NumPy

In [1]:

```
import numpy as np # import NumPy
D20=np.array([[1,6,4],[5,9,2],[11,6,8]])
D20
```

Out[1]:

```
array([[ 1,  6,  4],
       [ 5,  9,  2],
       [11,  6,  8]])
```

In [2]:

```
D20.shape # shape of Array
```

Out[2]:

```
(3, 3)
```

In [3]:

```
D20.size # total number of element
```

Out[3]:

```
9
```

In [4]:

```
D21=np.array([[50,90,60],[50,30,40],[80,90,70]])
D21
```

Out[4]:

```
array([[50, 90, 60],
       [50, 30, 40],
       [80, 90, 70]])
```

In [5]:

```
D22=np.array([[50,9000000,60],[50,300000,40],[80,9000000000,700000000]])
D22
```

Out[5]:

```
array([[        50,    9000000,         60],
       [        50,     300000,         40],
       [        80, 9000000000,  700000000]], dtype=int64)
```

In [6]:

```
print(D20.dtype) # space taken by array
```

```
int32
```

In [7]:

```
print(D21.dtype)
```

```
int32
```

In [8]:

```
print(D22.dtype)
```

```
int64
```

In [9]:

```
print(np.sum(D20)) # sum of all element of array
```

```
52
```

In [10]:

```
D20.sum(axis=0)
```

Out[10]:

```
array([17, 21, 14])
```

In [11]:

```
D20.sum(axis=1)
```

Out[11]:

```
array([11, 16, 25])
```

In [12]:

```
D21.sum(axis=0)
```

Out[12]:

```
array([180, 210, 170])
```

In [13]:

```
D21.sum(axis=1)
```

Out[13]:

```
array([200, 120, 240])
```

In [14]:

```
print(np.sum(D21))
```

```
560
```

In [15]:

```
np.ndim(D20) # Dimentation Of array
```

Out[15]:

```
2
```

In [16]:

```
np.ndim(D21)
```

Out[16]:

```
2
```

In [17]:

```
D10=np.array([8,7,5,8,6,2,1,6,9]) # create 1D Array
D10
```

Out[17]:

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

In [18]:

```
print(D10[2]) # Select index value 1D
```

```
5
```

In [19]:

```
print(D10[6])
```

```
1
```

In [20]:

```
D20
```

Out[20]:

```
array([[ 1,  6,  4],
       [ 5,  9,  2],
       [11,  6,  8]])
```

In [21]:

```
print(D20[2,1]) # select index value 2D
```

```
6
```

In [22]:

```
print(D20[0,2])
```

```
4
```

In [23]:

```
print(D21[0,2])
```

```
60
```

In [24]:

```
print(D20[0:2,0:2])
```

```
[[1 6]
 [5 9]]
```

In [25]:

```
print(D20[1:2,0:1])
```

```
[[5]]
```

In [26]:

```
print(D21[1:2,1:2])
```

```
[[30]]
```

In [27]:

```
print(D21[1:3,1:3])
```

```
[[30 40]
 [90 70]]
```

In [28]:

```
print(np.cumsum(D20)) # sum of number with last result
```

```
[ 1  7 11 16 25 27 38 44 52]
```

In [29]:

```
print(np.cumsum(D21))
```

```
[ 50 140 200 250 280 320 400 490 560]
```

In [30]:

```
print(np.cumsum(D10))
```

```
[ 8 15 20 28 34 36 37 43 52]
```

In [31]:

```
print(D20.T) # transpose of metriex 2D
```

```
[[ 1  5 11]
 [ 6  9  6]
 [ 4  2  8]]
```

In [32]:

```
print(D21.T)
```

```
[[50 50 80]
 [90 30 90]
 [60 40 70]]
```

In [33]:

```
print(D10.T)# transpose of metriex 1D
```

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

In [34]:

```
z=np.linalg.det(D20) # determinate of metriex 2D
z
```

Out[34]:

```
-324.00000000000017
```

In [35]:

```
z=np.linalg.det(D21)
z
```

Out[35]:

```
23999.999999999985
```

In [36]:

```
z1=np.linalg.inv(D20) # inverce of metriex
z1
```

Out[36]:

```
array([[-0.18518519,  0.07407407,  0.07407407],
       [ 0.05555556,  0.11111111, -0.05555556],
       [ 0.21296296, -0.18518519,  0.06481481]])
```

In [37]:

```
z2=np.linalg.inv(D21)
z2
```

Out[37]:

```
array([[-0.0625    , -0.0375    ,  0.075     ],
       [-0.0125    , -0.05416667,  0.04166667],
       [ 0.0875    ,  0.1125    , -0.125     ]])
```

In [38]:

```
r1=np.random.randint(10,size=(4,4)) # create Rendom 2D Array
r1
```

