creating numpy

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

In [4]:
```
a = np.array([1,2,3])
print(a) # vector
```

```
[1 2 3]
```

2d and 3d

In [5]:
```
b = np.array([[1,2,3],[2,3,4]]) # it also metrix formate
print(b)
```

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

In [45]:
```
# 3D
c = np.array([[[1,2],[3,4],[5,6]]])
print(c)

c.ndim
```

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

Out[45]:
```
3
```

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

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

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

In [9]:
```
# dtype
np.array([1,2,3],dtype = float)
```

Out[9]:
```
array([1., 2., 3.])
```

In [10]:
```
np.array([1,0,3],dtype = bool)
```

Out[10]:
```
array([ True, False,  True])
```

In [11]:
```
np.array([1,2,3],dtype = complex)
```

Out[11]:
```
array([1.+0.j, 2.+0.j, 3.+0.j])
```

np.arange

creating array

it works like loop range function (start,stop ,step)

In [13]:
```
np.arange(1,10)
```

Out[13]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [14]:
```
np.arange(1,10,2)
```

Out[14]:
```
array([1, 3, 5, 7, 9])
```

In [ ]:
```
# with reshape
reshape(row,column)
```

In [16]:
```
np.arange(1,11).reshape(5,2)
```

Out[16]:
```
array([[ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8],
       [ 9, 10]])
```

In [17]:
```
np.arange(1,11).reshape(2,5)
```

Out[17]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [18]:
```
np.arange(1,13).reshape(3,4)
```

Out[18]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

In [19]:
```
np.arange(1,13).reshape(2,6)
```

Out[19]:
```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12]])
```

In [37]:
```
np.arange(16).reshape(2,2,2,2)
```

Out[37]:
```
array([[[[ 0,  1],
         [ 2,  3]],

        [[ 4,  5],
         [ 6,  7]]],


       [[[ 8,  9],
         [10, 11]],

        [[12, 13],
         [14, 15]]]])
```

In [20]:
```
# np.ones
# it use in neural network for initialise
```

In [21]:
```
np.ones((3,4))
```

Out[21]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [22]:
```
# np.zeros
```

In [28]:
```
np.zeros((2,4))
```

Out[28]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [29]:
```
# np.random
```

In [31]:
```
np.random.random((3,4))
```

Out[31]:
```
array([[0.09126973, 0.75890653, 0.24578602, 0.43352899],
       [0.27630235, 0.63694283, 0.47314856, 0.52145831],
       [0.69012163, 0.94311731, 0.81172617, 0.23926643]])
```

In [ ]:
```
# np.linspace
full form lenear space
```

In [32]:
```
np.linspace(-10,10,10) #  (start,stop,how many number are you want in range)
```

Out[32]:
```
array([-10.        ,  -7.77777778,  -5.55555556,  -3.33333333,
        -1.11111111,   1.11111111,   3.33333333,   5.55555556,
         7.77777778,  10.        ])
```

In [33]:
```
 # np.identity
```

In [34]:
```
np.identity(3)
```

Out[34]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [35]:
```
np.identity(2)
```

Out[35]:
```
array([[1., 0.],
       [0., 1.]])
```

In [36]:
```
np.identity(4)
```

Out[36]:
```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

Array attributes

In [38]:
```
a1 = np.arange(10)
a2 = np.arange(12,dtype = float).reshape(3,4)
a3= np.arange(8).reshape(2,2,2)
```

In [39]:
```
a1 #evctor # 1d
```

Out[39]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [40]:
```
a2 # matrix 2d
```

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

In [41]:
```
a3 #3d
```

Out[41]:
```
array([[[0, 1],
        [2, 3]],

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

ndim

In [42]:
```
# ndim tells us dimention of array
a1.ndim
```

Out[42]:
```
1
```

In [43]:
```
a2.ndim
```

Out[43]:
```
2
```

In [44]:
```
a3.ndim
```

Out[44]:
```
3
```

shape

it tells us in given D- array how number are present here

In [48]:
```
# shape

a1.shape
```

Out[48]:
```
(10,)
```

In [49]:
```
a2.shape
```

Out[49]:
```
(3, 4)
```

In [50]:
```
print(a3.shape)
a3
```

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

Out[50]:
```
array([[[0, 1],
        [2, 3]],

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

In [ ]:
```
# size it tells us no.of item
```

In [51]:
```
a3.size
```

Out[51]:
```
8
```

In [53]:
```
a2.size
```

Out[53]:
```
12
```

In [60]:
```
# itemsize
# int = 4bytes
# float = 8 bytes
# memory allocation size
```

In [61]:
```
# itemsize

a3.itemsize
```

Out[61]:
```
4
```

In [56]:
```
a2.itemsize
```

Out[56]:
```
8
```

In [57]:
```
a1.itemsize
```

Out[57]:
```
4
```

changing Datatype

In [62]:
```
# astype
```

In [63]:
```
a3.dtype
```

Out[63]:
```
dtype('int32')
```

In [64]:
```
#here data size long to 16bytes so we use this (astype) to reduce the size of data ite
```

In [65]:
```
a3.astype(np.int16)
```

Out[65]:
```
array([[[0, 1],
        [2, 3]],

