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

In [2]:
```
import numpy as np
arr=np.array([1,2,3,4,5,6,7,8,9,10,11,12])  # this is 1d array
newarr1=arr.reshape(2,3,2) #(2x3x2) # here the reshape function converted into 3
newarr2=arr.reshape(4,3) # it is covert into 2d array bcuz it have two parameter
print(newarr1)
print(newarr2)
```

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

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

In [3]:
```
newarr1
```

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

       [[ 7,  8],
        [ 9, 10],
        [11, 12]]])
```

In [4]:
```
newarr1=[...,2]
print(newarr1)
```

```
[Ellipsis, 2]
```

In [5]:
```
newarr1=[1,...]
print(newarr1)
```

```
[1, Ellipsis]
```

In [6]:
```
newarr2
```

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

In [7]:
```
newarr2=[...,1]
```

In [8]:
```
print(newarr2)
```

```
[Ellipsis, 1]
```

[]

In [9]:
```
import numpy as np
b=np.arange(0,20)
print(b)
```

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

In [10]:
```
b1=np.reshape(b,(5,4))
print(b1)
```

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

In [11]:
```
b1[0,0]
```

Out[11]:
```
0
```

In [12]:
```
b1[-1]  #here it print the first line in the matrix
```

Out[12]:
```
array([16, 17, 18, 19])
```

In [13]:
```
b1[1:3,1:4]  # here the indexing is bet [1:3]-it means 1st index 3rd value
                  #indexing bet [1:4]- it means 1st index and 4th value
```

Out[13]:
```
array([[ 5,  6,  7],
       [ 9, 10, 11]])
```

In [14]:
```
b1
```

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

In [15]:
```
b1>15
```

Out[15]:
```
array([[False, False, False, False],
       [False, False, False, False],
       [False, False, False, False],
       [False, False, False, False],
       [ True,  True,  True,  True]])
```

In [16]:
```
b1>=15
```

Out[16]:
```
array([[False, False, False, False],
       [False, False, False, False],
       [False, False, False, False],
       [False, False, False,  True],
       [ True,  True,  True,  True]])
```

In [17]:
```
b1<=15
```

Out[17]:
```
array([[ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [False, False, False, False]])
```

In [18]:
```
b1==15
```

```
Out[18]: array([[False, False, False, False],
                [False, False, False, False],
                [False, False, False, False],
                [False, False, False,  True],
                [False, False, False, False]])
```

```
In [19]: b1!=15
```

```
Out[19]: array([[ True,  True,  True,  True],
                [ True,  True,  True,  True],
                [ True,  True,  True,  True],
                [ True,  True,  True, False],
                [ True,  True,  True,  True]])
```

```
In [20]: a=np.array([[0,1,2,3,4],[10,11,12,13,14],[20,21,22,23,24],[30,31,32,33,34]])
         c=np.array([True,False,True,False])
         print(a)
```

```
[[ 0  1  2  3  4]
 [10 11 12 13 14]
 [20 21 22 23 24]
 [30 31 32 33 34]]
```

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

```
Out[21]: array([[ 0,  1,  2,  3,  4],
                [20, 21, 22, 23, 24]])
```

```
In [22]: d=np.array([False,True,True,False,True])
         a[:,d]
```

```
Out[22]: array([[ 1,  2,  4],
                [11, 12, 14],
                [21, 22, 24],
                [31, 32, 34]])
```

```
In [23]: a
```

```
Out[23]: array([[ 0,  1,  2,  3,  4],
                [10, 11, 12, 13, 14],
                [20, 21, 22, 23, 24],
                [30, 31, 32, 33, 34]])
```

```
In [24]: i=np.array([0,1,2,1])
         j=np.array([1,2,3,4])
         a[i,j]
```

```
Out[24]: array([ 1, 12, 23, 14])
```

abs() and Absolute both are same

```
In [25]: import numpy as np
         arr=np.arange(-4,5).reshape(3,3)
         arr
```

```
Out[25]: array([[-4, -3, -2],
                [-1,  0,  1],
                [ 2,  3,  4]])
```

```
In [26]: abs(arr)    # here abs()-function which remove negative(-) values
```

```
Out[26]: array([[4, 3, 2],
                [1, 0, 1],
                [2, 3, 4]])
```

```
In [27]: arr1=1.2+5j
         arr1
```

```
Out[27]: (1.2+5j)
```

```
In [28]: abs(arr1) #UNNDER-ROOT OF Square of of real part & imaginary part
```