Out[38]:

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

In [39]:

```
r2=np.random.randint(50,80,size=(5,5))
r2
```

Out[39]:

```
array([[69, 65, 63, 79, 72],
       [62, 58, 78, 78, 57],
       [68, 69, 52, 72, 61],
       [62, 65, 65, 76, 66],
       [74, 79, 53, 76, 56]])
```

In [40]:

```
r3=np.random.randint(50,80,size=(10,5))
r3
```

Out[40]:

```
array([[52, 76, 66, 67, 79],
       [53, 64, 74, 62, 60],
       [52, 60, 51, 55, 52],
       [61, 59, 75, 70, 78],
       [71, 63, 57, 58, 72],
       [65, 78, 71, 57, 52],
       [64, 62, 70, 70, 56],
       [76, 61, 56, 67, 73],
       [52, 60, 78, 61, 56],
       [55, 78, 60, 57, 52]])
```

In [41]:

```
h1=np.hstack((D20,D21)) # Horizentel Merge
h1
```

Out[41]:

```
array([[ 1,  6,  4, 50, 90, 60],
       [ 5,  9,  2, 50, 30, 40],
       [11,  6,  8, 80, 90, 70]])
```

In [42]:

```
h2=np.hstack((D21,D20))
h2
```

Out[42]:

```
array([[50, 90, 60,  1,  6,  4],
       [50, 30, 40,  5,  9,  2],
       [80, 90, 70, 11,  6,  8]])
```

In [43]:

```
v1=np.vstack((D20,D21)) # Vertical Merge
v1
```

Out[43]:

```
array([[ 1,  6,  4],
       [ 5,  9,  2],
       [11,  6,  8],
       [50, 90, 60],
       [50, 30, 40],
       [80, 90, 70]])
```

In [44]:

```
v2=np.vstack((D21,D20)) # Vertical Merge
v2
```

Out[44]:

```
array([[50, 90, 60],
       [50, 30, 40],
       [80, 90, 70],
       [ 1,  6,  4],
       [ 5,  9,  2],
       [11,  6,  8]])
```

In [45]:

```
D20
```

Out[45]:

```
array([[ 1,  6,  4],
       [ 5,  9,  2],
       [11,  6,  8]])
```

In [46]:

```
print(D21[::-1]) # Inter Cahnge The Row
```

```
[[80 90 70]
 [50 30 40]
 [50 90 60]]
```

In [47]:

```
print(D20[::-1])
```

```
[[11  6  8]
 [ 5  9  2]
 [ 1  6  4]]
```

In [48]:

```
print(D20[::,::-1])# Inter Change of Column
```

```
[[ 4  6  1]
 [ 2  9  5]
 [ 8  6 11]]
```

In [49]:

```
print(D21[::,::-1])
```

```
[[60 90 50]
 [40 30 50]
 [70 90 80]]
```

In [50]:

```
for i1 in np.nditer(D20,order='C'): # Read ACcording Column
    print(i1)
```

```
1
6
4
5
9
2
11
6
8
```

In [51]:

```
for i2 in np.nditer(D21,order='C'):
    print(i2)
```

```
50
90
60
50
30
40
80
90
70
```

In [52]:

```
for i3 in np.nditer(D20,order='F'): # Read According Row
    print(i3)
```

```
1
5
11
6
9
6
4
2
8
```

In [53]:

```
for i4 in np.nditer(D21,order='F'):
    print(i4)
```

```
50
50
80
90
30
90
60
40
70
```

In [54]:

```
print(np.zeros((3,3))) # Create Zero Array
```

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

In [55]:

```
print(np.zeros((2,5)))
```

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

In [56]:

```
print(np.ones((4,4)))  # Create One Array
```

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

In [57]:

```
print(np.ones((13,10)))
```

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

In [58]:

```
np.any(D20>6) # Check the number Present
```

Out[58]:

```
True
```

In [59]:

```
np.any(D20<6)
```

Out[59]:

```
True
```

In [60]:

```
 np.any(D20>12)
```

Out[60]:

```
False
```

In [61]:

```
np.any(D21>60)
```

Out[61]:

```
True
```

In [62]:

```
np.any(D21<25)
```

Out[62]:

```
False
```

In [63]:

```
D20.max() # select Maximum or Minmum Number
```

Out[63]:

```
11
```

In [64]:

```
D20.min()
```

Out[64]:

```
1
```

In [65]:

```
D21.max()
```

Out[65]:

```
90
```

In [66]:

```
D21.min()
```

Out[66]:

```
30
```

In [67]:

```
l1=np.linspace(20,50,10) # Divide In equele Parts
l1
```

Out[67]:

```
array([20.        , 23.33333333, 26.66666667, 30.        , 33.33333333,
       36.66666667, 40.        , 43.33333333, 46.66666667, 50.        ])
```