       [[4, 5],
        [6, 7]]], dtype=int16)
```

Array Operation

In [67]:
```
x = np.arange(12).reshape(3,4)
y = np.arange(12,24).reshape(3,4)
```

In [68]:
```
x
```

Out[68]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [69]:
```
y
```

Out[69]:
```
array([[12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23]])
```

In [ ]:
```
# scalar operatin
```

In [70]:
```
x*2
```

Out[70]:
```
array([[ 0,  2,  4,  6],
       [ 8, 10, 12, 14],
       [16, 18, 20, 22]])
```

In [71]:
```
x-2
```

Out[71]:
```
array([[-2, -1,  0,  1],
       [ 2,  3,  4,  5],
       [ 6,  7,  8,  9]])
```

In [72]:
```
x+2
```

Out[72]:
```
array([[ 2,  3,  4,  5],
       [ 6,  7,  8,  9],
       [10, 11, 12, 13]])
```

In [73]:
```
x/2
```

Out[73]:
```
array([[0. , 0.5, 1. , 1.5],
       [2. , 2.5, 3. , 3.5],
       [4. , 4.5, 5. , 5.5]])
```

In [74]:
```
x//2
```

Out[74]:
```
array([[0, 0, 1, 1],
       [2, 2, 3, 3],
       [4, 4, 5, 5]], dtype=int32)
```

In [75]:
```
x**2
```

Out[75]:
```
array([[  0,   1,   4,   9],
       [ 16,  25,  36,  49],
       [ 64,  81, 100, 121]], dtype=int32)
```

In [76]:
```
x
```

Out[76]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [77]:
```
# relation
```

In [78]:
```
y == 15
```

Out[78]:
```
array([[False, False, False,  True],
       [False, False, False, False],
       [False, False, False, False]])
```

In [79]:
```
y > 15
```

Out[79]:
```
array([[False, False, False, False],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True]])
```

vector operation

when you apply operation at two array then it acts

In [81]:
```
# arithmetic
```

In [82]:
```
x+y
```

Out[82]:
```
array([[12, 14, 16, 18],
       [20, 22, 24, 26],
       [28, 30, 32, 34]])
```

In [83]:
```
x-y
```

Out[83]:
```
array([[-12, -12, -12, -12],
       [-12, -12, -12, -12],
       [-12, -12, -12, -12]])
```

In [85]:
```
x*y # when shape will not same then will not work
```

Out[85]:
```
array([[  0,  13,  28,  45],
       [ 64,  85, 108, 133],
       [160, 189, 220, 253]])
```

Array function

In [2]:
```
a = np.random.random((3,3))
a = np.round(a*100)
a
```

Out[2]:
```
array([[ 69.,  59., 100.],
       [ 59.,  51.,   9.],
       [ 73.,  28.,   8.]])
```

In [3]:
```
# max/min/sum/prod
```

In [4]:
```
np.max(a)
```

Out[4]:
```
100.0
```

In [5]:
```
np.min(a)
```

Out[5]:
```
8.0
```

In [6]:
```
np.sum(a)
```

Out[6]:
```
456.0
```

In [7]:
```
np.prod(a)
```

Out[7]:
```
180275487235200.0
```

In [8]:
```
# 0 -> col and 1 -> row
np.max(a , axis =0)
```

Out[8]:
```
array([ 73.,  59., 100.])
```

In [9]:
```
np.max(a , axis =1)
```

Out[9]:
```
array([100.,  59.,  73.])
```

In [10]:
```
np.min(a , axis =1)
```

Out[10]:
```
array([59.,  9.,  8.])
```

In [11]:
```
np.sum(a , axis =1)
```

Out[11]:
```
array([228., 119., 109.])
```

In [12]:
```
np.sum(a , axis =0)
```

Out[12]:
```
array([201., 138., 117.])
```

In [13]:
```
np.prod(a , axis =0)
```

Out[13]:
```
array([297183.,  84252.,   7200.])
```

In [ ]:
```
# mean/median/st/var
```

In [14]:
```
# mean
np.mean(a)
```

Out[14]:
```
50.666666666666664
```

In [15]:
```
np.mean(a,axis = 0)
```

Out[15]:
```
array([67., 46., 39.])
```

In [17]:
```
np.median(a, axis = 1)
```

Out[17]:
```
array([69., 51., 28.])
```

In [18]:
```
np.std(a, axis = 1) # stander divition
```

Out[18]:
```
array([17.45470328, 21.92917894, 27.18251072])
```

In [20]:
```
np.var(a, axis = 1)  # vareince
```

Out[20]:
```
array([304.66666667, 480.88888889, 738.88888889])
```

In [21]:
```
np.sin(a)
```

Out[21]:
```
array([[-0.11478481,  0.63673801, -0.50636564],
       [ 0.63673801,  0.67022918,  0.41211849],
       [-0.67677196,  0.27090579,  0.98935825]])
```

dot product

In [23]:
```
x = np.arange(12).reshape(3,4)
v = np.arange(12,24).reshape(4,3)
```

In [24]:
```
np.dot(x,v)
```

Out[24]:
```
array([[114, 120, 126],
       [378, 400, 422],
       [642, 680, 718]])
```

In [25]:
```
np.dot(v,x)
```

Out[25]:
```
array([[164, 203, 242, 281],
       [200, 248, 296, 344],
       [236, 293, 350, 407],
       [272, 338, 404, 470]])
```

In [26]:
```
# log and exponent
```

In [27]:
```
np.log(x)
```