```
Out[28]: 5.141984052872976
```

```
In [29]: np.absolute(arr1)
```

```
Out[29]: 5.141984052872976
```

```
In [30]: np.absolute([-10,-15])
```

```
Out[30]: array([10, 15])
```

Accumulate

```
In [31]: import numpy as np
```

```
In [32]: a1=np.arange(0,10)
         print(a1)
```

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

```
In [33]: np.add.accumulate(a1)  # progressive sum
```

```
Out[33]: array([ 0,  1,  3,  6, 10, 15, 21, 28, 36, 45])
```

```
In [34]: a2=np.arange(1,6)
         print(a2)
```

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

```
In [35]: np.multiply.accumulate(a2)
```

```
Out[35]: array([  1,   2,   6,  24, 120])
```

```
In [36]: ac=np.array([[1,2,3],[4,5,6]])
         ac
```

```
Out[36]: array([[1, 2, 3],
                [4, 5, 6]])
```

```
In [37]: np.add.accumulate(ac) # every colomn got accumulate
```

```
Out[37]: array([[1, 2, 3],
                [5, 7, 9]])
```

```
In [38]: np.multiply.accumulate(ac)
```

```
Out[38]: array([[ 1,  2,  3],
                [ 4, 10, 18]])
```

ADD() function

```
In [39]: import numpy as np
```

```
In [40]: ar1=[2,-6]
         ar2=[1,3]
         ar3=[2.2,8.7]
         ar4=[7,-3]
         print(ar1,ar2,ar3,ar4)

[2, -6] [1, 3] [2.2, 8.7] [7, -3]
```

```
In [41]: np.add(ar1,ar2)

Out[41]: array([ 3, -3])
```

```
In [42]: np.add(ar3,ar4)

Out[42]: array([9.2, 5.7])
```

```
In [43]: c1=5+2j
```

```
In [44]: np.add(ar3,c1)

Out[44]: array([ 7.2+2.j, 13.7+2.j])
```

ALL() Function

```
In [45]: a=np.arange(0,10)
         a

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

```
In [46]: np.all(a>5)

Out[46]: False
```

```
In [47]: np.all(a>=0)

Out[47]: True
```

```
In [48]: b7=np.array([True,False,True,False])
         np.all(b7)

Out[48]: False
```

allclose() function

```
In [49]: np.allclose([0.1, 0.003], [0.11, 0.003])

Out[49]: False
```

```
In [50]: np.allclose([0.11, 0.0033], [0.1100005, 0.0030001])
```

```
Out[50]:  False
```

```
In [51]:  np.allclose([1e10,1e-8], [1.0001e10,1e-9])
```

```
Out[51]:  False
```

```
In [52]:  np.allclose([1e10,1e-8], [1.00001e10,1e-9])
```

```
Out[52]:  True
```

Alltrue- same as all funtion

```
In [53]:  b=np.array([True,False,True,False])
          np.alltrue(b)
```

```
Out[53]:  False
```

```
In [54]:  a=np.array([1,2,3,5,7])
          np.alltrue(a>=5)
```

```
Out[54]:  False
```

```
In [55]:  a=np.array([1,2,3,5,7])
          np.alltrue(a)
```

```
Out[55]:  True
```

```
In [56]:  np.alltrue(a>0)
```

```
Out[56]:  True
```

Angle() function

```
In [57]:  np.angle([5+3j,1j,1.0])  # here this radian unit
```

```
Out[57]:  array([0.5404195 , 1.57079633, 0.        ])
```

```
In [58]:  np.angle([5+3j,1j,5],deg=True)  # in Degree
```

```
Out[58]:  array([30.96375653, 90.        ,  0.        ])
```

```
In [59]:  np.angle(2+2j,deg=True)
```

```
Out[59]:  45.0
```

```
In [60]:  np.angle(2+2j,deg=False)
```

```
Out[60]:  0.7853981633974483
```

Any() function

```
In [61]:  x2=np.array([True,False,True,False])
          x2
```

```
Out[61]:  array([ True, False,  True, False])
```

```
In [62]: np.any(x2)

Out[62]: True

In [63]: x3=np.arange(0,20)
         x3

Out[63]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
                17, 18, 19])

In [64]: any(x3>=5)

Out[64]: True
```

