   [ 85  86  87  88  89]
   [ 90  91  92  93  94]
   [ 95  96  97  98  99]]]


 [[[ 40  41  42  43  44]
   [ 45  46  47  48  49]
   [ 50  51  52  53  54]
   [ 55  56  57  58  59]]

  [[100 101 102 103 104]
   [105 106 107 108 109]
   [110 111 112 113 114]
   [115 116 117 118 119]]]]
ahter stack (3, 2, 4, 5)
```

or,

```
import numpy as np

x=np.arange(60).reshape(3,4,5)

print(x)

print('\n')

y=np.arange(60,120).reshape(3,4,5)

print(y)

print('\n')

print('after stack','\n',np.stack((x,y),2))

print('ahter stack',np.stack((x,y),2).shape)
```

o/p:

```
[[[ 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  99]]

 [[100 101 102 103 104]
  [105 106 107 108 109]
  [110 111 112 113 114]
  [115 116 117 118 119]]]


after stack
 [[[[  0   1   2   3   4]
   [ 60  61  62  63  64]]

  [[  5   6   7   8   9]
   [ 65  66  67  68  69]]

  [[ 10  11  12  13  14]
   [ 70  71  72  73  74]]

  [[ 15  16  17  18  19]
   [ 75  76  77  78  79]]]


 [[[ 20  21  22  23  24]
   [ 80  81  82  83  84]]

  [[ 25  26  27  28  29]
   [ 85  86  87  88  89]]

  [[ 30  31  32  33  34]
   [ 90  91  92  93  94]]

  [[ 35  36  37  38  39]
   [ 95  96  97  98  99]]]


 [[[ 40  41  42  43  44]
   [100 101 102 103 104]]

  [[ 45  46  47  48  49]
   [105 106 107 108 109]]

  [[ 50  51  52  53  54]
   [110 111 112 113 114]]

  [[ 55  56  57  58  59]
   [115 116 117 118 119]]]]
ahter stack (3, 4, 2, 5)
```

np.hstack and np.vstack

```
import numpy as np

x=np.arange(24).reshape(4,6)

print(x)

print('\n')

y=np.arange(24,48).reshape(4,6)

print(y)

print('\n')

print('hstack','\n',np.hstack((x,y)))

print('np.concatenate((x,y),axis=1)','\n',np.concatenate((x,y),axis=1))

print('vstack','\n',np.vstack((x,y)))

print('np.concatenate((x,y),axis=0)','\n',np.concatenate((x,y),axis=0))
```

o/p:

```
[[ 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]]


hstack
 [[ 0  1  2  3  4  5 24 25 26 27 28 29]
 [ 6  7  8  9 10 11 30 31 32 33 34 35]
 [12 13 14 15 16 17 36 37 38 39 40 41]
 [18 19 20 21 22 23 42 43 44 45 46 47]]
np.concatenate((x,y),axis=1)
 [[ 0  1  2  3  4  5 24 25 26 27 28 29]
 [ 6  7  8  9 10 11 30 31 32 33 34 35]
 [12 13 14 15 16 17 36 37 38 39 40 41]
 [18 19 20 21 22 23 42 43 44 45 46 47]]
vstack
```

```
 [[ 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]]
np.concatenate((x,y),axis=0)
 [[ 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]]
```

np.split

```
import numpy as np

x=np.arange(12)

print(x)

print('\n')

y=np.split(x,3)

print(y)

print('\n')

z=np.split(x,[4,7])

print(z)
```

o/p:

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


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


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

Or,

```
import numpy as np

x=np.arange(12).reshape(3,4)

print(x)

print('\n')

y=np.split(x,1)

print(y)
```

o/p:

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


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

Or:

```
import numpy as np

x=np.arange(12).reshape(3,4)

print(x)

print('\n')

y=np.split(x,2)

print(y)
```

o/p:

```
ValueError: array split does not result in an equal division
```

Or,

```
import numpy as np

x=np.arange(12).reshape(3,4)

print(x)

print('\n')

y=np.split(x,3)

print(y)
```

o/p:

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


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

Or,

```
import numpy as np

x=np.arange(12).reshape(3,4)

print(x)

print('\n')

y=np.split(x,[4,7])

print(y)
```

o/p:

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


[array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]]), array([], shape=(0, 4), dtype=int32), array([],
shape=(0, 4), dtype=int32)]
```

Or,

```
import numpy as np

x=np.arange(60).reshape(3,4,5)

print(x)

print('\n')

y=np.split(x,3)

print(y)
```

o/p:

```
[[[ 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]]]


[array([[[ 0,  1,  2,  3,  4],
        [ 5,  6,  7,  8,  9],
        [10, 11, 12, 13, 14],
        [15, 16, 17, 18, 19]]]), array([[[20, 21, 22, 23, 24],
        [25, 26, 27, 28, 29],
        [30, 31, 32, 33, 34],
        [35, 36, 37, 38, 39]]]), array([[[40, 41, 42, 43, 44],
        [45, 46, 47, 48, 49],
        [50, 51, 52, 53, 54],
        [55, 56, 57, 58, 59]]])]
```

Or,

```
import numpy as np

x=np.arange(60).reshape(3,4,5)

print(x)

print('\n')

y=np.split(x,4)

print(y)
```

o/p:

```
[[[ 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]]]

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\Anaconda3\lib\site-packages\numpy\lib\shape_base.py in split(ary,
indices_or_sections, axis)
```

```
    864     try:
--> 865         len(indices_or_sections)
    866     except TypeError:

TypeError: object of type 'int' has no len()

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-44-4679b9b1d32f> in <module>
      3 print(x)
      4 print('\n')
----> 5 y=np.split(x,4)
      6 print(y)

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

~\Anaconda3\lib\site-packages\numpy\lib\shape_base.py in split(ary,
indices_or_sections, axis)
    869         if N % sections:
    870             raise ValueError(
--> 871                 'array split does not result in an equal division')
    872     return array_split(ary, indices_or_sections, axis)
    873

ValueError: array split does not result in an equal division
```

Or,

```
import numpy as np

x=np.arange(60).reshape(3,4,5)

print(x)

print('\n')

y=np.split(x,1)

print(y)
```

o/p:\

```
[[[ 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]]]


[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]]])]
```

np.hsplit and np.vsplit

```
import numpy as np

a=np.arange(16).reshape(4,4)

print(a)

print('\n')

print(np.hsplit(a,2))

print('\n')

print(np.vsplit(a,2))
```

o/p:

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]


[array([[ 0,  1],
       [ 4,  5],
       [ 8,  9],
       [12, 13]]), array([[ 2,  3],
       [ 6,  7],
       [10, 11],
       [14, 15]])]


[array([[0, 1, 2, 3],
       [4, 5, 6, 7]]), array([[ 8,  9, 10, 11],
       [12, 13, 14, 15]])]
```

Or,

```
import numpy as np

a=np.arange(16).reshape(3,4)

print(a)

print('\n')

print(np.hsplit(a,2))

print('\n')

print(np.vsplit(a,2))
```

o/p:

```
ValueError: cannot reshape array of size 16 into shape (3,4)
```

Or,,

```
import numpy as np

a=np.arange(16).reshape(4,3)

print(a)

print('\n')

print(np.hsplit(a,2))

print('\n')

print(np.vsplit(a,2))
```

o/p:

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call
last)
<ipython-input-9-a2c1bb62aac2> in <module>
      1 import numpy as np
----> 2 a=np.arange(16).reshape(4,3)
      3 print(a)
      4 print('\n')
      5 #print(np.hsplit(a,2))

ValueError: cannot reshape array of size 16 into shape (4,3)
```

Or,

```
import numpy as np

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

print(a)

print('\n')

y=np.resize(a,(2,3))

print('after resize','\n',y)
```

o/p:

```
[[0 1]
 [2 3]
 [4 5]]


after resize
 [[0 1 2]
 [3 4 5]]
```

Or,

```
import numpy as np
a=np.arange(60).reshape(3,4,5)
print(a)
print('\n')
y=np.resize(a,(5,4,3))
print('after resize','\n',y)
```

o/p:

```
[[[ 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]]]


after resize
 [[[ 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]]]
```

Or,

```
import numpy as np
a=np.arange(60).reshape(3,4,5)
print(a)
print('\n')
y=np.resize(a,(5,6,2))
print('after resize','\n',y)
```

o/p,

```
[[[ 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]]]


after resize
 [[[ 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]]]
```

Or,

```
import numpy as np
a=np.arange(6).reshape(3,2)
print(a)
print('\n')
y=np.resize(a,(3,3))
print('after resize','\n',y)
```

o/p:

```
[[0 1]
 [2 3]
 [4 5]]


after resize
 [[0 1 2]
 [3 4 5]
 [0 1 2]]
```

Or,

```
import numpy as np
a=np.arange(6).reshape(3,2)
print(a)
print('\n')
y=np.resize(a,(3,4))
print('after resize','\n',y)
```

o/p:

```
[[0 1]
 [2 3]
 [4 5]]


after resize
 [[0 1 2 3]
 [4 5 0 1]
 [2 3 4 5]]
```

Or,

```
import numpy as np
a=np.arange(6).reshape(3,2)
print(a)
print('\n')
y=np.resize(a,(4,4))
print('after resize','\n',y)
```

o/p:

```
[[0 1]
 [2 3]
 [4 5]]


after resize
 [[0 1 2 3]
 [4 5 0 1]
 [2 3 4 5]
 [0 1 2 3]]
```

Or,

```
import numpy as np
a=np.arange(6).reshape(3,2)
print(a)
print('\n')
y=np.resize(a,(4,3))
print('after resize','\n',y)
```

o/p:

```
[[0 1]
 [2 3]
 [4 5]]


after resize
 [[0 1 2]
 [3 4 5]
 [0 1 2]
 [3 4 5]]
```

Or,

```
import numpy as np
a=np.arange(6).reshape(3,2)
print(a)
print('\n')
y=np.resize(a,(4,6))
print('after resize','\n',y)
```

o/p:

```
[[0 1]
 [2 3]
 [4 5]]



after resize
 [[0 1 2 3 4 5]
 [0 1 2 3 4 5]
 [0 1 2 3 4 5]
 [0 1 2 3 4 5]]
```

In [ ]:

Or,

```
import numpy as np
a=np.arange(60).reshape(3,4,5)
print(a)
print('\n')
y=np.resize(a,(4,3,2,3))
print('after resize','\n',y)
```

o/p:

```
[[[ 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]]]


after resize
 [[[[ 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]]

  [[ 0  1  2]
   [ 3  4  5]]

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

np.append

```
import numpy as np
a=np.arange(6).reshape(2,3)
print(a)
print('\n')
y=np.append(a,[7,8,9])
print(y)
```

o/p:

```
[[0 1 2]
 [3 4 5]]
```

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

Or,

```
import numpy as np
a=np.arange(6).reshape(2,3)
print(a)
print('\n')
y=np.append(a,[[7,8,9]],axis=0)
print(y)
```

o/p:

```
[[0 1 2]
 [3 4 5]]


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

Or,

```
import numpy as np
a=np.arange(6).reshape(2,3)
print(a)
print('\n')
y=np.append(a,[[7,8,9]],axis=1)
print(y)
```

o/p:

```
ValueError: all the input array dimensions for the concatenation axis must
match exactly, but along dimension 0, the array at index 0 has size 2 and
the array at index 1 has size 1
```

Or,

```
import numpy as np
a=np.arange(6).reshape(2,3)
print(a)
print('\n')
y=np.append(a,[[7,8,9],[5,6,4]],axis=1)
print(y)
```

o/p:

```
[[0 1 2]
 [3 4 5]]


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

np.insert:

```
import numpy as np

x=np.arange(6).reshape(3,2)

print(x)

print('\n')

y=np.insert(x,3,[11,12])

print('after insert','\n',y)
```

o/p:

```
[[0 1]
 [2 3]
 [4 5]]


after insert
 [ 0  1  2 11 12  3  4  5]
```

In [ ]:

Or,

```
import numpy as np

x=np.arange(6).reshape(3,2)

print(x)

print('\n')

y=np.insert(x,5,[11,12])

print('after insert','\n',y)
```

o/p:

```
[[0 1]
 [2 3]
 [4 5]]


after insert
 [ 0  1  2  3  4 11 12  5]
```

Or,

```
import numpy as np

x=np.arange(6).reshape(3,2)

print(x)

print('\n')

y=np.insert(x,5,[11,12,14,15])

print('after insert','\n',y)
```

```
o/p:


[[0 1]
 [2 3]
 [4 5]]


after insert
 [ 0  1  2  3  4 11 12 14 15  5]
```

Or,

```
import numpy as np

x=np.arange(6).reshape(3,2)

print(x)

print('\n')

z=np.insert(x,2,11,axis=0)

print('after insert','\n',z)
```

o/p:

```
[[0 1]
 [2 3]
 [4 5]]


after insert
 [[ 0  1]
 [ 2  3]
 [11 11]
 [ 4  5]]
```

Or,

```
import numpy as np

x=np.arange(48).reshape(6,8)

print(x)

print('\n')

z=np.insert(x,6,11,axis=0)

print('after insert','\n',z)
```

o/p:

```
[[ 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]]


after insert
 [[ 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]
 [11 11 11 11 11 11 11 11]]
```

Or,

```
import numpy as np

x=np.arange(48).reshape(6,8)

print(x)

print('\n')

z=np.insert(x,6,11,axis=1)

print('after insert','\n',z)
```

o/p:

```
[[ 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]]


after insert
 [[ 0  1  2  3  4  5 11  6  7]
 [ 8  9 10 11 12 13 11 14 15]
 [16 17 18 19 20 21 11 22 23]
 [24 25 26 27 28 29 11 30 31]
 [32 33 34 35 36 37 11 38 39]
 [40 41 42 43 44 45 11 46 47]]
```

Or,

```
import numpy as np

a=np.arange(20).reshape(4,5)

print(a)

print('\n')
```

```
y=np.delete(a,5)

print(y)
```

o/p:

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]


[ 0  1  2  3  4  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

Or,