```
C:\Users\DESKTOP\AppData\Local\Temp\ipykernel_1516\1277889159.py:1: RuntimeWarning: d
ivide by zero encountered in log
  np.log(x)
```

Out[27]:
```
array([[      -inf, 0.        , 0.69314718, 1.09861229],
       [1.38629436, 1.60943791, 1.79175947, 1.94591015],
       [2.07944154, 2.19722458, 2.30258509, 2.39789527]])
```

In [28]:
```
np.exp(x)
```

Out[28]:
```
array([[1.00000000e+00, 2.71828183e+00, 7.38905610e+00, 2.00855369e+01],
       [5.45981500e+01, 1.48413159e+02, 4.03428793e+02, 1.09663316e+03],
       [2.98095799e+03, 8.10308393e+03, 2.20264658e+04, 5.98741417e+04]])
```

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

In [30]:
```
x = np.random.random((2,3))
x
```

Out[30]:
```
array([[0.50778229, 0.91363883, 0.14032692],
       [0.61277731, 0.86924197, 0.92394721]])
```

In [31]:
```
x = np.random.random((2,3))*100
x
```

Out[31]:
```
array([[88.55275408, 28.1362232 , 92.60362129],
       [51.67377273, 17.08162434, 20.55129989]])
```

In [34]:
```
np.round(np.random.random((2,3))*100)
```

Out[34]:
```
array([[67.,  0., 66.],
       [ 6., 81., 37.]])
```

In [35]:
```
np.floor(np.random.random((2,3))*100)
```

Out[35]:
```
array([[39., 80., 97.],
       [25.,  1., 36.]])
```

In [37]:
```
np.ceil(np.random.random((2,3))*100)
```

Out[37]:
```
array([[17., 11., 43.],
       [48., 13., 51.]])
```

Indexing and Slicing

In [24]:
```
a = np.arange(10)
b = np.arange(12).reshape(3,4)
c = np.arange(8).reshape(2,2,2)
```

In [39]:
```
# indexing
```

In [42]:
```
a
```

Out[42]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [43]:
```
a[-1]
```

Out[43]:
```
9
```

In [44]:
```
a[4]
```

Out[44]:
```
4
```

In [45]:
```
b
```

Out[45]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [ ]:
```
# for 2D array indexing syntax arr[row ,col]
```

In [46]:
```
b[2,3]
```

Out[46]:
```
11
```

In [48]:
```
b[0,3]
```

Out[48]:
```
3
```

In [49]:
```
b[2,2]
```

Out[49]:
```
10
```

In [50]:
```
c
```

Out[50]:
```
array([[[0, 1],
        [2, 3]],