Append() function

```
In [65]: c3=np.append([1,2,3],[[4,5,6],[7,8,9]])
         c3

Out[65]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [66]: c4=np.append(c3,10)

In [67]: c4

Out[67]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])

In [68]: np.append(c4,('ABC',1+2j))

Out[68]: array(['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'ABC', '(1+2j)'],
               dtype='<U64')

In [69]: c5=np.arange(10,81,10)

In [70]: c5

Out[70]: array([10, 20, 30, 40, 50, 60, 70, 80])

In [71]: np.append(c5,(90,100,110))

Out[71]: array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100, 110])

In [72]: c10=np.reshape(c5,(2,4))

In [73]: c10

Out[73]: array([[10, 20, 30, 40],
                [50, 60, 70, 80]])

In [74]: c11 = np.append(c10, [[70, 80,90,100]], axis = 0)   # here at  Row
         c11

Out[74]: array([[ 10,  20,  30,  40],
                [ 50,  60,  70,  80],
                [ 70,  80,  90, 100]])
```

```
In [75]: c12=np.append(c11,[[100],[120],[130]],axis=1)#here we adding the atributes(colou
         c12
```

```
Out[75]: array([[ 10,  20,  30,  40, 100],
                [ 50,  60,  70,  80, 120],
                [ 70,  80,  90, 100, 130]])
```

Apply_along_axis()

```
In [76]: def func1(Q):
             return(Q[0]+Q[-1])
         abc=np.array([[1,2,3],[4,5,6],[7,8,9]])
         abc
```

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

```
In [77]: np.apply_along_axis(func1,1,abc)
```

```
Out[77]: array([ 4, 10, 16])
```

```
In [78]: np.apply_along_axis(func1,0,abc)
```

```
Out[78]: array([ 8, 10, 12])
```

```
In [81]: def func2(P):
             return(P[0]+P[-1])*2
         abc1=np.array([[1,2,3],[4,5,6],[7,8,9]])
         abc1
```

```
Out[81]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])
```

```
In [82]: np.apply_along_axis(func1,1,abc1)
```

```
Out[82]: array([ 4, 10, 16])
```

```
In [83]: np.apply_along_axis(func1,0,abc1)
```

```
Out[83]: array([ 8, 10, 12])
```

Apply_over_axes() function

```
In [87]: D=np.arange(24)
         D
```

```
Out[87]: 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])
```

```
In [88]: D=np.reshape(D,(2,3,4))
         D
```

```
Out[88]:  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]]])
```

```
In [90]:  dq=np.apply_over_axes(np.sum,D,[0,2])
          dq
```

```
Out[90]:  array([[[ 60],
                  [ 92],
                  [124]]])
```

Arrange() function

```
In [92]:  np.arange(9)
```

```
Out[92]:  array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

```
In [93]:  np.arange(11.0)
```

```
Out[93]:  array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

```
In [94]:  np.arange(8,dtype=float)
```

```
Out[94]:  array([0., 1., 2., 3., 4., 5., 6., 7.])
```

```
In [96]:  np.arange(5,10) #start,stop
```

```
Out[96]:  array([5, 6, 7, 8, 9])
```

```
In [97]:  np.arange(3,30,9)  # (strat,stop,num)
```

```
Out[97]:  array([ 3, 12, 21])
```

```
In [99]:  np.arccosh([np.e,10.0])
```

```
Out[99]:  array([1.65745445, 2.99322285])
```

```
In [100…  np.arccosh(1)
```

```
Out[100…  0.0
```

```
In [101…  np.arccos([1,-1])
```

```
Out[101…  array([0.        , 3.14159265])
```

```
In [103…  np.arcsin([1,-1,0])
```

```
Out[103…  array([ 1.57079633, -1.57079633,  0.        ])
```

```
In [104…  np.arcsinh([np.e,10.0])
```

```
Out[104…  array([1.72538256, 2.99822295])
```

In [105…

```
np.arctan([0,1,-1])
```

Out[105…

```
array([ 0.        ,  0.78539816, -0.78539816])
```

In [106…

```
x=np.array([-1,+1,+1,-1])
y=np.array([-1,-1,+1,+1])
np.arctan2(y,x)*180/np.pi
```