```
import numpy as np

a=np.arange(20).reshape(4,5)

print(a)

print('\n')

y=np.delete(a,[5,6,7,8,9,10])

print(y)
```

o/p:

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]


[ 0  1  2  3  4 11 12 13 14 15 16 17 18 19]
```

Or,

```
import numpy as np

a=np.arange(20).reshape(4,5)

print(a)

print('\n')

y=np.delete(a,2,axis=0)

print(y)
```

o/p:

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]


[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [15 16 17 18 19]]
```

Or,

```
import numpy as np

a=np.arange(20).reshape(4,5)

print(a)

print('\n')

y=np.delete(a,0,axis=0)

print(y)
```

o/p:

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]


[[ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
```

Or,

```
import numpy as np

a=np.arange(20).reshape(4,5)

print(a)

print('\n')

y=np.delete(a,[1,2],axis=0)

print(y)
```

o/p:

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]


[[ 0  1  2  3  4]
 [15 16 17 18 19]]
```

Or,

```
import numpy as np

a=np.arange(20).reshape(4,5)

print(a)
print('\n')

y=np.delete(a,[1,2],axis=1)

print(y)
```

o/p:

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]


[[ 0  3  4]
 [ 5  8  9]
 [10 13 14]
 [15 18 19]]
```

Or,

```
import numpy as np

a=np.arange(20).reshape(4,5)

print(a)

print('\n')

y=np.delete(a,4,axis=1)

print(y)
```

o/p:

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]


[[ 0  1  2  3]
 [ 5  6  7  8]
 [10 11 12 13]
 [15 16 17 18]]
```

Binary operation:

```
import numpy as np

a=15

b=17

print(bin(a),bin(b))

print('bitwise or=',np.bitwise_or(a,b))

print('bitwise and=',np.bitwise_and(a,b))
```

o/p:

```
0b1111 0b10001
bitwise or= 31
bitwise and= 1
```

or,

```
import numpy as np

a=78

b=87

print(bin(a),bin(b))

print('bitwise or=',np.bitwise_or(a,b))

print('bitwise and=',np.bitwise_and(a,b))
```

o/p:

```
0b1001110 0b1010111
bitwise or= 95
bitwise and= 70
```

or,

```
import numpy as np

print(np.invert(np.array([13],dtype=np.int8)))

print('\n')

print('binary representation of -14','\n',bin(-14))
```

```
o/p:


[-14]


binary representation of -14
 -0b1110
```

Or,

```
import numpy as np

print(np.invert(np.array([13],dtype=np.uint8)))

print('\n')

print('binary representation of 242','\n',bin(242))
```

```
o/p:


[242]


binary representation of 242
 0b11110010
```

Or,

```
import numpy as np

print(np.invert(np.array([13],dtype=np.uint8)))
print('\n')

print('binary representation of 242','\n',np.binary_repr(242))

print('binary representation of 13','\n',np.binary_repr(13,width=8))
```

o/p:

```
[242]


binary representation of 242
 11110010
binary representation of 13
 00001101
```

Or,

```
import numpy as np

print('left shift of  20','\n',np.left_shift(20,2))

print('\n')

print('right shift of 20','\n',np.right_shift(20,2))

print('\n')

print('binary representation of 20=',np.binary_repr(20,width=8))

print('binary representation of 80=',np.binary_repr(80,width=8))

print('binary representation of 5=',np.binary_repr(5,width=8))
```

o/p:

```
left shift of  20
 80


right shift of 20
 5
```

```
binary representation of 20= 00010100
binary representation of 80= 01010000
binary representation of 5= 00000101
```

or,

```
import numpy as np

print('left shift of  16','\n',np.left_shift(16,2))

print('\n')

print('right shift of 16','\n',np.right_shift(16,2))

print('\n')

print('binary representation of 16=',np.binary_repr(16,width=8))

print('binary representation of 64=',np.binary_repr(64,width=8))

print('binary representation of 4=',np.binary_repr(4,width=8))
```

o/p:

```
left shift of  16
 64


right shift of 16
 4


binary representation of 16= 00010000
binary representation of 64= 01000000
binary representation of 4= 00000100
```

or,

```
import numpy as np

print('left shift of  220','\n',np.left_shift(220,2))

print('\n')

print('right shift of 220','\n',np.right_shift(220,2))
print('\n')

print('binary representation of 220=',np.binary_repr(220,width=10))

print('binary representation of 880=',np.binary_repr(880,width=10))

print('binary representation of 55=',np.binary_repr(55,width=10))
```

o/p:

```
left shift of  220
 880


right shift of 220
 55


binary representation of 220= 0011011100
binary representation of 880= 1101110000
binary representation of 55= 0000110111
```

or,

```
import numpy as np

print('left shift of  240','\n',np.left_shift(240,2))

print('\n')

print('right shift of 240','\n',np.right_shift(240,2))

print('\n')

print('binary representation of 240=',np.binary_repr(240,width=10))

print('binary representation of 960=',np.binary_repr(960,width=10))

print('binary representation of 60=',np.binary_repr(60,width=10))
```

o/p:

```
left shift of  240
 960


right shift of 240
 60


binary representation of 240= 0011110000
binary representation of 960= 1111000000
binary representation of 60= 0000111100
```

or,

```
import numpy as np

print('left shift of  240','\n',np.left_shift(240,2))

print('\n')

print('right shift of 240','\n',np.right_shift(240,2))

print('\n')

print('binary representation of 240=',np.binary_repr(240,width=8))

print('binary representation of 960=',np.binary_repr(960,width=8))

print('binary representation of 60=',np.binary_repr(60,width=8))
```

o/p:

```
left shift of  240
 960


right shift of 240
 60


binary representation of 240= 11110000
binary representation of 960= 1111000000
binary representation of 60= 00111100
```

```
C:\Users\Shubhamay\Anaconda3\lib\site-packages\ipykernel_launcher.py:7:
DeprecationWarning: Insufficient bit width provided. This behavior will raise
an error in the future.
  import sys
```

or,

```
import numpy as np
print(np.char.add(['hello'],['shubha']))
```

o/p:

```
['helloshubha']
```

Or,

```
import numpy as np
print(np.char.add(['hello','hi'],['shubha','babu']))
```

o/p:

```
['helloshubha' 'hibabu']
```

Or,

```
import numpy as np
print(np.char.multiply('hello',3))
```

o/p:

```
hellohellohello
```

or,

```
import numpy as np
print(np.char.center('hello',20,fillchar='*'))
```

o/p:

```
*******hello********
```

Or,

```
import numpy as np
print(np.char.center('fanus',15,fillchar='#'))
```

o/p:

```
#####fanus#####
```

Or,

```
import numpy as np
print(np.char.capitalize('shubha'))
```

o/p:

```
Shubha
```

Or,

```
import numpy as np

print(np.char.capitalize('good morning'))
```

o/p:

```
Good morning
```

Or,

```
import numpy as np

print(np.char.title('good morning'))
```

o/p:

```
Good Morning
```

Or,

```
import numpy as np
print(np.char.title('how are you? '))
```

o/p:

```
How Are You?
```

Or,

```
import numpy as np
print(np.char.lower(['HELLO','GOOD','MORNING']))
```

o/p:

```
['hello' 'good' 'morning']
```

Or,

```
import numpy as np
print(np.char.upper(['hello','good','afternoon']))
```

o/p:

```
['HELLO' 'GOOD' 'AFTERNOON']
```

or

```
import numpy as np
print(np.char.split('hello how are you'))
print('\n')
print(np.char.split('hydrabad:kolkata:bengalore',sep=':'))
```

o//p:

```
['hello', 'how', 'are', 'you']

['hydrabad', 'kolkata', 'bengalore']
```

Or,

```
import numpy as np
print(np.char.split('hello how are you'))
print('\n')
print(np.char.split('hydrabad-kolkata-bengalore',sep='-'))
```

o/p:

```
['hello', 'how', 'are', 'you']

['hydrabad', 'kolkata', 'bengalore']
```

Or,

```
import numpy as np

print(np.char.splitlines('hello\nhow are you?'))

print(np.char.splitlines('good morning\rhave a nice day..'))

print(np.char.splitlines('sorry\r\nthis is afternoon!!'))
```

o/p

```
['hello', 'how are you?']
['good morning', 'have a nice day..']
['sorry', 'this is afternoon!!']
```

Or,

```
import numpy as np

print(np.char.strip('ashock arora','a'))
```

o/p:

```
shock aror
```

or,

```
import numpy as np

print(np.char.strip('arora ashock','a'))
```

o/p:

```
rora ashock
```

or,

```
import numpy as np

print(np.char.strip('arora ashocka','a'))
```

o/p:

```
rora ashock
```

or,

```
import numpy as np

print(np.char.strip('arora ashock akrossa','a'))
```

o/p:

```
rora ashock akross
```

HERE STRIP A ONLY IN EDGES

Or,

```
import numpy as np

print(np.char.strip([['java'],['admin'],['arora']],'a'))
```

o/p:

```
[['jav']
 ['dmin']
 ['ror']]
```

Or,

```
import numpy as np
print(np.char.strip(['java','admin','arora'],'a'))
```

o/p:

```
['jav' 'dmin' 'ror']
```

Or,

```
import numpy as np
x=np.char.replace('I was fan of virat kholi','was','am')
print(x)
```

o/p:

```
I am fan of virat kholi
```

Or,

```
import numpy as np
x=np.char.replace('i am from india','i am','he was')
print(x)
```

o/p:

```
he was from india
```

or,,

```
import numpy as np
x=np.char.encode('hello','cp500')
print(x)
```

o/p:

```
b'\x88\x85\x93\x93\x96'
```

or,

```
import numpy as np

x=np.char.encode('shubha','cp500')

print(x)

print(np.char.decode(x,'cp500'))
```

o/p:

```
b'\xa2\x88\xa4\x82\x88\x81'
shubha
```

or,

```
import numpy as np

x=np.char.encode('hi','cp500')

print(x)

print(np.char.decode(x,'cp500'))
```

o/p:

```
b'\x88\x89'
hi
```

trigonometric operation:

```
import numpy as np

a=np.array([0,30,45,60,90])

print('sin value:','\n',np.sin(a*np.pi/180))

print('cos value:','\n',np.cos(a*np.pi/180))

print('tan value:','\n',np.tan(a*np.pi/180))
```

o/p:

```
sin value:
 [0.         0.5        0.70710678 0.8660254  1.        ]
cos value:
 [1.00000000e+00 8.66025404e-01 7.07106781e-01 5.00000000e-01
 6.12323400e-17]
tan value:
 [0.00000000e+00 5.77350269e-01 1.00000000e+00 1.73205081e+00
 1.63312394e+16]
```

Or,

```
import numpy as np

a=np.arange(50,74).reshape(4,6)

print(a)

print('\n')

print('sin value:','\n',np.sin(a*np.pi/180))

print('\n')

print('cos value:','\n',np.cos(a*np.pi/180))

print('\n')

print('tan value:','\n',np.tan(a*np.pi/180))
```

o/p:

```
[[50 51 52 53 54 55]
 [56 57 58 59 60 61]
 [62 63 64 65 66 67]
 [68 69 70 71 72 73]]


sin value:
 [[0.76604444 0.77714596 0.78801075 0.79863551 0.80901699 0.81915204]
 [0.82903757 0.83867057 0.8480481  0.8571673  0.8660254  0.87461971]
 [0.88294759 0.89100652 0.89879405 0.90630779 0.91354546 0.92050485]
 [0.92718385 0.93358043 0.93969262 0.94551858 0.95105652 0.95630476]]


cos value:
 [[0.64278761 0.62932039 0.61566148 0.60181502 0.58778525 0.57357644]
 [0.5591929  0.54463904 0.52991926 0.51503807 0.5        0.48480962]
 [0.46947156 0.4539905  0.43837115 0.42261826 0.40673664 0.39073113]
 [0.37460659 0.35836795 0.34202014 0.32556815 0.30901699 0.2923717 ]]


tan value:
 [[1.19175359 1.23489716 1.27994163 1.32704482 1.37638192 1.42814801]
 [1.48256097 1.53986496 1.60033453 1.66427948 1.73205081 1.80404776]
 [1.88072647 1.96261051 2.05030384 2.14450692 2.24603677 2.35585237]
 [2.47508685 2.60508906 2.74747742 2.90421088 3.07768354 3.27085262]]
```

Or,

```
import numpy as np

a=np.array([27,57,54])

print(a)

print('\n')

sin=np.sin(a*np.pi/180)

print('sin value:','\n',sin)

print('\n')

inv=np.arcsin(sin)

print('inverse value:','\n',inv)
print('\n')

print('inverse in degree values','\n',np.degrees(inv))
```

o/p:

```
[27 57 54]


sin value:
 [0.4539905  0.83867057 0.80901699]


inverse value:
 [0.4712389  0.99483767 0.9424778 ]


inverse in degree values
 [27. 57. 54.]
```

Or,

```
import numpy as np
a=np.arange(50,62).reshape(3,4)
print(a)
print('\n')
sin=np.sin(a*np.pi/180)
print('sin value:','\n',sin)
print('\n')
inv=np.arcsin(sin)
print('inverse value:','\n',inv)
print('\n')
degree=np.degrees(inv)
print('inverse value in degree','\n',degree)
```

o/p:

```
[[50 51 52 53]
 [54 55 56 57]
 [58 59 60 61]]


sin value:
 [[0.76604444 0.77714596 0.78801075 0.79863551]
 [0.80901699 0.81915204 0.82903757 0.83867057]
 [0.8480481  0.8571673  0.8660254  0.87461971]]


inverse value:
 [[0.87266463 0.89011792 0.90757121 0.9250245 ]
 [0.9424778  0.95993109 0.97738438 0.99483767]
 [1.01229097 1.02974426 1.04719755 1.06465084]]


inverse value in degree
 [[50. 51. 52. 53.]
 [54. 55. 56. 57.]
 [58. 59. 60. 61.]]
```

Or,

```
import numpy as np
aa=np.array([45,65,69])
print(aa)
print('\n')
cosine=np.cos(aa*np.pi/180)
print('cosine value','\n',cosine)
print('\n')
inverse=np.arccos(cosine)
print('inverse value','\n',inverse)
print('\n')
degree=np.degrees(inverse)
print('inverse in degree values','\n',degree)
```

o/p:

```
[45 65 69]


cosine value
 [0.70710678 0.42261826 0.35836795]


inverse value
 [0.78539816 1.13446401 1.20427718]


inverse in degree values
 [45. 65. 69.]
```

Or,

```
import numpy as np
aa=np.arange(20,44).reshape(4,6)
print(aa)
print('\n')
cosine=np.cos(aa*np.pi/180)
print('cosine value','\n',cosine)
print('\n')
inverse=np.arccos(cosine)
print('inverse value','\n',inverse)
print('\n')
degree=np.degrees(inverse)
print('inverse in degree values','\n',degree)
```

o/p:

```
[[20 21 22 23 24 25]
 [26 27 28 29 30 31]
 [32 33 34 35 36 37]
 [38 39 40 41 42 43]]


cosine value
 [[0.93969262 0.93358043 0.92718385 0.92050485 0.91354546 0.90630779]
 [0.89879405 0.89100652 0.88294759 0.87461971 0.8660254  0.8571673 ]
 [0.8480481  0.83867057 0.82903757 0.81915204 0.80901699 0.79863551]
 [0.78801075 0.77714596 0.76604444 0.75470958 0.74314483 0.7313537 ]]


inverse value
 [[0.34906585 0.36651914 0.38397244 0.40142573 0.41887902 0.43633231]
 [0.45378561 0.4712389  0.48869219 0.50614548 0.52359878 0.54105207]
 [0.55850536 0.57595865 0.59341195 0.61086524 0.62831853 0.64577182]
 [0.66322512 0.68067841 0.6981317  0.71558499 0.73303829 0.75049158]]


inverse in degree values
 [[20. 21. 22. 23. 24. 25.]
 [26. 27. 28. 29. 30. 31.]
 [32. 33. 34. 35. 36. 37.]
 [38. 39. 40. 41. 42. 43.]]
```

Or,

```
import numpy as np
x=np.array([10,12,20],dtype=float)
print(x)
print('\n')
tan=np.tan(x*np.pi/180)
print('tan value','\n',tan)
print('\n')
inverse=np.arctan(tan)
print('inverse value','\n',inverse)
print('\n')
inverse_in_degree=np.degrees(inverse)
print('inverse in degrees','\n',inverse_in_degree)
```

o/p:

```
[10. 12. 20.]


tan value
```

```
 [0.17632698 0.21255656 0.36397023]


inverse value
 [0.17453293 0.20943951 0.34906585]


inverse in degrees
 [10. 12. 20.]
```

Or,

```
import numpy as np
x=np.arange(50,74).reshape(4,6)
print(x)
print('\n')
tan=np.tan(x*np.pi/180)
print('tan value','\n',tan)
print('\n')
inverse=np.arctan(tan)
print('inverse value','\n',inverse)
print('\n')
inverse_in_degree=np.degrees(inverse)
print('inverse in degrees','\n',inverse_in_degree)
```

o/p

```
[[50 51 52 53 54 55]
 [56 57 58 59 60 61]
 [62 63 64 65 66 67]
 [68 69 70 71 72 73]]


tan value
 [[1.19175359 1.23489716 1.27994163 1.32704482 1.37638192 1.42814801]
 [1.48256097 1.53986496 1.60033453 1.66427948 1.73205081 1.80404776]
 [1.88072647 1.96261051 2.05030384 2.14450692 2.24603677 2.35585237]
 [2.47508685 2.60508906 2.74747742 2.90421088 3.07768354 3.27085262]]


inverse value
 [[0.87266463 0.89011792 0.90757121 0.9250245  0.9424778  0.95993109]
 [0.97738438 0.99483767 1.01229097 1.02974426 1.04719755 1.06465084]
 [1.08210414 1.09955743 1.11701072 1.13446401 1.15191731 1.1693706 ]
 [1.18682389 1.20427718 1.22173048 1.23918377 1.25663706 1.27409035]]


inverse in degrees
 [[50. 51. 52. 53. 54. 55.]
 [56. 57. 58. 59. 60. 61.]
 [62. 63. 64. 65. 66. 67.]
 [68. 69. 70. 71. 72. 73.]]
```

Or,

```
import numpy as np

x=np.array([1.56,25.78,12.34,13.64])

print(x)

print('\n')

y=np.around(x)

print(y)

print('\n')

z=np.around(x,decimals=1)

print(z)

w=np.around(x,decimals=-1)

print('\n')

print(w)
```

o/p:

```
[ 1.56 25.78 12.34 13.64]

[ 2. 26. 12. 14.]

[ 1.6 25.8 12.3 13.6]

[ 0. 30. 10. 10.]
```

Or,

```
import numpy as np

x=np.array([[1.56,25.78,12.34,13.64],[74.75,56.56,84.44,0.23],[24.51,12.58,14
.52,18.54]])

print(x)

print('\n')

y=np.around(x)

print(y)

print('\n')

z=np.around(x,decimals=1)

print(z)

w=np.around(x,decimals=-1)

print('\n')

print(w)
```

o/p:

```
[[ 1.56 25.78 12.34 13.64]
 [74.75 56.56 84.44  0.23]
 [24.51 12.58 14.52 18.54]]




[[ 2. 26. 12. 14.]
 [75. 57. 84.  0.]
 [25. 13. 15. 19.]]


[[ 1.6 25.8 12.3 13.6]
 [74.8 56.6 84.4  0.2]
 [24.5 12.6 14.5 18.5]]


[[ 0. 30. 10. 10.]
 [70. 60. 80.  0.]
 [20. 10. 10. 20.]]
```

Or,

```
import numpy as np

x=np.array([[1.56,25.78,12.34,13.64],[74.75,56.56,84.44,0.23],[24.51,12.58,14
.52,18.54]])

print(x)

print('\n')

y=np.floor(x)

print(y)

print('\n')

z=np.floor(x,decimals=1)

print(z)

w=np.floor(x,decimals=-1)

print('\n')

print(w)
```

o/p:

```
[[ 1.56 25.78 12.34 13.64]
 [74.75 56.56 84.44  0.23]
 [24.51 12.58 14.52 18.54]]


[[ 1. 25. 12. 13.]
 [74. 56. 84.  0.]
 [24. 12. 14. 18.]]


---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-41-c75cdc4f72b0> in <module>
      6 print(y)
      7 print('\n')
----> 8 z=np.floor(x,decimals=1)
```

```
print(z)
w=np.floor(x,decimals=-1)

TypeError: 'decimals' is an invalid keyword to ufunc 'floor'
```

Or,

```
import numpy as np

x=np.array([[1.56,25.78,12.34,13.64],[74.75,56.56,84.44,0.23],[24.51,12.58,14
.52,18.54]])

print(x)

print('\n')

y=np.floor(x)#reurn the integer less than x

print(y)
```

o/p:

```
[[ 1.56 25.78 12.34 13.64]
 [74.75 56.56 84.44  0.23]
 [24.51 12.58 14.52 18.54]]


[[ 1. 25. 12. 13.]
 [74. 56. 84.  0.]
 [24. 12. 14. 18.]
```

Or,

```
import numpy as np
x=np.array([[1.56,25.78,12.34,13.64],[74.75,56.56,84.44,0.23],[24.51,12.58
,14.52,18.54]])
print(x)
print('\n')
y=np.ceil(x)#reurn the integer g greater than x
print(y)
```

o/p:

```
[[ 1.56 25.78 12.34 13.64]
 [74.75 56.56 84.44  0.23]
 [24.51 12.58 14.52 18.54]]


[[ 2. 26. 13. 14.]
 [75. 57. 85.  1.]
 [25. 13. 15. 19.]]
```

Or,

```
import numpy as np
x=np.array([-1.7,5.6,-5.78,-6.32,8.78])
y=np.floor(x)
print(y)
```

o/p:

```
[-2.  5. -6. -7.  8.]
```

Or,

```
import numpy as np
x=np.array([-1.7,5.6,-5.78,-6.32,8.78])
y=np.ceil(x)
print(y)
```

o/p:

```
[-1.  6. -5. -6.  9.]
```

Arrithmatic operation:

```
import numpy as np

a=np.arange(12).reshape(3,4)

print(a)

print('\n')

b=np.arange(60,72).reshape(4,3)

print(b)

print('\n')

y=np.dot(a,b)
```

```
print(y)
```

or,

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


[[60 61 62]
 [63 64 65]
 [66 67 68]
 [69 70 71]]


[[ 402  408  414]
 [1434 1456 1478]
 [2466 2504 2542]]
```

Or,

```
import numpy as np

a=np.arange(9).reshape(3,3)

print('first array','\n',a)

print('\n')

b=np.array([15,14,12])

print('2nd array','\n',b)

print('\n')

print('add of two array','\n',np.add(a,b))

print('\n')

print('subtract of two array','\n',np.subtract(a,b))

print('\n')

print('mutiplication of two array','\n',np.multiply(a,b))

print('\n')
print('divide of two array','\n',np.divide(a,b))
```

o/p:

```
first array
 [[0 1 2]
 [3 4 5]
 [6 7 8]]

2nd array
 [15 14 12]

add of two array
 [[15 15 14]
 [18 18 17]
 [21 21 20]]

subtract of two array
 [[-15 -13 -10]
 [-12 -10  -7]
 [ -9  -7  -4]]

mutiplication of two array
 [[ 0 14 24]
 [45 56 60]
 [90 98 96]]

divide of two array
 [[0.         0.07142857 0.16666667]
 [0.2        0.28571429 0.41666667]
 [0.4        0.5        0.66666667]]
```

Or,,

```
import numpy as np

x=np.array([0.25,0.33,0.65,2,0.23,100])

y=np.reciprocal(x)

print(y)
```

o/p

```
[4.         3.03030303 1.53846154 0.5        4.34782609 0.01      ]
```

Or,

```
import numpy as np

x=np.array([0.25,0.33,0.65,2,0,100])

y=np.reciprocal(x)

print(y)
```

o/p,

```
[4.         3.03030303 1.53846154 0.5               inf 0.01      ]
C:\Users\Shubhamay\Anaconda3\lib\site-packages\ipykernel_launcher.py:3:
RuntimeWarning: divide by zero encountered in reciprocal
  This is separate from the ipykernel package so we can avoid doing imports
until
```

Or,

```
import numpy as np
x=np.array([2])
y=np.reciprocal(x)
print(y)
```

o/p:

```
[0]
```

Or,

```
import numpy as np
x=np.array([100])
y=np.reciprocal(x)
print(y)
```

o/p:

```
[0]
```

Or,

```
import numpy as np
x=np.array([100,0.5])
y=np.reciprocal(x)
print(y)
```

o/p:

```
[0.01 2.  ]
```

Or,

```
import numpy as np
x=np.array([2,0.5])
y=np.reciprocal(x)
print(y)
```

op,

```
[0.5 2. ]
```

Or,

```
import numpy as np
a=np.array([10,100,1000])
print('our original array is:',a)
y=np.power(a,2)
print('\n')
print(y)
z=np.array([2])
print('\n')
b=np.power(a,z)
print(b)
c=np.array([1,2,3])
d=np.power(a,c)
print('\n')
print(d)
```

o/p:

```
our original array is: [  10  100 1000]


[    100   10000 1000000]


[    100   10000 1000000]


[        10      10000 1000000000]
```

In [ ]:

Or,

```
import numpy as np
a=np.arange(9).reshape(3,3)
print(a)
print('\n')
b=np.array([1,2,3])
c=np.power(a,b)
print(c)
```

o/p:

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


[[  0   1   8]
 [  3  16 125]
 [  6  49 512]]
```

Or,

```
import numpy as np
a=np.arange(9).reshape(3,3)
print(a)
print('\n')
b=np.array([[1],[2],[3]])
c=np.power(a,b)
print(c)
```

```
o/p:

[[0 1 2]
 [3 4 5]
 [6 7 8]]


[[  0   1   2]
 [  9  16  25]
 [216 343 512]]
```

Or,

```
import numpy as np
a=np.array([16,20,27])
print(a)
print('\n')
b=np.array([3,6,7])
print(b)
print('\n')
c=np.mod(a,b)
print('np.mod(a,b)','\n',c)
print('\n')
d=np.remainder(a,b)
print('np.remainder(a,b)','\n',d)
```

```
o/p:

[16 20 27]


[3 6 7]


np.mod(a,b)
 [1 2 6]


np.remainder(a,b)
 [1 2 6]
```

Or,

```
import numpy as np
a=np.array([5+2j,-2+3j,2+3j])
print('our original matrix',a)
print('\n')
```

```
y=np.real(a)
print('real part of those complex number',y)
print('\n')
x=np.imag(a)
print('imaginary part of those complex number',x)
print('\n')
z=np.conj(a)
print('conjugate of those complex number',z)
print('\n')
w=np.angle(a)
print('angle of those complex number',w)
p=np.angle(a,deg=True)
print('angle of this complex number in degree',p)
```

o/p:

```
our original matrix [ 5.+2.j -2.+3.j  2.+3.j]


real part of those complex number [ 5. -2.  2.]


imaginary part of those complex number [2. 3. 3.]


conjugate of those complex number [ 5.-2.j -2.-3.j  2.-3.j]


angle of those complex number [0.38050638 2.15879893 0.98279372]
angle of this complex number in degree [ 21.80140949 123.69006753
56.30993247]
```

or,

```
import numpy as np

a=np.array([[10,6,7],[8,9,4],[5,2,6]])

print('our original array','\n',a)

print('\n')

y=np.amin(a)
```

```
print(' minimum value of this array','\n',y)

z=np.amin(a,0)

print('minimum value of this array along 0 axis','\n',z)

w=np.amin(a,1)

print('minimum value of this array along 1 axis','\n',w)

p=np.amax(a)

print('maximum value of this array ','\n',p)

q=np.amax(a,axis=0)

print('maximum value of array along axis 0','\n',q)

r=np.amax(a,axis=1)

print('maximum value of array along axis=1','\n',r)
```

o/p:

```
our original array
 [[10  6  7]
 [ 8  9  4]
 [ 5  2  6]]


 minimum value of this array
 2
minimum value of this array along 0 axis
 [5 2 4]
minimum value of this array along 1 axis
 [6 4 2]
maximum value of this array
 10
maximum value of array along axis 0
 [10  9  7]
maximum value of array along axis=1
 [10  9  6]
```

np.ptp:

ptp() function return maximum- minimum value of an array

```
import numpy as np

x=np.array([[5,7,8],[8,7,2],[6,8,2]])

print('our original array','\n',x)

print('\n')

y=np.ptp(x) #here maximum value is 8 and minimum value is 2. so ptp is 8-2=6

print('maximum-minimum value of array','\n',y)

z=np.ptp(x,axis=0)

print('maximum-minimum value along axis 0','\n',z)

w=np.ptp(x,axis=1)

print('maximum-minimum value along axis 0','\n',w)
```

```
our original array
 [[5 7 8]
 [8 7 2]
 [6 8 2]]


maximum-minimum value of array
 6
maximum-minimum value along axis 0
 [3 1 6]
maximum-minimum value along axis 0
 [3 6 6]
```

np.percentile:

how to compute percentile?

Suppose you have 25 test scores and in order from lowest to highest they look like this:43,54,56,61,62,66,68,69,69,70,71,72,77,78,79,85,87,88,89,93,95,96,98,99, 99.To find the 90th percentile for this scores, start by multiplying 90% times the total number of the score, which gives 90%*25=0.90*25=22.5(the index).Rounding up to nearest whole number,you get 23.

Counting from left to right(from smallest to largest in the data set),you go until you find the 23 rd value in the data set. That value is 98 and it is the 90th percentile of the data set.

Now you want to find the 20th percentile. Start by taking 0.20*25=5(the index or m).this is a whole number, so here you have to find 5th and 6th element of the data set. Which are 62 and 66. Take the average of those to which is 64.so the 20th percentile of this number is 64.

Alternative method when index is whole number:

You have to (m+1)th element,

Then k th percentile=(m+1)th number-((m+1) th number – m th number)*k/100

The meiden (the 50th percentile) for the test score is the 13 th score :77

```
import numpy as np

x=np.array([[20,40,50,60,30],[10,80,90,100,70],[110,120,150,140,160],[200,220,190,210,250]])

print(x)

print('\n')

y=np.percentile(x,50)

print('the 50th percentile','\n',y)
```

o/p:

```
[[ 20  40  50  60  30]
 [ 10  80  90 100  70]
 [110 120 150 140 160]
 [200 220 190 210 250]]


the 50th percentile
 105.0
```

```
import numpy as np

x=np.array([[2,4,5,6,3],[1,8,9,10,7],[11,12,15,14,16],[20,22,19,21,25]])

print(x)

print('\n')

y=np.percentile(x,20)
```

```
print('the 20th percentile','\n',y)
```

o/p:

```
[[ 2  4  5  6  3]
 [ 1  8  9 10  7]
 [11 12 15 14 16]
 [20 22 19 21 25]]


the 20th percentile
 4.800000000000001
```

Putting those in order we get 1,2,3,4,5,6,7,8,,10,11,12,14,15,16,19,20,21,22

Here index(m)=.2*20=4

So (m+1)th number is 5.

So 20th percentile is 5-1*.2=4.8

Or,

```
import numpy as np

x=np.array([[2,34,5,6,3],[1,8,9,10,7],[11,12,15,14,16],[20,22,19,21,25]])

print(x)

print('\n')

y=np.percentile(x,90)

print('the 90th percentile','\n',y)
```

o/p:

```
[[ 2 34  5  6  3]
 [ 1  8  9 10  7]
 [11 12 15 14 16]
 [20 22 19 21 25]]
```

```
the 90th percentile
 22.300000000000004
```

Ordering this number we get
1,2,3,5,6,7,8,9,10,11,12,14,15,16,19,20,21,22,25,34

Here m=.9*20=18

So 90th percentile=(m+1) th element-3*.9=22.3

Or,

```
import numpy as np

x=np.array([[2,34,5,6,3],[1,8,9,10,7],[11,12,15,14,16],[20,22,19,21,25]])

print(x)

print('\n')

y=np.percentile(x,70)

print('the 70th percentile','\n',y)
```

o/p:

```
[[ 2 34  5  6  3]
 [ 1  8  9 10  7]
 [11 12 15 14 16]
 [20 22 19 21 25]]


the 70th percentile
 16.9
```

Or,

```
import numpy as np
p=np.array([[11,20,55],[30,47,60],[70,92,80]])
print(p)
print('\n')
q=np.percentile(p,30)
print(q)
print('\n')
```

o/p:

```
[[11 20 55]
 [30 47 60]
 [70 92 80]]
```

```
36.8
```

<u>What is percentile?</u>

Percentile of rank x=[(no Of value below x)/n]*100

in 2,2,3,4,5,5,5,6,7,8,8,8,8,8,9,9,10,11,11,12 find the percentile of 10=16/20*100=80%

<u>what value exixt at the percentile ranking of 25% in the above data set?ile/</u>

Value=(percent/100)*(n+1)

Here value=(25/100)*(20+1)=5.25

There is no 5.25 value ,so I take the average of 5th and 6th value to find what value exit in 25th percentile

<u>1st quartile=25 th percentile</u>

<u>2nd quartile=50th percentile</u>

<u>3rd quartile=75th percentile</u>

```
import numpy as np

x=np.array([[2,34,5,6,3],[1,8,9,10,7],[11,12,15,14,16],[20,22,19,21,25]])

print('our array','\n',x)

print('\n')

y=np.percentile(x,70)

print('the 70th percentile of this array','\n',y)

z=np.percentile(x,70,axis=0)

print('the 70th percentile of this array along axis 0','\n',z)

w=np.percentile(x,70,axis=1)
```

```
print('the 70th percentile of this array along axis 0','\n',w)
```

```
o/p

our array
 [[ 2 34  5  6  3]
 [ 1  8  9 10  7]
 [11 12 15 14 16]
 [20 22 19 21 25]]


the 70th percentile of this array
 16.9
the 70th percentile of this array along axis 0
 [11.9 23.2 15.4 14.7 16.9]
the 70th percentile of this array along axis 0
 [ 5.8  8.8 14.8 21.8]
```

How to compute median:

1. **If the number(n) of element in the is odd then the value of median is the value of (n+1)/2 th term.**

   Suppose you have dataset like 2,5,8,6,7,9,10,4,12,19,17

   Then rearrange the data set in assending order we get
   2,4,5,6,7,8,9,10,12,17,19
   Then the median is (11+1)/2=6th element=8
2. **If the number (n)of element in data set is even the the median =average of n/2 th term and ((n/2)+1))th term**

   Suppose you have dataset like 2,5,8,6,7,9,10,4,12,19,17,1

   Then rearrange the data set in assending order we get
   1,2,4,5,6,7,8,9,10,12,17,19
   Then the median is (6+7)/2=7.5

   Or,

   ```
   import numpy as np
   a=np.array([[13,15,18],[17,19,20],[12,14,16],[8,9,11]])
   print('our array','\n',a)
   y=np.median(a)
   print('median of this array:','\n',y)
   z=np.median(a,axis=0)
   print('median of this array along 0 axis','\n',z)
   w=np.median(a,axis=1)
   ```

```
      print('median of this array along axis 1','\n',w)
```

o/p:

```
our array
 [[13 15 18]
 [17 19 20]
 [12 14 16]
 [ 8  9 11]]
median of this array:
 14.5
median of this array along 0 axis
 [12.5 14.5 17. ]
median of this array along axis 1
 [15. 19. 14.  9.]
```

Or,

```
import numpy as np

a=np.array([[13,15,18],[17,19,20],[12,14,16]])

print('our array','\n',a)

y=np.median(a)

print('median of this array:','\n',y)

z=np.median(a,axis=0)

print('median of this array along 0 axis','\n',z)

w=np.median(a,axis=1)

print('median of this array along axis 1','\n',w)
```

o/p:

```
our array
 [[13 15 18]
 [17 19 20]
 [12 14 16]]
median of this array:
 16.0
median of this array along 0 axis
 [13. 15. 18.]
```

```
median of this array along axis 1
 [15. 19. 14.]
```

Mean=sum of the element/no.of element in data set

```
import numpy as np

x=np.array([[10,12,42],[15,16,18],[14,16,17]])

print('our array:','\n',x)

print('\n')

y=np.mean(x)

print('mean of this array','\n',y)

print('\n')

z=np.mean(x,axis=0)

print('mean of this array along axis=0','\n',z)

print('\n')

p=np.mean(x,axis=1)

print('mean of this array along axis=1','\n',p)
```

o/p:

```
our array:
 [[10 12 42]
 [15 16 18]
 [14 16 17]]


mean of this array
 17.77777777777778


mean of this array along axis=0
 [13.         14.66666667 25.66666667]


mean of this array along axis=1
```

```
 [21.33333333 16.33333333 15.66666667]
```

Weighted average:

```
import numpy as np

x=np.array([1,4,5,6])

print('our array','\n',x)

print('\n')

y=np.average(x)

print('average:','\n',y)

print('\n')

p=np.array([2,7,5,4])

q=np.average(x,weights=p)

print('weighted average','\n',q)

print('\n')

r=np.average(x,weights=p,returned=True)

print('weighted average with sum of the weight','\n',r)
```

o/p:

```
our array
 [1 4 5 6]


average:
 4.0


weighted average
 4.388888888888889


weighted average with sum of the weight
 (4.388888888888889, 18.0)
```

 Here weighted average=(1*2+4*7+5*5+6*4)/(2+7+5+4)=4.38

Or,

```
import numpy as np

x=np.arange(9).reshape(3,3)

print('our array','\n',x)

print('\n')

y=np.average(x)

print('average:','\n',y)

print('\n')

p=np.arange(9,18).reshape(3,3)

print('weights array','\n',p)

print('\n')

q=np.average(x,weights=p)

print('weighted average','\n',q)

print('\n')

r=np.average(x,weights=p,returned=True)

print('weighted average with sum of the weight','\n',r)
```

o/p:

```
our array
 [[0 1 2]
 [3 4 5]
 [6 7 8]]


average:
 4.0


weights array
 [[ 9 10 11]
 [12 13 14]
 [15 16 17]]


weighted average
 4.512820512820513


weighted average with sum of the weight
 (4.512820512820513, 117.0)
```

Or,

```
import numpy as np

x=np.arange(9).reshape(3,3)

print('our array','\n',x)

print('\n')

y=np.average(x)

print('average:','\n',y)

print('\n')

p=np.arange(3)

print('weights array','\n',p)

print('\n')

q=np.average(x,axis=0,weights=p)

print('weighted average','\n',q)

print('\n')

r=np.average(x,axis=0,weights=p,returned=True)

print('weighted average with sum of the weight','\n',r)
```

o/p:

```
our array
 [[0 1 2]
 [3 4 5]
 [6 7 8]]


average:
 4.0


weights array
 [0 1 2]


weighted average
 [5. 6. 7.]


weighted average with sum of the weight
 (array([5., 6., 7.]), array([3., 3., 3.]))
```

<u>How to compute standerd deviation:</u>

Suppose array is [1,2,3,4]
Mean of this array=2.5
Squre deviation=(|1-2.5|**2+|2-2.5|**2+|3-2.5|**2+|4-2.5|**2)/4=5/4=1.25
Standerd devion=sqrt(1.25)=1.1180

The formula of standerd deviation is
std = sqrt(mean(abs(x - x.mean())**2))

```
import numpy as np
y=np.array([1,2,3,4])
x=np.std(y)
print(x)
```

o/p:

```
1.118033988749895
```

Or,

```
import numpy as np
y=np.arange(12).reshape(3,4)
x=np.std(y)
print(x)
```

o/p:

```
3.452052529534663
```

Variance:

The formula of variance is
Variance= **mean(abs(x - x.mean())**2).**

Standerd deviation=sqrt(variance)

Or,

```
import numpy as np
x=np.array([1,2,3,4])
y=np.var(x)
print(y)
```

o/p:

```
1.25
```

Or,

```
import numpy as np
x=np.arange(9).reshape(3,3)
y=np.var(x)
print(y)
```

o/p:

```
6.666667
```

Variance and standerd function:

```
def mean(x):
    return sum(x)/len(x)
def variance(x):
    l=[]
    for element in x:
        l.append((element-mean(x))**2)
    return mean(l)
def standerd_deviation(x):
    return (variance(x))**0.5
```

if I call this function standerd_deviation([1,2,3,4]) and variance([1,2,3,4]), we get 1.1180 and 1.25 as output respectively

np.sort()

```
import numpy as np
rand=np.random
x=rand.randint(0,10,(4,6))
print('our original array','\n',x)
print('\n')
y=np.sort(x)
```

```
print('after sorting the array','\n',y)
z=np.sort(x,axis=0)
print('after sorting the array along 0 axis','\n',z)
p=np.sort(x,axis=1)
print('after sorting the array along 1 axis','\n',p)
```

o/p:

```
our original array
 [[1 0 0 6 4 5]
 [8 4 1 8 6 9]
 [8 6 5 5 6 0]
 [8 7 5 8 4 5]]


after sorting the array
 [[0 0 1 4 5 6]
 [1 4 6 8 8 9]
 [0 5 5 6 6 8]
 [4 5 5 7 8 8]]
after sorting the array along 0 axis
 [[1 0 0 5 4 0]
 [8 4 1 6 4 5]
 [8 6 5 8 6 5]
 [8 7 5 8 6 9]]
after sorting the array along 1 axis
 [[0 0 1 4 5 6]
 [1 4 6 8 8 9]
 [0 5 5 6 6 8]
 [4 5 5 7 8 8]]
```

Or,

```
import numpy as np
rand=np.random
x=rand.randint(0,10,10)
print('our original array','\n',x)
print('\n')
y=np.sort(x)
print('after sorting the array','\n',y)
```

o/p:

```
our original array
 [2 6 5 9 8 8 8 5 9 8]


after sorting the array
 [2 5 5 6 8 8 8 8 9 9]
```

```
      Or,

      import numpy as np
      dt=np.dtype([('name','S10'),('age','i4')])
      a=np.array([('ram',27),('arjun',25),('anil',26),('hari',28)],dtype=dt)
      print('our original array:','\n',a)
      print('\n')
      b=np.sort(a,order='name')
      print('array after sort by name:','\n',b)

      o/p:


our original array:
 [(b'ram', 27) (b'arjun', 25) (b'anil', 26) (b'hari', 28)]


array after sort by name:
 [(b'anil', 26) (b'arjun', 25) (b'hari', 28) (b'ram', 27)]


Or,

import numpy as np
dt=np.dtype([('name','S10'),('age','i4')])
a=np.array([('ram',27),('arjun',25),('anil',26),('hari',28)],dtype=dt)
print('our original array:','\n',a)
print('\n')
b=np.sort(a,order='age')
print('array after sort by age:','\n',b)

o/p:

our original array:
 [(b'ram', 27) (b'arjun', 25) (b'anil', 26) (b'hari', 28)]


array after sort by age:
 [(b'arjun', 25) (b'anil', 26) (b'ram', 27) (b'hari', 28)]


Or,
      import numpy as np
      rand=np.random
      x=rand.randint(10,20,10)
      print('our original array:','\n',x)
      print('\n')
      y=np.sort(x)
      print('sort by value:','\n',y)
      print('\n')
      z=np.argsort(x)
      print('sort by the index:','\n',z)
      print('\n')
      print('reconstructed array x in sorted order:','\n',x[z])
      l=[]
      for element in z:
          value=x[element]
          l.append(value)
      print('reconstructed the array using loop:','\n',l)
```

```
      or,
```

```
our original array:
 [14 18 13 13 16 17 13 18 12 13]


sort by value:
 [12 13 13 13 13 14 16 17 18 18]


sort by the index:
 [8 2 3 6 9 0 4 5 1 7]


reconstructed array x in sorted order:
 [12 13 13 13 13 14 16 17 18 18]


reconstructed the array using loop:
 [12, 13, 13, 13, 13, 14, 16, 17, 18, 18]
```

Or,

```
import numpy as np
rand=np.random
x=rand.randint(10,20,(3,3))
print(x)
print('\n')
y=rand.randint(50,100,(3,3))
print(y)
print('\n')
z=np.argsort((y,x))
print(z)
print('\n')
```

o/p:

```
[[16 19 19]
 [15 14 11]
 [16 14 11]]
```

```
[[77 94 75]
 [62 74 52]
 [92 71 92]]


[[[2 0 1]
  [2 0 1]
  [1 0 2]]

 [[0 1 2]
  [2 1 0]
  [2 1 0]]]
```

Or,

```
#top five total in a scholl of madhyamik examination
import numpy as np
num=np.array([633,657,687,654,689,632,624,645])
student_name=np.array(['arjun','dhruba','paschima','ramanuj','avinab','
shubha','sujit','babu'])
y=np.argsort(num)
z=student_name[y]
top_5=z[::-1][:6]
print('name of top five scorer:','\n',top_5)
```

or,

```
name of top five scorer:
 ['avinab' 'paschima' 'dhruba' 'ramanuj' 'babu' 'arjun']
```

np.lexsort:

```
import numpy as np
x=np.array([2,3,6,3,4,8,5,4,8,2])
y=np.array([5,8,6,4,7,8,5,7,6,5])
print(np.sort(x))
print(np.argsort(x))
print('\n')
print(y)
print('\n')
print(np.lexsort((y,x)))
```

o/p:

```
[2 2 3 3 4 4 5 6 8 8]
```

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


[5 8 6 4 7 8 5 7 6 5]


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

here for 2 and 2 index are 0 and 9 respectively .the decider is 5 and 8 respectively, which are in correct order. So index remain 0 and 9

for 3 and 3 index are 1 and 3.the decider are 6 and 4 ,which are not in correct order. those will be 4 and 6. So index be 3 and 1.

For 4 and 4 ,index are 4 and 7. The decider are 7 and 8,which are in order. So index remain 4 and 7.

For last 8 and 8 ,the index are 5 and 8, decider are 8 and 5,which are not in correct order,those will be 5 and 8,so the index will be 8 and 5.

So the resultant index be 0 9 3 1 4 7 6 2 8 5

Or

```
import numpy as np
x=np.array([1,5,4,5,6,2,4])
y=np.array([2,3,8,4,5,7,6])
print(np.sort(x))
print(np.argsort(x))
print('\n')
print(y)
print('\n')
print(np.lexsort((y,x)))
```

o/p:

```
[1 2 4 4 5 5 6]
[0 5 2 6 1 3 4]


[2 3 8 4 5 7 6]


[0 5 6 2 1 3 4]
```

Here index of two for are 2&6 respectiely. Decider are in rank 8 & 4 ,which are not in order,correct rank will be 4 & 8. So the index will be 6 & 2.

The index of two five are 1 & 3. Decidor are in rank 5&7 .which are in correct order.so index will be 1&3.

Resultant index will be
0 5 6 2 1 3 4

Or,

```
import numpy as np
nm=np.array(['virat','sachin','virat','sachin'])
print(np.sort(nm))
print(np.argsort(nm))
print('\n')
dy=np.array(['bowler','filder','batsman','bowler'])
print(np.argsort(dy))
print('\n')
print(np.lexsort((dy,nm)))
```

o/p:

```
['sachin' 'sachin' 'virat' 'virat']
[1 3 0 2]


[2 0 3 1]


[3 1 2 0]
```

Here the index of two sachin are 1,3 ,decider are in rank 2,0 respective .those are not in correct order. The correct order be 0,2. So the index be 3,1.
The index of two virat are 0,2.the decider are in rank 3,1. Whivh are not in correct order.correct order be 1,3.so the index be 2,0.

The resultant index are
3 1 2 0

Or,

```
import numpy as np
nm=np.array(['virat','sachin','virat','sachin'])
dy=np.array(['bowler','filder','batsman','bowler'])
a=np.lexsort((dy,nm))
print(a)
print([nm[i]+','+dy[i] for i in a])
```

o/p:

```
[3 1 2 0]
['sachin,bowler', 'sachin,filder', 'virat,batsman', 'virat,bowler']
```

Or,

```
import numpy as np
rand=np.random
x=rand.randint(10,20,(3,3))
```

```
print(x)
print('\n')
y=rand.randint(50,100,(3,3))
print(y)
print('\n')
z=np.lexsort((y,x))
print(z)
print('\n')
print([str(x[i])+','+str (y[i]) for i in z])
```

o/p:

```
[[11 19 13]
 [11 13 12]
 [12 19 18]]


[[71 56 52]
 [83 97 86]
 [97 66 92]]


[[0 2 1]
 [0 2 1]
 [0 2 1]]


['[[11 19 13]\n [12 19 18]\n [11 13 12]],[[71 56 52]\n [97 66 92]\n [83 97
86]]', '[[11 19 13]\n [12 19 18]\n [11 13 12]],[[71 56 52]\n [97 66 92]\n
[83 97 86]]', '[[11 19 13]\n [12 19 18]\n [11 13 12]],[[71 56 52]\n [97 66
92]\n [83 97 86]]']
```

Or,

```
import numpy as np
x=np.array([8,5,6,8,9,10,6,3])
y=np.argsort(x)
print('sort by index:','\n',y)
z=np.argmax(x)
print(' index of number with maximum value:','\n',z)
w=np.argmin(x)
print('index of number with minimum value:','\n',w)
```

o/p:

```
sort by index:
 [7 1 2 6 0 3 4 5]
 index of number with maximum value:
 5
index of number with minimum value:
 7
```

Or,

```
import numpy as np
rand=np.random
x=rand.randint(10,19,(3,3))
print('our original array:','\n',x)
print('\n')
z=np.argmax(x,axis=0)
print('index of maximum value accross:','\n',z)
w=np.argmax(x,axis=1)
print('index of maximum value accross:','\n',w)
```

o/p:

```
our original array:
 [[15 10 11]
 [14 17 10]
 [12 15 16]]


index of maximum value accross:
 [0 1 2]
index of maximum value accross:
 [0 1 2]
```

Or,

```
import numpy as np
rand=np.random
x=rand.randint(10,19,(3,3))
print('our original array:','\n',x)
print('\n')
z=np.argmax(x,axis=0)
print('index of maximum value accross axis 0:','\n',z)
w=np.argmax(x,axis=1)
print('index of maximum value accross axis 1:','\n',w)
p=np.argmin(x,axis=0)
print('index of minimum value accross axis 0:','\n',p)
q=np.argmin(x,axis=1)
print('index of minimum value accross axis 1:','\n',q)
```

o/p:

```
our original array:
 [[18 16 17]
 [18 13 16]
 [13 13 17]]
```

```
index of maximum value accross axis 0:
 [0 0 0]
index of maximum value accross axis 1:
 [0 0 2]
index of minimum value accross axis 0:
 [2 1 1]
index of minimum value accross axis 1:
 [1 1 0]
```

np.nonzero

```
import numpy as np
rand=np.random
x=rand.randint(0,5,10)
print(x)
print('\n')
print('index of non-zero element:','\n',np.nonzero(x))
```

o/p:

```
our original array:
 [3 2 1 1 2 0 0 3 3 0]


index of non-zero element:
 (array([0, 1, 2, 3, 4, 7, 8], dtype=int64),)
```

Or,

```
import numpy as np
x=np.array([[30,40,0],[20,0,30],[0,50,60]])
print('our original array','\n',x)
print('\n')
print(np.nonzero(x))
```

o/p:

```
our original array
 [[30 40  0]
 [20  0 30]
 [ 0 50 60]]


(array([0, 0, 1, 1, 2, 2], dtype=int64), array([0, 1, 0, 2, 1, 2],
dtype=int64))
```

Or,

```
import numpy as np
x=np.array([[7,8,5],[4,3,5],[6,2,5],[8,5,4],[5,6,3]])
print('our original array:','\n',x)
print('\n')
y=np.where(x>5)
print('index of element greater than 5:','\n',y)#(index of non-zero
element accross axis 0),(.... axis 1)
print('\n')
print('element greater than 5:','\n',x[y])
```

o/p:

```
our original array:
 [[7 8 5]
 [4 3 5]
 [6 2 5]
 [8 5 4]
 [5 6 3]]


index of element greater than 5:
 (array([0, 0, 2, 3, 4], dtype=int64), array([0, 1, 0, 0, 1],
dtype=int64))


element greater than 5:
 [7 8 6 8 6]
```

Or,

```
import numpy as np
x=np.array([[-1,2,-3,-4],[8,-1,5,-5],[4,5,-6,5],[5,8,-5,-6]])
print('our original array','\n',x)
y=np.where(x>0,x,100)# if the condition is not true than replace
print('replace the element which is not greater than 0 with
100:','\n',y)
```

o/p:

```
our original array
 [[-1  2 -3 -4]
 [ 8 -1  5 -5]
 [ 4  5 -6  5]
 [ 5  8 -5 -6]]
replace the element less than 0 with 100:
 [[100   2 100 100]
 [  8 100   5 100]
 [  4   5 100   5]
 [  5   8 100 100]]
```

Or,

```
import numpy as np
x=np.array([[-1,7,-3,-4],[8,-1,5,-5],[4,9,-6,5],[5,8,-5,-6]])
print('our original array','\n',x)
y=np.where(x>5,x,100)# if the condition is not true than replace
print('replace the element which is not greater than 0 with
100:','\n',y)
```

o/p:

```
our original array
 [[-1  7 -3 -4]
 [ 8 -1  5 -5]
 [ 4  9 -6  5]
 [ 5  8 -5 -6]]
replace the element which is not greater than 0 with 100:
 [[100   7 100 100]
 [  8 100 100 100]
 [100   9 100 100]
 [100   8 100 100]]
```

<u>np.extract:</u>

```
import numpy as np
x=np.array([[5,8,6],[8,7,6],[9,5,7]])
print(x)
y=np.where(np.mod(x,2)==0)
print(y)
```

or,

```
[[5 8 6]
 [8 7 6]
 [9 5 7]]
(array([0, 0, 1, 1], dtype=int64), array([1, 2, 0, 2], dtype=int64))
```

Or,

```
import numpy as np
x=np.array([[5,8,6],[8,7,6],[9,5,7]])
print('our original array:','\n',x)
print('\n')
condition=np.mod(x,2)==0
print('codition:','\n',condition)
print('\n')
```

```
y=np.extract(condition,x)
print('element which give reminder as 0 when it is divide by 2:','\n',y)

o/p,

our original array:
 [[5 8 6]
 [8 7 6]
 [9 5 7]]


codition:
 [[False  True  True]
 [ True False  True]
 [False False False]]


element which give reminder as 0 when it is divide by 2:
 [8 6 8 6]

Or,

import numpy as np
x=np.array([[5,8,6],[8,7,6],[9,5,7]])
print('our original array:','\n',x)
print('\n')
condition=np.mod(x,3)==0
print('codition:','\n',condition)
print('\n')
y=np.extract(condition,x)
print('element which give reminder as 0 when it is divide by 3:','\n',y)

o/p:

our original array:
 [[5 8 6]
 [8 7 6]
 [9 5 7]]


codition:
 [[False False  True]
 [False False  True]
 [ True False False]]


element which give reminder as 0 when it is divide by 3:
 [6 6 9]



NB

import numpy as np
```

```
x=np.array([8,256,8755],dtype=np.int8)
print(x)
```

o/p:

```
[ 8  0 51]
```

Or,

```
import numpy as np
x=np.array([8,256,8755],dtype=np.int32)
print(x)
```

o/p:

```
[   8  256 8755]
```

NB: <u>celsious to fheranhite convertion using map</u>

```
temps=[('barlin',24),('argentina',25),('india',45)]
c_to_f=lambda data:(data[0],(9/5)*data[1]+32)
print(list(map(c_to_f,temps)))
```

o/p:

```
[('barlin', 75.2), ('argentina', 77.0), ('india', 113.0)]
```

<u>Lambda function:</u>

<u>Squre of a given number:</u>

```
f=lambda x:x**2
f(2)
```

o/p:

```
4
```

<u>np.byteswap:</u>

```
import numpy as np
x=np.arange(3,dtype=np.int16)
print(x)
print(list(map(hex,x)))# maf(func,data)
print(x.byteswap(True))
print(list(map(hex,x)))
```

o/p:

```
[0 1 2]
```

```
['0x0', '0x1', '0x2']
[  0 256 512]
['0x0', '0x100', '0x200']
```

Or,

```
import numpy as np
rand=np.random
x=rand.randint(0,10,10)
print(x)
print('\n')
print(list(map(hex,x)))
print('\n')
y=x.byteswap(True)
print(y)
print('\n')
print(list(map(hex,x)))
```

o/p:

```
[1 8 6 5 0 4 6 2 7 6]


['0x1', '0x8', '0x6', '0x5', '0x0', '0x4', '0x6', '0x2', '0x7', '0x6']


[ 16777216 134217728 100663296  83886080         0  67108864 100663296
  33554432 117440512 100663296]


['0x1000000', '0x8000000', '0x6000000', '0x5000000', '0x0', '0x4000000',
'0x6000000', '0x2000000', '0x7000000', '0x6000000']
```

Or

```
import numpy as np
rand=np.random
x=rand.randint(0,10,(3,3))
print(x)
print('\n')
print(list(map(hex,x)))
print('\n')
y=x.byteswap(True)
print(y)
print('\n')
print(list(map(hex,x)))
```

o/p:

```
[[3 5 0]
 [3 0 5]
 [2 1 0]]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-66-bcdf71e9c30a> in <module>
      4 print(x)
      5 print('\n')
----> 6 print(list(map(hex,x)))
      7 print('\n')
      8 y=x.byteswap(True)

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

Id()

```
import numpy as np
a=np.arange(9)
print('our original array:','\n',a)
print('\n')
print('id of a:','\n',id(a))
print('\n')
b=a
print('id of b:','\n',id(b))
print('\n')
b.shape=(3,3)
print('reshape of b','\n',b)
print('\n')
print('reshape of a:','\n',a)
```

o/p:

```
our original array:
 [0 1 2 3 4 5 6 7 8]


id of a:
 762620575952


id of b:
 762620575952


reshape of b
 [[0 1 2]
 [3 4 5]
 [6 7 8]]


reshape of a:
 [[0 1 2]
 [3 4 5]
 [6 7 8]]
```

Or,

```
import numpy as np
x=np.arange(12).reshape(2,6)
print('our original array:','\n',x)
print('\n')
print('id of x:','\n',id(x))
print('\n')
y=x
print('original y:','\n',y)
print('\n')
print('can we write y is x','\n',y is x)
y.shape=(4,3)
print('y after reshape:','\n',y)
print('\n')
print('x after reshape:','\n',x)
```

o/p:

```
our original array:
 [[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]


id of x:
 580635892672


original y:
 [[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]


can we write y is x
 True
y after reshape:
 [[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

```
x after reshape:
 [[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

view()

```
import numpy as np
x=np.arange(12).reshape(6,2)
print('our original array:','\n',x)
print('\n')
y=x.view()
print('id of x','\n',id(x))
print('\n')
print('id of y:','\n',id(y))# id of x and y are different
print('\n')
print('view of x: before reshape','\n',y)
print('\n')
print('can we write y is x','\n',y is x)
y.shape=(3,4)
print('y after reshape:','\n',y)
print('\n')
print('x after reshape of y:','\n',x)#the shape of x is not affected by
the change of shape of y
```

o/p:

```
our original array:
 [[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]]


id of x
 580635955248


id of y:
 580635955648


view of x: before reshape
 [[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]]


can we write y is x
 False
y after reshape:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]


x after reshape of y:
 [[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]]
```

Slice:

```
import numpy as np
x=np.arange(9).reshape(3,3)
print('our original array:','\n',x)
print('\n')
s=x[:,:2]
print('after slice:','\n',s)
```

o/p:

```
our original array:
 [[0 1 2]
 [3 4 5]
 [6 7 8]]


after slice:
 [[0 1]
 [3 4]
 [6 7]]
```

copy()

```
import numpy as np
a=np.arange(9).reshape(3,3)
print('our original array','\n',a)
b=a.copy()
print('copy of a','\n',a)
print('\n')
print('if a is equal to b:,','\n',b is a)
b[0,0]=100
```

```
print('b after modification:','\n',b)
print('a after modification:','\n',a)
```

o/p:

```
our original array
 [[0 1 2]
 [3 4 5]
 [6 7 8]]
copy of a
 [[0 1 2]
 [3 4 5]
 [6 7 8]]


if a is equal to b:,
 False
b after modification:
 [[100   1   2]
 [  3   4   5]
 [  6   7   8]]
a after modification:
 [[0 1 2]
 [3 4 5]
 [6 7 8]]
```

Or,

```
import numpy as np
a=np.arange(9).reshape(3,3)
print('our original array','\n',a)
b=a.view()
print('copy of a','\n',a)
print('\n')
print('if a is equal to b:,','\n',b is a)
b[0,0]=100
print('b after modification:','\n',b)
print('a after modification:','\n',a)
```

o/p:

```
our original array
 [[0 1 2]
 [3 4 5]
 [6 7 8]]
copy of a
 [[0 1 2]
 [3 4 5]
 [6 7 8]]


if a is equal to b:,
 False
b after modification:
 [[100   1   2]
 [  3   4   5]
 [  6   7   8]]
a after modification:
 [[100   1   2]
 [  3   4   5]
 [  6   7   8]]
```

Or,

```
import numpy as np
x=np.arange(12).reshape(6,2)
print('our original array:','\n',x)
print('\n')
y=x.copy()
print('id of x','\n',id(x))
print('\n')
print('id of y:','\n',id(y))# id of x and y are different
print('\n')
print('view of x: before reshape','\n',y)
print('\n')
print('can we write y is x','\n',y is x)
y.shape=(3,4)
print('y after reshape:','\n',y)
print('\n')
print('x after reshape of y:','\n',x)#the shape of x is not affected by
the change of shape of y
```

o/p:

```
our original array:
 [[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]]


id of x
 580635970336


id of y:
 580635970976


view of x: before reshape
 [[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]]


can we write y is x
 False
y after reshape:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]


x after reshape of y:
 [[ 0  1]
 [ 2  3]
 [ 4  5]
 [ 6  7]
 [ 8  9]
 [10 11]]
```

<u>matlib()</u>

```
import numpy.matlib
import numpy as np
x=np.matlib.empty((4,3))
print(x)
```

o/p:

```
[[3.22117906e-312 3.16202013e-322 0.00000000e+000]
 [0.00000000e+000 1.29060870e-306 5.50079071e+170]
 [1.95138605e+160 4.42911191e-062 5.88650721e-062]
 [5.49266350e+174 2.81973031e+179 9.14210547e-043]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.matlib.zeros((4,3),dtype=int)
print(x)
```

o/p:

```
[[0 0 0]
 [0 0 0]
 [0 0 0]
 [0 0 0]]
```

<u>numpy.matlib.ones()</u>

```
import numpy.matlib
import numpy as np
x=np.matlib.ones((3,3),dtype=int)
```

```
print(x)
```

o/p:

```
[[1 1 1]
 [1 1 1]
 [1 1 1]]
```

<u>numpy.matlib.eye()</u>

```
import numpy.matlib
import numpy as np
x=np.matlib.eye(n=3,M=4,k=1,dtype=int)
print(x)
```

o/p:

```
[[0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.matlib.eye(n=10,M=11,k=3,dtype=int)
print(x)
```

o/p:

```
[[0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.matlib.eye(n=10,M=11,k=4,dtype=int)
print(x)
```

o/p:

```
[[0 0 0 0 1 0 0 0 0 0 0]
```

```
 [0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 1]
 [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 0 0 0 0 0 0]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.matlib.eye(n=10,M=11,k=2,dtype=int)
print(x)
```

o/p:

```
[[0 0 1 0 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 0 0]]
```

numpy.matlib.identity()

```
import numpy.matlib
import numpy as np
x=np.matlib.identity(n=5,dtype=int)
print(x)
```

o/p:

```
[[1 0 0 0 0]
 [0 1 0 0 0]
 [0 0 1 0 0]
 [0 0 0 1 0]
 [0 0 0 0 1]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.matlib.identity(n=8,dtype=int)
print(x)
```

o/p:

```
[[1 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 0 0 1 0 0 0 0]
 [0 0 0 0 1 0 0 0]
 [0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 1]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.matlib.rand((3,3))
print(x)
```

o/p:

```
[[0.19531676 0.93655034 0.47804248]
 [0.06056418 0.90539927 0.71727369]
 [0.57216098 0.604494   0.29231445]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.matlib.rand((4,3))
print(x)
```

o/p:

```
[[0.55489049 0.99510696 0.3092385 ]
 [0.92656746 0.33370762 0.54427414]
 [0.19670969 0.33092572 0.22346803]
 [0.60092629 0.44933379 0.99311094]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.matrix('1,2;3,4')
print(x)
```

o/p:

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

Or,

```
import numpy.matlib
import numpy as np
x=np.matrix('1,2,2;3,4,5;3,6,8')
print(x)
```

o/p:

```
[[1 2 2]
 [3 4 5]
 [3 6 8]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.matrix('1,2,2;3,4,5;3,6,8')
print(x)
print('\n')
j=np.asarray(x)
print(j)
```

o/p:

```
[[1 2 2]
 [3 4 5]
 [3 6 8]]


[[1 2 2]
 [3 4 5]
 [3 6 8]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.matrix('1,2,2;3,4,5;3,6,8')
print(x)
print('\n')
j=np.asarray(x)
print(j)
print('\n')
k=np.asmatrix(j)
print(k)
```

o/p:

```
[[1 2 2]
 [3 4 5]
 [3 6 8]]


[[1 2 2]
 [3 4 5]
 [3 6 8]]


[[1 2 2]
 [3 4 5]
 [3 6 8]]
```

Or,

```
import numpy.matlib
import numpy as np
x=np.asmatrix('1,2,3,4;4,5,6,7;8,9,7,5')
print(x)
```

o/p:

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

Or,

```
import numpy.matlib
import numpy as np
x=np.asarray('1,2,3,4;4,5,6,7;8,9,7,5')
print(x)
```

```
      o/p:


1,2,3,4;4,5,6,7;8,9,7,5
```

<u>Dot multiplication:</u>

```
import numpy as np
x=np.array([[1,2,3],[2,3,4],[7,8,9]])
print('our first array:','\n',x)
print('\n')
y=np.array([[4,5,6],[1,2,3],[4,5,6]])
print( 'our 2nd array:','\n',y)
print('\n')
z=np.dot(x,y)
print('array after dot multiplication:','\n',z)
```

```
o/p:

our first array:
 [[1 2 3]
 [2 3 4]
 [7 8 9]]


our 2nd array:
 [[4 5 6]
 [1 2 3]
 [4 5 6]]


array after dot multiplication:
 [[ 18  24  30]
 [ 27  36  45]
 [ 72  96 120]]
```

Or,

```
import numpy as np
x=np.matrix('1,2,3;2,3,4;7,8,9')
print('our first array:','\n',x)
print('\n')
y=np.matrix('4,5,6;1,2,3;4,5,6')
print( 'our 2nd array:','\n',y)
print('\n')
z=np.dot(x,y)
print('array after dot multiplication:','\n',z)
```

```
o/p:

our first array:
 [[1 2 3]
 [2 3 4]
 [7 8 9]]


our 2nd array:
 [[4 5 6]
 [1 2 3]
 [4 5 6]]


array after dot multiplication:
 [[ 18  24  30]
 [ 27  36  45]
 [ 72  96 120]]
```

Or,

```
import numpy as np
x=np.array([[1,2,3],[4,7,8],[8,9,7]])
print(x)
print('\n')
y=np.array([[4,5,6],[4,5,9],[7,8,9]])
print(y)
print('\n')
z=np.vdot(x,y)
print(z)
```

o/p:

```
[[1 2 3]
 [4 7 8]
 [8 9 7]]


[[4 5 6]
 [4 5 9]
 [7 8 9]]


346
```

Or,

```
1*4+2*5+3*6+4*4+7*5+8*9+8*7+9*8+7*9 # Vector dot product
```

o/p:

```
346
```

Or,

```
      import numpy as np
      x=np.array([[4,5,8],[8,9,7],[4,5,8]])
      print(x)
      print('\n')
      y=np.array([[7,8,9],[2,5,6],[5,6,8]])
      print(y)
      print('\n')
      z=np.inner(x,y)
      print(z)#[[4*7+5*8+8*9],...]

      o/p:
```

```
[[4 5 8]
 [8 9 7]
 [4 5 8]]


[[7 8 9]
 [2 5 6]
 [5 6 8]]


[[140  81 114]
 [191 103 150]
 [140  81 114]]
```

Or,

```
import numpy as np
print(np.inner(np.array([1,2,3]),np.array([3,4,5])))
```

o/p:

```
26
```

<u>np.matmul:</u>

```
import numpy as np
x=[[2,3],[1,2]]
print(x)
print('\n')
y=[1,2]
print(y)
print('\n')
z=np.matmul(x,y)
print(z)
```

o/p:

```
[[2, 3], [1, 2]]
```

```
[1, 2]


[8 5]
```

Or,

```
import numpy as np
x=[[2,3],[1,2]]
print(x)
print('\n')
y=[1,2]
print(y)
print('\n')
z=np.matmul(y,x)
print(z)
```

o/p:

```
[[2, 3], [1, 2]]


[1, 2]


[4 7]
```

Or,

```
import numpy as np
x=[[2,3,4],[1,2,3],[1,2,3]]
print(x)
print('\n')
y=[[1,2,4],[4,5,6]]
print(y)
print('\n')
z=np.matmul(y,x)
print(z)
```

o/p:

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


[[1, 2, 4], [4, 5, 6]]


[[ 8 15 22]
 [19 34 49]]
```

Or,

```
import numpy as np
x=[[2,3,4],[1,2,3],[1,2,3]]
print(x)
print('\n')
y=[[1,2,4],[4,5,6]]
print(y)
print('\n')
z=np.matmul(x,y)
print(z)
```

o/p:

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


[[1, 2, 4], [4, 5, 6]]



---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-20-9f29e3afd0a1> in <module>
      6 print(y)
      7 print('\n')
----> 8 z=np.matmul(x,y)
      9 print(z)

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0,
with gufunc signature (n?,k),(k,m?)->(n?,m?) (size 2 is different from 3)
```

Or,

```
import numpy.matlib
import numpy as np
a=np.arange(12).reshape(4,3)
print(a)
print('\n')
b=np.arange(6).reshape(3,2)
print(b)
print('\n')
c=np.matmul(a,b)
print(c)
```

o/p:

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


[[0 1]
 [2 3]
 [4 5]]


[[10 13]
 [28 40]
 [46 67]
 [64 94]]
```

Or,

```
import numpy as np
a=np.arange(12).reshape(4,3)
print(a)
print('\n')
b=np.arange(6).reshape(3,2)
print(b)
print('\n')
c=np.matmul(a,b)
print(c)
```

o/p:

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


[[0 1]
 [2 3]
 [4 5]]


[[10 13]
 [28 40]
 [46 67]
 [64 94]]
```

Or,

```
import numpy as np
a=np.arange(27).reshape(3,3,3)
```

```
print('our first matrix:','\n',a)
print('\n')
b=np.arange(6).reshape(3,2)
print('our 2nd matrix:','\n',b)
print('\n')
c=np.matmul(a,b)
print('matrix after mutiplication','\n',c)
```

o/p:

```
our first matrix:
 [[[ 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]]]


our 2nd matrix:
 [[0 1]
 [2 3]
 [4 5]]


matrix after mutiplication
 [[[ 10  13]
  [ 28  40]
  [ 46  67]]

 [[ 64  94]
  [ 82 121]
  [100 148]]

 [[118 175]
  [136 202]
  [154 229]]]
```

Or,

```
import numpy as np
a=np.arange(27).reshape(3,3,3)
print('our first matrix:','\n',a)
print('\n')
b=np.arange(6).reshape(3,2)
```

```
print('our 2nd matrix:','\n',b)
print('\n')
c=np.dot(a,b)
print('matrix after mutiplication','\n',c)
```

o/p:

```
our first matrix:
 [[[ 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]]]


our 2nd matrix:
 [[0 1]
 [2 3]
 [4 5]]


matrix after mutiplication
 [[[ 10  13]
  [ 28  40]
  [ 46  67]]

 [[ 64  94]
  [ 82 121]
  [100 148]]

 [[118 175]
  [136 202]
  [154 229]]]
```

np.linalg.det()

```
import numpy as np
a=np.array([[1,5],[2,5]])
print(a)
print('\n')
y=np.linalg.det(a)
print('determinate of this matrix','\n',y)
```

```
o/p:

[[1 5]
 [2 5]]


determinate of this matrix
 -5.000000000000001



Or,

import numpy as np
a=np.array([[1,2,3],[4,5,7],[8,9,7]])
print(a)
print('\n')
y=np.linalg.det(a)
print(y)

o/p:

[[1 2 3]
 [4 5 7]
 [8 9 7]]


15.999999999999991



Or,

det_of_above_matrix=1*(35-63)-2*(28-56)+3*(36-40)
print(det_of_above_matrix)

o/p:

16
```

<u>np.linalg.solve()</u>

```
import numpy as np
x=np.array([[1,1],[1,-1]])#x+y=8,x-y=4
print(x)
print('\n')
y=np.array([[8],[4]])
print(y)
print('\n')
z=np.linalg.solve(x,y)
print('soluation of this eqution:','\n',z)#here x=6,y=2
```

```
o/p:

[[ 1  1]
 [ 1 -1]]


[[8]
 [4]]


soluation of this eqution:
 [[6.]
 [2.]]
```

Or,

<u>Soluation of this eqution achive by alternative way:</u>

```
import numpy as np
x=np.array([[1,1],[1,-1]])#x+y=8,x-y=4
print(x)
print('\n')
x_inv=np.linalg.inv(x)
print('inverse array:','\n',x_inv)
print('\n')
y=np.array([[8],[4]])
print(y)
print('\n')
z=np.dot(x_inv,y)
print('soluation of this equation:','\n',z)
```

```
o/p:

[[ 1  1]
 [ 1 -1]]


inverse array:
 [[ 0.5  0.5]
 [ 0.5 -0.5]]


[[8]
 [4]]


soluation of this equation:
 [[6.]
 [2.]]
```

Or,

```
import numpy as np
x=np.array([[2,5,6],[1,-1,5],[0,4,3]]) #2x+3y+6z=8,x-y+5z=4,4y+3z=6
print(x)
print('\n')
y=np.array([[8],[4],[6]])
print(y)
print('\n')
z=np.linalg.solve(x,y)
print('solution of this equation achive by:','\n',z)
```

o/p:

```
[[ 2  5  6]
 [ 1 -1  5]
 [ 0  4  3]]


[[8]
 [4]
 [6]]


solution of this equation achive by:
 [[-1.02702703]
 [ 0.64864865]
 [ 1.13513514]]
```

Or,

```
import numpy as np
x=np.array([[2,5,6],[1,-1,5],[0,4,3]]) #2x+3y+6z=8,x-y+5z=4,4y+3z=6
print(x)
print('\n')
x_inv=np.linalg.inv(x)
print('inverse of this matrix:','\n',x_inv)
print('\n')
y=np.array([[8],[4],[6]])
print(y)
print('\n')
z=np.dot(x_inv,y)
print('soluation of this matrix:','\n',z)
```

o/p:

```
[[ 2  5  6]
 [ 1 -1  5]
 [ 0  4  3]]


inverse of this matrix:
 [[ 0.62162162 -0.24324324 -0.83783784]
 [ 0.08108108 -0.16216216  0.10810811]
 [-0.10810811  0.21621622  0.18918919]]
```

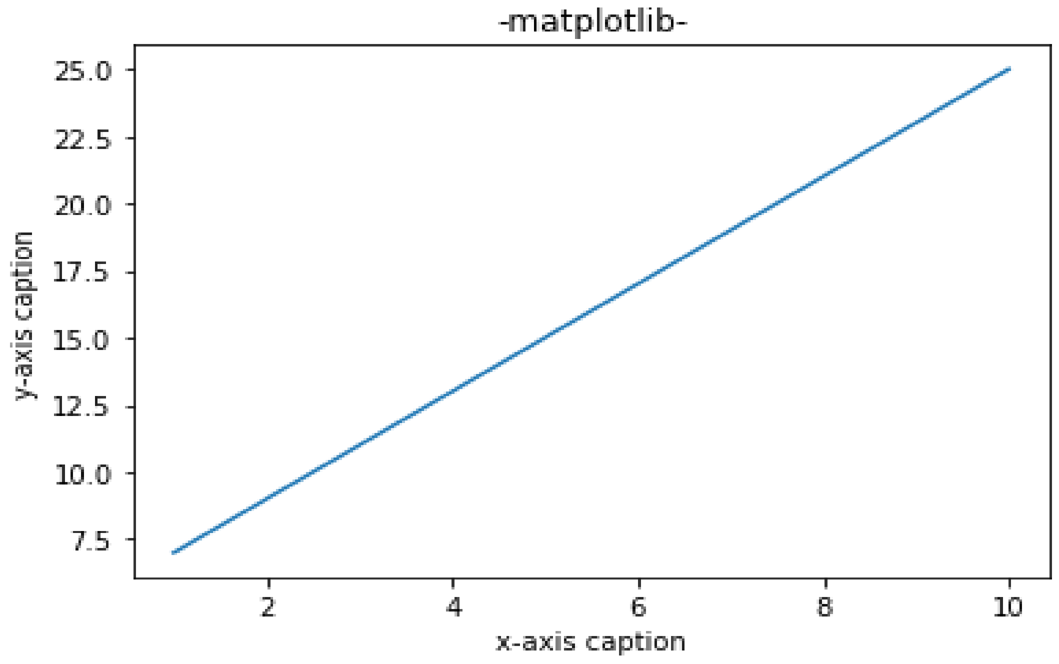
```
[[8]
 [4]
 [6]]


soluation of this matrix:
 [[-1.02702703]
 [ 0.64864865]
 [ 1.13513514]]

Or,

import numpy as np
from matplotlib import pyplot as plt
x=np.arange(1,11)
y=2*x+5
plt.title('-matplotlib-')
plt.xlabel('x-axis caption')
plt.ylabel('y-axis caption')
plt.plot(x,y)
plt.show()

o/p:
```

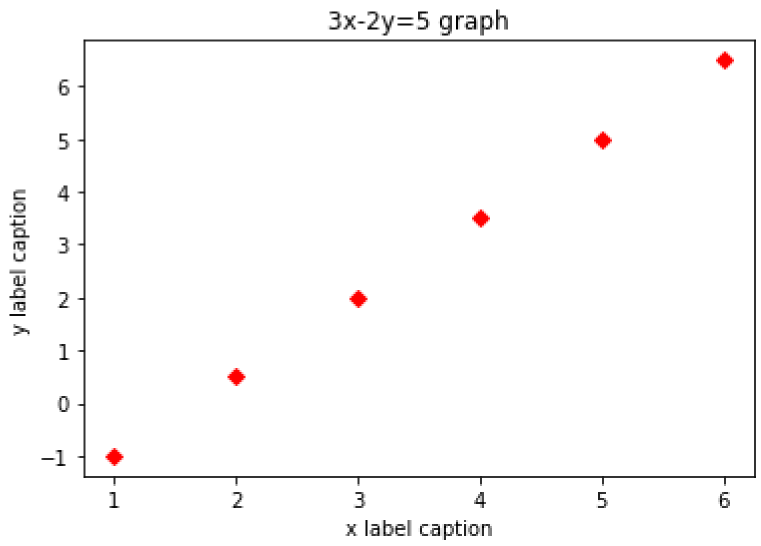


```
Or,

from matplotlib import pyplot as plt
import numpy as np
x=np.array([5,4,3,6,2,1])
y=(3*x-5)/2
```

```
plt.title('3x-2y=5 graph')
plt.xlabel('x label caption')
plt.ylabel('y label caption')
plt.plot(x,y,'Dr')
plt.show()
```

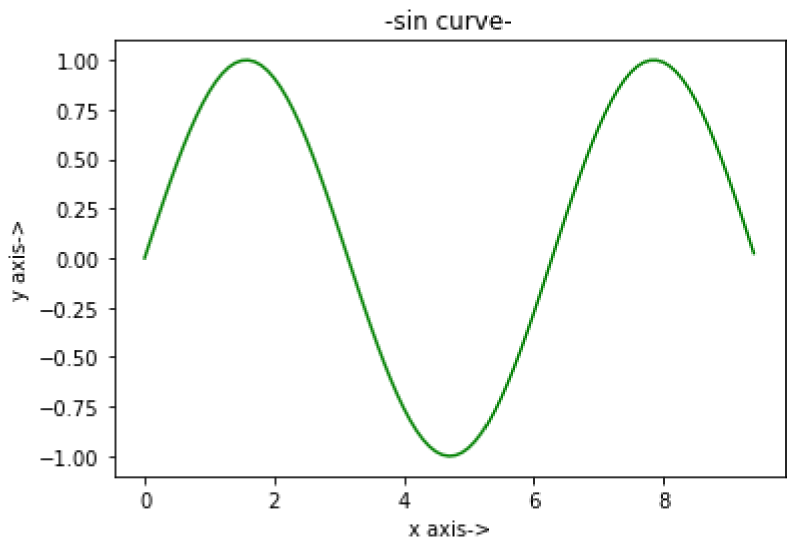
o/p:



Or,

```
from matplotlib import pyplot as plt
import numpy as np
x=np.arange(0,3*np.pi,0.1)
y=np.sin(x)
plt.title('-sin curve-')
plt.xlabel('x axis->')
plt.ylabel('y axis->')
plt.plot(x,y,'-g')
plt.show()
```

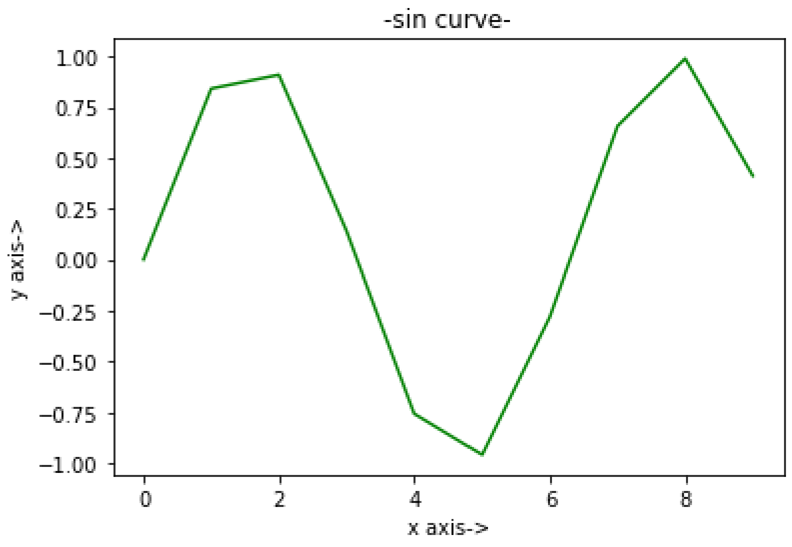
o/p:



Or,

```
from matplotlib import pyplot as plt
import numpy as np
x=np.arange(0,3*np.pi,1)
y=np.sin(x)
plt.title('-sin curve-')
plt.xlabel('x axis->')
plt.ylabel('y axis->')
plt.plot(x,y,'-g')
plt.show()
```

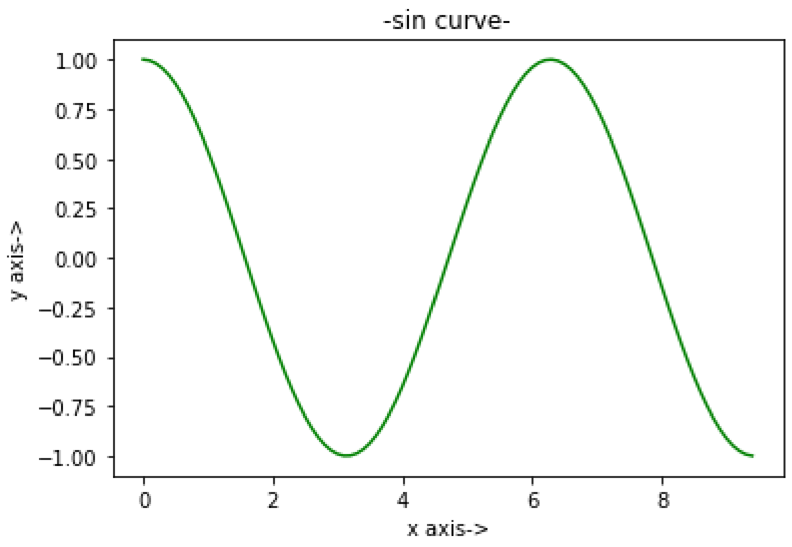
o/p:



Or,

```
from matplotlib import pyplot as plt
import numpy as np
x=np.arange(0,3*np.pi,0.1)
y=np.cos(x)
plt.title('-sin curve-')
plt.xlabel('x axis->')
plt.ylabel('y axis->')
plt.plot(x,y,'-g')
plt.show()
```

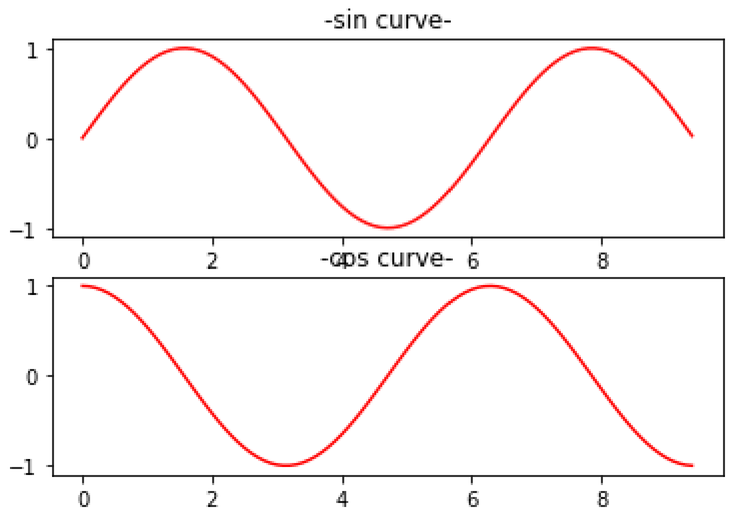
o/p:



Or,

```
from matplotlib import pyplot as plt
import numpy as np
x=np.arange(0,3*np.pi,0.1)
y_sin=np.sin(x)
y_cos=np.cos(x)
plt.subplot(2,1,1)
plt.plot(x,y_sin,'-r')
plt.title('-sin curve-')
plt.subplot(2,1,2)
plt.plot(x,y_cos,'-r')
plt.title('-cos curve-')
plt.show()
```

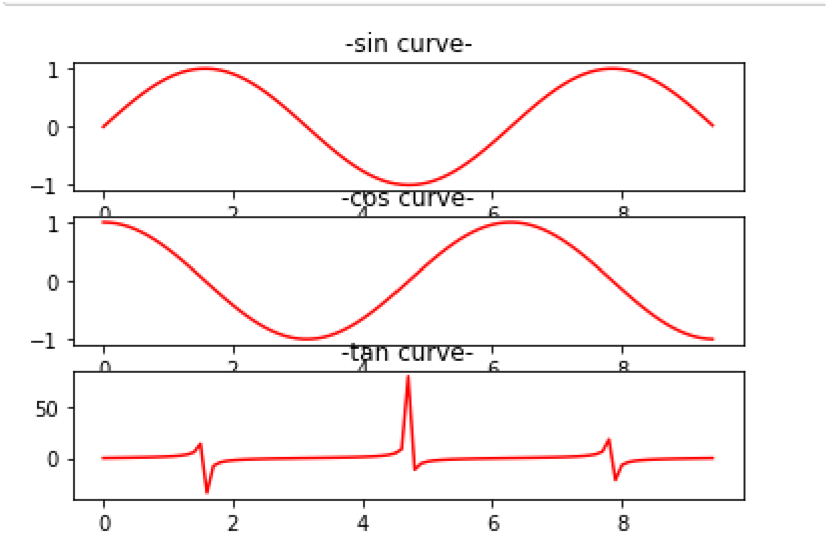
o/p:



Or,

```
from matplotlib import pyplot as plt
import numpy as np
x=np.arange(0,3*np.pi,0.1)
y_sin=np.sin(x)
y_cos=np.cos(x)
y_tan=np.tan(x)
plt.subplot(3,1,1)
plt.plot(x,y_sin,'-r')
plt.title('-sin curve-')
plt.subplot(3,1,2)
plt.plot(x,y_cos,'-r')
plt.title('-cos curve-')
plt.subplot(3,1,3)
plt.plot(x,y_tan,'-r')
plt.title('-tan curve-')
plt.show()
```

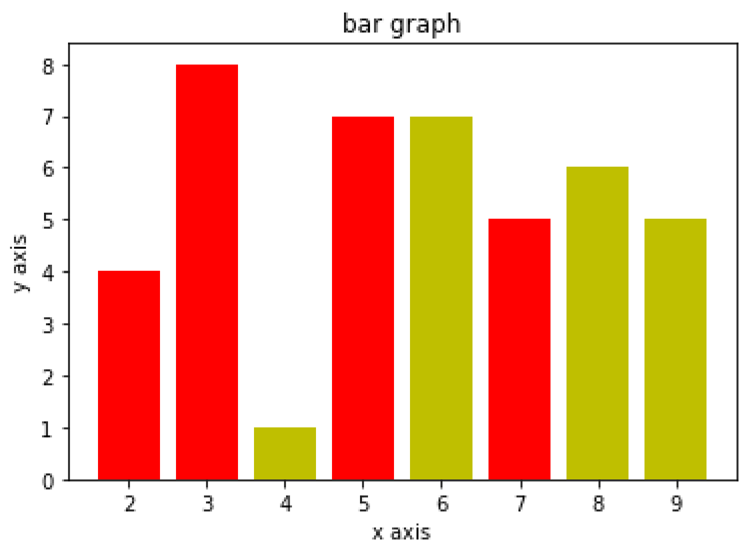
o/p:



Plt.bar()

```
from matplotlib import pyplot as plt
import numpy as np
x1=[7,2,3,5]
y1=[5,4,8,7]
x2=[9,8,6,4]
y2=[5,6,7,1]
plt.bar(x1,y1,color='r',align='center')
plt.bar(x2,y2,color='y',align='center')
plt.title('bar graph')
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.show()
```

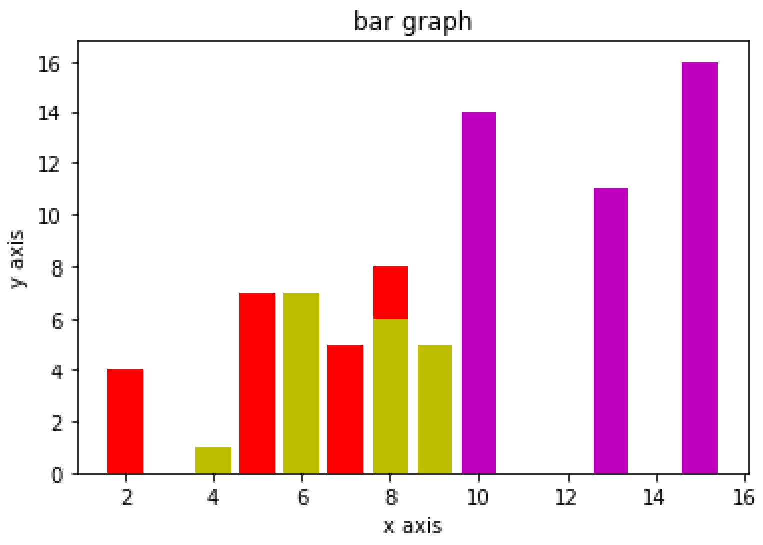
o/p:



Or,

```
from matplotlib import pyplot as plt
import numpy as np
x1=[7,2,8,5]
y1=[5,4,8,7]
x2=[9,8,6,4]
y2=[5,6,7,1]
x3=[10,15,13]
y3=[14,16,11]
plt.bar(x1,y1,color='r',align='center')
plt.bar(x2,y2,color='y',align='center')
plt.bar(x3,y3,color='m',align='center')
plt.title('bar graph')
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.show()
```

o/p:



Histogram:

```
import numpy as np
a=np.array([24,25,21,27,23,57,59,19,15,16,8,48,26,28,29,35,26,36,87,88,
82,37,39])
bins,hist=np.histogram(a,bins=[0,10,20,30,40,50,60,70,80,90,100])
print(hist)
print(bins)
```

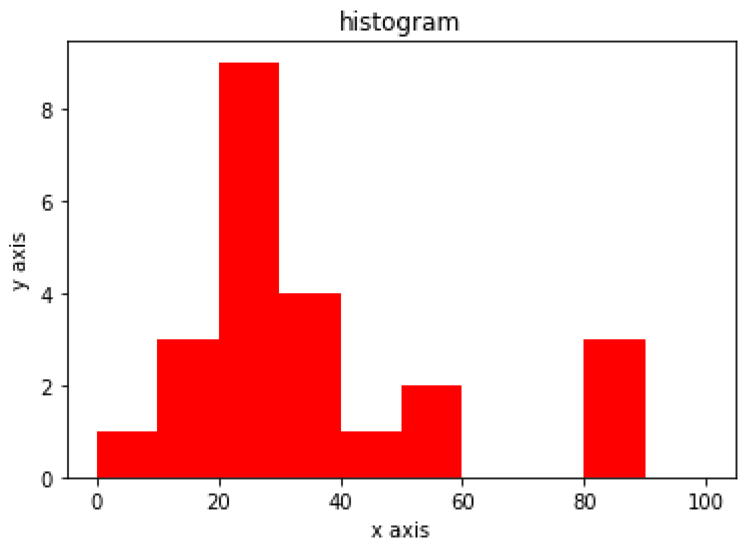
o/p:

```
[  0  10  20  30  40  50  60  70  80  90 100]
[1 3 9 4 1 2 0 0 3 0]
```

Or,

```
import numpy as np
a=np.array([24,25,21,27,23,57,59,19,15,16,8,48,26,28,29,35,26,36,87,88,
82,37,39])
plt.hist(a,bins=[0,10,20,30,40,50,60,70,80,90,100],color='r')
plt.title('histogram')
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.show()
```

o/p:



Or,

```
import numpy as np
a=np.array([24,25,21,27,23,57,59,19,15,16,9,48,26,28,29,35,26,36,87,88,
82,37,39])
bins,hist=np.histogram(a,bins=20)
print(bins)
print(hist)
```

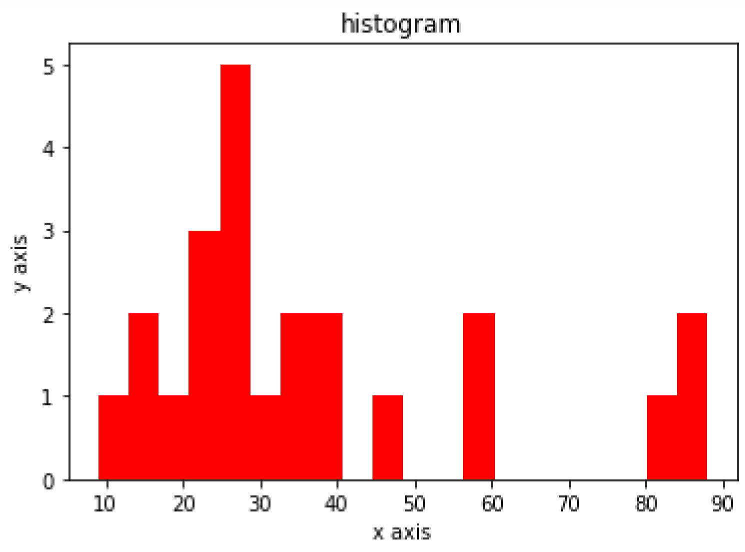
o/p:

```
[1 2 1 3 5 1 2 2 0 1 0 0 2 0 0 0 0 0 1 2]
[ 9.    12.95 16.9   20.85 24.8   28.75 32.7   36.65 40.6   44.55 48.5   52.45
 56.4   60.35 64.3   68.25 72.2   76.15 80.1   84.05 88.  ]
```

Or

```
import numpy as np
a=np.array([24,25,21,27,23,57,59,19,15,16,9,48,26,28,29,35,26,36,87,88,
82,37,39])
plt.hist(a,bins=20,color='r')
plt.title('histogram')
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.show()
```

o/p:



<u>save() & load()</u>

```
import numpy as np
a=np.array([1,2,3,4,45,6])
np.save('outfile',a)
b=np.load('outfile.npy')
print(b)

o/p:
```

```
[ 1  2  3  4 45  6]
```

Or,

```
import numpy as np
a=np.array([[1,2,3],[4,5,6],[8,9,7]])
np.save('myfile',a)
b=np.load('myfile.npy')
print(b)

o/p:
```

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

Or,

```
import numpy as np
a=np.array([[1,2,3],[4,5,6]])
```

```
      np.savetxt('out.txt',a)
      b=np.loadtxt('out.txt')
      print(b)

      o/p:
```

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

Or,

```
      import numpy as np
      a=np.array([[5,6,4],[2,3,4],[6,9,8]])
      np.savetxt('out.txt',a)
      b=np.loadtxt('out.txt')
      print(b)

      o/p:
```

```
[[5. 6. 4.]
 [2. 3. 4.]
 [6. 9. 8.]]
```

Or,

```
import numpy as np
a=np.array([[5,6,4],[2,3,4],[6,9,8]])
np.savetxt('out.txt',a)
b=np.loadtxt('out.txt',dtype=int)
print(b)

o/p:

[[5 6 4]
 [2 3 4]
 [6 9 8]]
```