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

In [51]:
```
#arr[which D in,row ,col]
```

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

Out[52]:
```
6
```

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

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

In [54]:
```
c[1,1,1]
```

Out[54]:
```
7
```

In [55]:
```
# slicing
```

In [57]:
```
a[2:5]
```

Out[57]:
```
array([2, 3, 4])
```

In [58]:
```
a[::-1]
```

Out[58]:
```
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

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

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

In [60]:
```
# in 2D
```

In [61]:
```
b
```

Out[61]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

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

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

In [64]:
```
b[:,2]
```

Out[64]:
```
array([ 2,  6, 10])
```

In [65]:
```
b[1:,1:3]
```

Out[65]:
```
array([[ 5,  6],
       [ 9, 10]])
```

In [70]:
```
b[::2,::3]
```

Out[70]:
```
array([[ 0,  3],
       [ 8, 11]])
```

In [75]:
```
b[0:,0:]
```

Out[75]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [76]:
```
b[:2,:3]
```

Out[76]:
```
array([[0, 1, 2],
       [4, 5, 6]])
```

In [77]:
```
b
```

Out[77]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [80]:
```
b[0::,::3]
```

Out[80]:
```
array([[ 0,  3],
       [ 4,  7],
       [ 8, 11]])
```

In [81]:
```
b[::2]
```

Out[81]:
```
array([[ 0,  1,  2,  3],
       [ 8,  9, 10, 11]])
```

In [82]:
```
b[::2,1::2]
```

Out[82]:
```
array([[ 1,  3],
       [ 9, 11]])
```

In [85]:
```
b[1,0::3]
```

Out[85]:
```
array([4, 7])
```

In [86]:
```
b[1,::3]
```

Out[86]:
```
array([4, 7])
```

In [87]:
```
b
```

Out[87]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [104…
```
b[0:2,1:]
```

Out[104]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [105…
```
b[0:2,1::2]
```

Out[105]:
```
array([[1, 3],
       [5, 7]])
```

In [2]:
```
c = np.arange(8).reshape(2,2,2)
```

In [3]:
```
c
```

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

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

In [36]:
```
c = np.arange(27).reshape(3,3,3)
```

In [37]:
```
c
```

Out[37]:
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [7]:
```
c[1]
```

Out[7]:
```
array([[ 9, 10, 11],
       [12, 13, 14],
       [15, 16, 17]])
```

In [8]:
```
c[0::2]
```

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

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [9]:
```
c[::2]
```

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

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [10]:
```
c[0,1]
```

Out[10]:
```
array([3, 4, 5])
```

In [12]:
```
c[1,:,1]
```

Out[12]:
```
array([10, 13, 16])
```

In [13]:
```
c
```

Out[13]:
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [14]:
```
c[2,0,::2]
```

Out[14]:
```
array([18, 20])
```

In [15]:
```
c[1,1,2]
```

Out[15]:
```
14
```

In [16]:
```
c[0,2,2]
```

Out[16]:
```
8
```

In [18]:
```
c[2,1:,1:]
```

Out[18]:
```
array([[22, 23],
       [25, 26]])
```

In [19]:
```
c[::2,0,::2]
```

Out[19]:
```
array([[ 0,  2],
       [18, 20]])
```

In [21]:
```
c[:,:,1]
```

Out[21]:
```
array([[ 1,  4,  7],
       [10, 13, 16],
       [19, 22, 25]])
```

In [22]:
```
c[:,1,0]
```

Out[22]:
```
array([ 3, 12, 21])
```

Iterating

In [29]:
```
a
for i in a:
    print(i)
```

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

In [31]:
```
b
for i in b:
    print(i)
```

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

In [27]:
```
c
```

Out[27]:
```
array([[[0, 1],
        [2, 3]],

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

In [32]:
```
for i in c:
    print(i)
```

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

In [33]:
```
for i in np.nditer(c):
    print(i)
```

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

Transpose

it convers the row into column

In [40]:
```
# transpose
c
```

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

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [39]:
```
np.transpose(c)
```

Out[39]:
```
array([[[ 0,  9, 18],
        [ 3, 12, 21],
        [ 6, 15, 24]],

       [[ 1, 10, 19],
        [ 4, 13, 22],
        [ 7, 16, 25]],

       [[ 2, 11, 20],
        [ 5, 14, 23],
        [ 8, 17, 26]]])
```

In [41]:
```
b
```

Out[41]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [42]:
```
np.transpose(b)
```

Out[42]:
```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

In [47]:
```
# simple syntax of transpose
## array_name.T
```

In [46]:
```
b.T
```

Out[46]:
```
array([[ 0,  4,  8],
       [ 1,  5,  9],
       [ 2,  6, 10],
       [ 3,  7, 11]])
```

In [48]:
```
c.T
```

Out[48]:
```
array([[[ 0,  9, 18],
        [ 3, 12, 21],
        [ 6, 15, 24]],

       [[ 1, 10, 19],
        [ 4, 13, 22],
        [ 7, 16, 25]],

       [[ 2, 11, 20],
        [ 5, 14, 23],
        [ 8, 17, 26]]])
```

In [49]:
```
# ravel
# the main aim is to convert in 1D array
```

In [50]:
```
c.ravel()
```

Out[50]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26])
```