Out[106…

```
array([-135.,  -45.,   45.,  135.])
```

In [107…

```
np.arctan2(y,x)
```

Out[107…

```
array([-2.35619449, -0.78539816,  0.78539816,  2.35619449])
```

In [108…

```
np.arctan2([1.,-1.],[0.,0.])
```

Out[108…

```
array([ 1.57079633, -1.57079633])
```

In [109…

```
np.arctan2([0,1],[1,0])
```

Out[109…

```
array([0.        , 1.57079633])
```

In [111…

```
np.arctanh([0,-0.5])
```

Out[111…

```
array([ 0.        , -0.54930614])
```

In [112…

```
j=np.array([0,11,2,95,-5,44])
```

In [113…

```
np.argmax(j)
```

Out[113…

```
3
```

In [115…

```
j1=np.arange(6).reshape(2,3)+10
j1
```

Out[115…

```
array([[10, 11, 12],
       [13, 14, 15]])
```

In [116…

```
np.argmax(j1)
```

Out[116…

```
5
```

In [118…

```
J2 = np.array([[1, 9, 0, 4], [2, 0, 8, -1]])
J2
```

Out[118…

```
array([[ 1,  9,  0,  4],
       [ 2,  0,  8, -1]])
```

In [119…

```
np.argmax(J2, axis =0)
```

Out[119…

```
array([1, 0, 1, 0], dtype=int64)
```

In [120…

```
np.argmax(J2, axis=1)
```

Out[120…

```
array([1, 2], dtype=int64)
```

In [121…
```
d4 =np.array([0, 11, 95, 2, -5, 55])
```

In [122…
```
np.argmin(d4)
```

Out[122…
```
4
```

In [123…
```
d9 = np.arange(6).reshape(2,3) +10
d9
```

Out[123…
```
array([[10, 11, 12],
       [13, 14, 15]])
```

In [124…
```
np.argmin(d9)
```

Out[124…
```
0
```

In [125…
```
da2=np.array([[50,50,10],[60,10,40]])
da2
```

Out[125…
```
array([[50, 50, 10],
       [60, 10, 40]])
```

In [126…
```
np.argmin(da2)
```

Out[126…
```
2
```

In [127…
```
np.argmin(da2, axis=0)
```

Out[127…
```
array([0, 1, 0], dtype=int64)
```

In [128…
```
np.argmin(da2, axis=1)
```

Out[128…
```
array([2, 1], dtype=int64)
```

In [129…
```
AB1 = np.array([ 2, 0,  1, 5, 4, 1, 9])
```

In [130…
```
np.argsort(AB1)
```

Out[130…
```
array([1, 2, 5, 0, 4, 3, 6], dtype=int64)
```

In [131…
```
BQ = np.argsort(AB1)
BQ
```

Out[131…
```
array([1, 2, 5, 0, 4, 3, 6], dtype=int64)
```

In [132…
```
AB1[BQ]
```

Out[132…
```
array([0, 1, 1, 2, 4, 5, 9])
```

In [133…
```
aq1 = np.array([[8,4,1],[2,0,9]])
aq1
```

Out[133…
```
array([[8, 4, 1],
       [2, 0, 9]])
```

In [134…
```
JQ1 = aq1.argsort(axis=0) # sorts on columns. NOT the same as aq1.sort(axis=1)
```

```
JQ1
```

Out[134…

```
array([[1, 1, 0],
       [0, 0, 1]], dtype=int64)
```

In [135…

```
aq1[JQ1,[[0,1,2], [0,1,2]]] # 2-D arrays need fancy indexing if you want to sort
```

Out[135…

```
array([[2, 0, 1],
       [8, 4, 9]])
```

In [136…

```
JQ1 = aq1.argsort(axis=1)  # sort along rows. Can use aq1.argsort(axis=-1) for l
JQ1
```

Out[136…

```
array([[2, 1, 0],
       [1, 0, 2]], dtype=int64)
```

In [137…

```
np.array([1, 2, 3.0])
```

Out[137…

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

In [138…

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

Out[138…

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

In [139…

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

Out[139…

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

In [140…

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

Out[140…

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

In [141…

```
np.array(1, copy=0, subok=1, ndmin=1) # basically equivalent to atleast_1d
```

Out[141…

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

In [142…

```
np.array(1, copy=0, subok=1, ndmin=2)
```

Out[142…

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

In [143…

```
np.array(1, subok=1, ndmin=2)
```

Out[143…

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

In [144…

```
c4 = np.array([[1,2,3,4],[5,6,7,8]])
c4
```

Out[144…

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

In [145…

```
np.array_split(c4,2,axis=0)
```

Out[145…

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

In [146…

```
np.array_split(c4,4,axis=1)
```

```
Out[146…  [array([[1],
                  [5]]),
           array([[2],
                  [6]]),
           array([[3],
                  [7]]),
           array([[4],
                  [8]])]
```

```
In [147…  np.array_split(c4,3,axis=1)
```

```
Out[147…  [array([[1, 2],
                  [5, 6]]),
           array([[3],
                  [7]]),
           array([[4],
                  [8]])]
```

```
In [148…  np.array_split(c4,[2,3],axis=1)
```

```
Out[148…  [array([[1, 2],
                  [5, 6]]),
           array([[3],
                  [7]]),
           array([[4],
                  [8]])]
```

```
In [ ]:
```