In [68]:

```
l2=np.linspace(1,5,5)
l2
```

Out[68]:

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

In [69]:

```
emp=np.empty((3,2)) # get empty Array
emp
```

Out[69]:

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

In [70]:

```
ide=np.identity(3) # get Identity metrix
ide
```

Out[70]:

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

In [71]:

```
ide1=np.identity(10)
ide1
```

Out[71]:

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

In [72]:

```
ide1.shape
```

Out[72]:

```
(10, 10)
```

In [73]:

```
D10.reshape(3,3) # 1D to 2D Exchange
```

Out[73]:

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

In [74]:

```
D20.flatten() # 2D to 1D Exchabge
```

Out[74]:

```
array([ 1,  6,  4,  5,  9,  2, 11,  6,  8])
```

In [75]:

```
D20.ravel()
```

Out[75]:

```
array([ 1,  6,  4,  5,  9,  2, 11,  6,  8])
```

In [76]:

```
D20.nbytes # storage use by array
```

Out[76]:

```
36
```

In [77]:

```
D21.nbytes
```

Out[77]:

```
36
```

In [78]:

```
D20.argmax() # largest number Index
```

Out[78]:

```
6
```

In [79]:

```
D20.argmin() # smallest Number Index
```

Out[79]:

```
0
```

In [80]:

```
np.sort(D20) # shorting of Array
```

Out[80]:

```
array([[ 1,  4,  6],
       [ 2,  5,  9],
       [ 6,  8, 11]])
```

In [81]:

```
D20.argsort() # index shorting
```

Out[81]:

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

In [82]:

```
D20+D21 # Array Element Calculation
```

Out[82]:

```
array([[51, 96, 64],
       [55, 39, 42],
       [91, 96, 78]])
```

In [83]:

```
D20*D21
```

Out[83]:

```
array([[ 50, 540, 240],
       [250, 270,  80],
       [880, 540, 560]])
```

In [84]:

```
D20-D21
```

Out[84]:

```
array([[-49, -84, -56],
       [-45, -21, -38],
       [-69, -84, -62]])
```

In [85]:

```
D20/D21
```

Out[85]:

```
array([[0.02      , 0.06666667, 0.06666667],
       [0.1       , 0.3       , 0.05      ],
       [0.1375    , 0.06666667, 0.11428571]])
```

In [86]:

```
print(np.sqrt(D20)) # Squre Root OF metrix
```

```
[[1.         2.44948974 2.        ]
 [2.23606798 3.         1.41421356]
 [3.31662479 2.44948974 2.82842712]]
```

In [87]:

```
print(np.sqrt(D21))
```

```
[[7.07106781 9.48683298 7.74596669]
 [7.07106781 5.47722558 6.32455532]
 [8.94427191 9.48683298 8.36660027]]
```

In [88]:

```
print(np.sqrt(D10))
```

```
[2.82842712 2.64575131 2.23606798 2.82842712 2.44948974 1.41421356
 1.         2.44948974 3.        ]
```

In [89]:

```
print(np.mean(D20)) # Mean Value of metrix
```

```
5.777777777777778
```

In [90]:

```
print(np.mean(D21))
```

```
62.22222222222222
```

In [91]:

```
print(np.mean(D10))
```

```
5.777777777777778
```

In [92]:

```
print(np.std(D21))
```

```
20.42752923427804
```

In [93]:

```
print(np.std(D20))
```

```
3.046957600178242
```

In [94]:

```
np.where(D20==11)
```

Out[94]:

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

In [95]:

```
np.where(D20<11)
```

Out[95]:

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

In [96]:

```
np.count_nonzero(D20)
```

Out[96]:

```
9
```

In [97]:

```
np.nonzero(D20)
```

Out[97]:

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

In [98]:

```
D20[1,2]=0 # update value of array element
D20
```

Out[98]:

```
array([[ 1,  6,  4],
       [ 5,  9,  0],
       [11,  6,  8]])
```

In [99]:

```
np.count_nonzero(D20)
```

Out[99]:

```
8
```

In [100]:

```
np.nonzero(D20)
```

Out[100]:

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

In [101]:

```
x1=D20[:2,:2].copy() # copy Of particuler array
x1
```

Out[101]:

```
array([[1, 6],
       [5, 9]])
```

In [102]:

```
x2=D21[:2,:2].copy() # copy Of particuler array
x2
```

Out[102]:

```
array([[50, 90],
       [50, 30]])
```

In [103]:

```
len(D21)
```

Out[103]:

```
3
```

```
LINK for NumPy Array methods and attributes
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.html
```