In [51]:
```
np.ravel(b)
```

Out[51]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

In [52]:
```
c
```

Out[52]:
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [62]:
```
# hstack()
## it is use to add two or more array with horegentelly
```

In [54]:
```
v=c*2
print(v)
```

```
[[[ 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]]]
```

In [55]:
```
v
```

Out[55]:
```
array([[[ 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]]])
```

In [58]:
```
np.hstack((c,v))
```

Out[58]:
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8],
        [ 0,  2,  4],
        [ 6,  8, 10],
        [12, 14, 16]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17],
        [18, 20, 22],
        [24, 26, 28],
        [30, 32, 34]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26],
        [36, 38, 40],
        [42, 44, 46],
        [48, 50, 52]]])
```

In [60]:
```
v = b+2
print(v)
```

```
[[ 2  3  4  5]
 [ 6  7  8  9]
 [10 11 12 13]]
```

In [61]:
```
np.hstack((b,v))
```

Out[61]:
```
array([[ 0,  1,  2,  3,  2,  3,  4,  5],
       [ 4,  5,  6,  7,  6,  7,  8,  9],
       [ 8,  9, 10, 11, 10, 11, 12, 13]])
```

In [63]:
```
# vstack()
## ## it is use to add two or more array with verticaly
```

In [65]:
```
np.vstack((b,v))
```

Out[65]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [ 2,  3,  4,  5],
       [ 6,  7,  8,  9],
       [10, 11, 12, 13]])
```

In [67]:
```
np.vstack((b,v,v)) # it works at same dimention  2*2 , 2*2
```

Out[67]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [ 2,  3,  4,  5],
       [ 6,  7,  8,  9],
       [10, 11, 12, 13],
       [ 2,  3,  4,  5],
       [ 6,  7,  8,  9],
       [10, 11, 12, 13]])
```

splitting

it is reverse operation of stack

In [68]:
```
d= np.hstack((b,v))
d
```

Out[68]:
```
array([[ 0,  1,  2,  3,  2,  3,  4,  5],
       [ 4,  5,  6,  7,  6,  7,  8,  9],
       [ 8,  9, 10, 11, 10, 11, 12, 13]])
```

In [69]:
```
# hsplit(arr,part_no)
```

In [70]:
```
np.hsplit(d,2)
```

Out[70]:
```
[array([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]]),
 array([[ 2,  3,  4,  5],
        [ 6,  7,  8,  9],
        [10, 11, 12, 13]])]
```

In [72]:
```
np.hsplit(d,4)
```

Out[72]:
```
[array([[0, 1],
        [4, 5],
        [8, 9]]),
 array([[ 2,  3],
        [ 6,  7],
        [10, 11]]),
 array([[ 2,  3],
        [ 6,  7],
        [10, 11]]),
 array([[ 4,  5],
        [ 8,  9],
        [12, 13]])]
```

In [74]:
```
p=np.hsplit(d,4)
p
```

Out[74]:
```
[array([[0, 1],
        [4, 5],
        [8, 9]]),
 array([[ 2,  3],
        [ 6,  7],
        [10, 11]]),
 array([[ 2,  3],
        [ 6,  7],
        [10, 11]]),
 array([[ 4,  5],
        [ 8,  9],
        [12, 13]])]
```

In [75]:
```
p[1]
```

Out[75]:
```
array([[ 2,  3],
       [ 6,  7],
       [10, 11]])
```

In [76]:
```
# vsplit()
```

In [78]:
```
np.vsplit(d,3)
```

Out[78]:
```
[array([[0, 1, 2, 3, 2, 3, 4, 5]]),
 array([[4, 5, 6, 7, 6, 7, 8, 9]]),
 array([[ 8,  9, 10, 11, 10, 11, 12, 13]])]
```

In [ ]:
```
c =np.vstack((b,v,v))
```

In [79]:
```
c
```

Out[79]:
```
array([[[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8]],

       [[ 9, 10, 11],
        [12, 13, 14],
        [15, 16, 17]],

       [[18, 19, 20],
        [21, 22, 23],
        [24, 25, 26]]])
```

In [80]:
```
np.vsplit(c,3)
```

Out[80]:
```
[array([[[0, 1, 2],
         [3, 4, 5],
         [6, 7, 8]]]),
 array([[[ 9, 10, 11],
         [12, 13, 14],
         [15, 16, 17]]]),
 array([[[18, 19, 20],
         [21, 22, 23],
         [24, 25, 26]]])]
```

In [ ]:

In [ ]: