```
## Numpy .,mntr`
```

- Numpy intorduction
- Array Creation -> 1D,2D,3D
- Attributes->ndim,size,shape,dtype
- Functions -> arange,linspace,ravel,random,rand
- stacking -> hstack,vstack
- splitting-> hspllit,vsplit
- Matrik Operations
- boaedcasting ,masking
- other functions->eye,zeros,ones,max,min ,cpoy,transpose,fliplr

numpy -> Numerical Python

- homogeneous miltidimensional collection of elements
- scientific and mathematicalcomputations

why use Numpy

- Numpy arrays are faster and more compact than python lists
- Numpy uses much less memory to store data

In [1]:

```
import numpy as np
```

In [2]:

```
# creation of Array
#1D
arr1 = np.array([3,4,5])
arr1
```

Out[2]:

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

In [5]:

```
#Attritubtes
```

In [6]:

```
print('Dimension',arr1.ndim)
print('Size',arr1.size)
print('Shape',arr1.shape)
print('DataType',arr1.dtype)
```

```
Dimension 1
Size 3
Shape (3,)
DataType int32
```

In [20]:

```
# 2d array
```

In [4]:

```
arr2 = np.array([[4,5,6],[1,2,3]])
arr2
```

Out[4]:

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

In [5]:

```
print('size',arr2.size)
```

```
size 6
```

In [6]:

```
print('Dimension',arr2.ndim)
print('Size',arr2.size)
print('Shape',arr2.shape)
print('DataType',arr2.dtype)
```

```
Dimension 2
Size 6
Shape (2, 3)
DataType int32
```

In [7]:

```
arr3 = np.array([[4,7,9,2],[1,2,3,4],[9,8,7,6]])
arr3
```

Out[7]:

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

In [17]:

```
print('Dimension',arr3.ndim)
print('Size',arr3.size)
print('Shape',arr3.shape)
print('DataType',arr3.dtype)
```

```
Dimension 2
Size 12
Shape (3, 4)
DataType int32
```

In [47]:

```
# create an array with shape(4,2)
arr4 = np.array([[2,3],[4,5],[6,7],[8,9]])
arr4
```

Out[47]:

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

In [19]:

```
print('Dimension',arr4.ndim)
print('Size',arr4.size)
print('Shape',arr4.shape)
print('DataType',arr4.dtype)
```

```
Dimension 2
Size 8
Shape (4, 2)
DataType int32
```

In [21]:

```
# 3D array
```

In [9]:

```
arr5 = np.array([[[2,3],[4,5],[6,7],[8,9]],[[2,3],[4,5],[6,7],[8,9]],[[2,3],[4,5],[6,7],[8,9]]])
arr5
```

Out[9]:

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

       [[2, 3],
        [4, 5],
        [6, 7],
        [8, 9]],

       [[2, 3],
        [4, 5],
        [6, 7],
        [8, 9]]])
```

In [27]:

```
print('Dimension',arr5.ndim)
print('Size',arr5.size)
print('Shape',arr5.shape)
print('DataType',arr5.dtype)
```

```
Dimension 3
Size 24
Shape (3, 4, 2)
DataType int32
```

functions -> arrage,linspace,ravel,reshape,random,rand

In [32]:

```
#arange-> start,stop,(e),interval/step
np.arange(10) # stop(E)
```

Out[32]:

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

In [33]:

```
np.arange(50,40,-1)
```

Out[33]:

```
array([50, 49, 48, 47, 46, 45, 44, 43, 42, 41])
```

In [34]:

```
#linspace-> equally spaced numbers within the limt ->start,stop,count
(np.linspace(1,100,10)).astype('int')
```

Out[34]:

```
array([  1,  12,  23,  34,  45,  56,  67,  78,  89, 100])
```

In [11]:

```
np.linspace(1,100,9,retstep=True)
```

Out[11]:

```
(array([  1.   ,  13.375,  25.75 ,  38.125,  50.5  ,  62.875,  75.25 ,
         87.625, 100.   ]),
 12.375)
```

In [39]:

```
np.linspace(1,50,5,retstep=True)
```

Out[39]:

```
(array([ 1.  , 13.25, 25.5 , 37.75, 50.  ]), 12.25)
```

In [40]:

```
np.linspace(1,50,5,retstep=False)
```

Out[40]:
```
array([ 1.  , 13.25, 25.5 , 37.75, 50.  ])
```

In [41]:
```
np.linspace(1,100)
```

Out[41]:
```
array([  1.        ,   3.02040816,   5.04081633,   7.06122449,
         9.08163265,  11.10204082,  13.12244898,  15.14285714,
        17.16326531,  19.18367347,  21.20408163,  23.2244898 ,
        25.24489796,  27.26530612,  29.28571429,  31.30612245,
        33.32653061,  35.34693878,  37.36734694,  39.3877551 ,
        41.40816327,  43.42857143,  45.44897959,  47.46938776,
        49.48979592,  51.51020408,  53.53061224,  55.55102041,
        57.57142857,  59.59183673,  61.6122449 ,  63.63265306,
        65.65306122,  67.67346939,  69.69387755,  71.71428571,
        73.73469388,  75.75510204,  77.7755102 ,  79.79591837,
        81.81632653,  83.83673469,  85.85714286,  87.87755102,
        89.89795918,  91.91836735,  93.93877551,  95.95918367,
        97.97959184, 100.        ])
```

In [ ]:
```

```

In [42]:
```
np.linspace(1,100 ,retstep=True)
```

Out[42]:
```
(array([  1.        ,   3.02040816,   5.04081633,   7.06122449,
          9.08163265,  11.10204082,  13.12244898,  15.14285714,
         17.16326531,  19.18367347,  21.20408163,  23.2244898 ,
         25.24489796,  27.26530612,  29.28571429,  31.30612245,
         33.32653061,  35.34693878,  37.36734694,  39.3877551 ,
         41.40816327,  43.42857143,  45.44897959,  47.46938776,
         49.48979592,  51.51020408,  53.53061224,  55.55102041,
         57.57142857,  59.59183673,  61.6122449 ,  63.63265306,
         65.65306122,  67.67346939,  69.69387755,  71.71428571,
         73.73469388,  75.75510204,  77.7755102 ,  79.79591837,
         81.81632653,  83.83673469,  85.85714286,  87.87755102,
         89.89795918,  91.91836735,  93.93877551,  95.95918367,
         97.97959184, 100.        ]),
 2.020408163265306)
```

ravel-

coverts ND array to 1D array

In [10]:
```
arr5.ravel()
```

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

In [50]:
```
arr5.reshape(2,3,4)
```

Out[50]:
```
array([[[2, 3, 4, 5],
        [6, 7, 8, 9],
        [2, 3, 4, 5]],

       [[6, 7, 8, 9],
        [2, 3, 4, 5],
        [6, 7, 8, 9]]])
```

In [51]:
```
arr5.reshape(8,3)
```

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

In [8]:
```
arr5.reshape(8,3)
```

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

In [9]:
```
#random-> nos .b/w 0 to 1
```

In [11]:
```
np.random.random(10)#stop(E)
```

Out[11]:
```
array([0.87082321, 0.75864823, 0.06598673, 0.1585546 , 0.51708344,
       0.6961007 , 0.22906696, 0.98906724, 0.19314418, 0.35775023])
```

In [12]:
```
np.random.randint(1,15) # start,stop(E)
```

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

In [16]:
```
np.random.randint(1,15,3)# start stop(E),count
```

Out[16]:
```
array([12,  3,  4])
```

In [17]:
```
np.random.rand(2,3)#generate random array of given shape
```

Out[17]:
```
array([[0.49041132, 0.68746554, 0.53349168],
       [0.1020317 , 0.35476406, 0.50409949]])
```

In [19]:
```
np.random.randint(1,1000,10).reshape(2,5)
```

Out[19]:
```
array([[128, 169, 305, 306, 394],
       [834, 425, 788, 766, 931]])
```

In [20]:
```
np.random.randint(10000,999999,10).reshape(2,5)
```

Out[20]:
```
array([[351907, 969315, 202220,  86121, 542381],
       [487320, 857373, 296431, 157163,  70908]])
```

In [44]:
```
#Q) Generate  10 random nos.and convert into 6 digit int and reshape it into 2,5
(np.random.random(10)*1000000).astype(int)
```

Out[44]:
```
array([895739, 196142, 618672, 606073, 991176, 430143,  24279,  66250,
       549406, 819322])
```

In [21]:
```
#stacking-> hstack,vstack,column_stack
```

In [40]:
```
n1 = np.array([10,20,30])
n2 = np.array([40,50,60])

np.hstack((n1,n2))
```

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

In [41]:
```
np.vstack((n1,n2))
```

Out[41]:
```
array([[10, 20, 30],
       [40, 50, 60]])
```

In [43]:
```
np.column_stack((n1,n2))
```

Out[43]:
```
array([[10, 40],
       [20, 50],
       [30, 60]])
```

In [29]:
```
#splitting ->hsplit,vsplit
arr4
```

Out[29]:
```
array([[2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

In [32]:
```
np.hsplit(arr4,2) #rows
```

Out[32]:
```
[array([[2],
        [4],
        [6],
        [8]]),
 array([[3],
        [5],
        [7],
        [9]])]
```

In [34]:
```
np.vsplit(arr4,2)
```

Out[34]:
```
[array([[2, 3],
        [4, 5]]),
 array([[6, 7],
        [8, 9]])]
```

In [40]:
```
a1 =  np.random.randint(0,20,9).reshape(3,3)
print(a1)
np.hsplit(a1,3)
```

```
[[19 16 13]
 [ 2 13 18]
 [16  9  0]]
```

Out[40]:
```
[array([[19],
        [ 2],
        [16]]),
 array([[16],
        [13],
        [ 9]]),
 array([[13],
        [18],
        [ 0]])]
```

In [41]:
```
np.vsplit(a1,3)
```

Out[41]:
```
[array([[19, 16, 13]]), array([[ 2, 13, 18]]), array([[16,  9,  0]])]
```

matrix operations

In [44]:
```
# add,subb,multiply
```

In [4]:
```
a1 = np.arange(1,11).reshape(5,2)
a1
```

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

In [6]:
```
a2 = np.arange(21,31).reshape(5,2)
a2
```

Out[6]:
```
array([[21, 22],
       [23, 24],
       [25, 26],
       [27, 28],
       [29, 30]])
```

In [7]:
```
np.add(a1,a2)
```

Out[7]:
```
array([[22, 24],
       [26, 28],
       [30, 32],
       [34, 36],
       [38, 40]])
```

In [9]:
```
np.subtract(a2,a1)
```

Out[9]:
```
array([[20, 20],
       [20, 20],
       [20, 20],
       [20, 20],
       [20, 20]])
```

In [10]:
```
np.multiply(a1,a2)
```

Out[10]:
```
array([[ 21,  44],
       [ 69,  96],
       [125, 156],
       [189, 224],
       [261, 300]])
```

unsupported cell Type.Double-click to inspect/edit the content

In [11]:
```
np.random.random(12).reshape(4,3)
```

Out[11]:
```
array([[0.76152239, 0.27482111, 0.00375749],
       [0.38841687, 0.80658772, 0.71906041],
       [0.75180558, 0.16118557, 0.79413961],
       [0.93803762, 0.70720407, 0.04104104]])
```

In [12]:
```
a = np.random.randint(1,15,12).reshape(4,3)
a
```

Out[12]:
```
array([[ 9,  7, 10],
       [13,  2,  3],
       [12, 13, 10],
       [12, 11, 14]])
```

In [13]:
```
# [start: stop :step, start : stop : step ]
a[1:3]
```

Out[13]:
```
array([[13,  2,  3],
       [12, 13, 10]])
```

In [16]:
```
a[:,1:2]
```

Out[16]:
```
array([[ 7],
       [ 2],
       [13],
       [11]])
```

In [17]:
```
a[2:3]
```

Out[17]:
```
array([[12, 13, 10]])
```

In [18]:
```
a[1:3,1:]
```

Out[18]:
```
array([[ 2,  3],
       [13, 10]])
```

In [19]:
```
a[1:3,0:2]
```

Out[19]:
```
array([[13,  2],
       [12, 13]])
```

Broadcasting and Boolean Masking

In [20]:
```
a
```

Out[20]:
```
array([[ 9,  7, 10],
       [13,  2,  3],
       [12, 13, 10],
       [12, 11, 14]])
```

In [21]:
```
a[:,:] = 500
a
```

Out[21]:
```
array([[500, 500, 500],
       [500, 500, 500],
       [500, 500, 500],
       [500, 500, 500]])
```

In [22]:
```
ab = np.random.randint(1,15,12).reshape(4,3)
ab
```

Out[22]:
```
array([[ 2, 10,  2],
       [ 6, 13, 13],
       [11,  9,  3],
       [ 7,  1,  6]])
```

In [23]:
```
ab[1:3] = 100
ab
```

Out[23]:
```
array([[  2,  10,   2],
       [100, 100, 100],
       [100, 100, 100],
       [  7,   1,   6]])
```

In [ ]:
```
# Q) create an array from 11 to 30 reshape as 4,5 and broadcast 3 and 4th column as 500
```

In [24]:
```
## Boolean Masking

ac = np.random.randint(1,15,12).reshape(4,3)
ac
```

Out[24]:
```
array([[12,  3,  8],
       [ 6, 12,  3],
       [14, 10, 10],
       [ 9,  8, 11]])
```

In [25]:
```
ac%5 ==0
```

Out[25]:
```
array([[False, False, False],
       [False, False, False],
       [False,  True,  True],
       [False, False, False]])
```

In [26]:
```
ac > 5
```

Out[26]:
```
array([[ True, False,  True],
       [ True,  True, False],
       [ True,  True,  True],
       [ True,  True,  True]])
```

In [27]:
```
ac[1:2,:1]%5 == 0
```

Out[27]:
```
array([[False]])
```

other functions->eye,Zeros,ones,max,min,copy,transpose,fliplr

In [28]:
```
np.eye(3,dtype= 'int')
```

Out[28]:
```
array([[1, 0, 0],
       [0, 1, 0],
       [0, 0, 1]])
```

In [29]:
```
np.zeros((2,3))
```

Out[29]:
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [30]:
```
np.ones((2,4))
```

Out[30]:
```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

In [31]:
```
ac
```

Out[31]:
```
array([[12,  3,  8],
       [ 6, 12,  3],
       [14, 10, 10],
       [ 9,  8, 11]])
```

In [33]:
```
np.max(ac)
```

Out[33]:
```
14
```

In [34]:
```
np.min(ac)
```

Out[34]:
```
3
```

In [35]:
```
ad = np.copy(ac)
ad
```

Out[35]:
```
array([[12,  3,  8],
       [ 6, 12,  3],
       [14, 10, 10],
       [ 9,  8, 11]])
```

In [36]:
```
np.transpose(ac)
```

Out[36]:
```
array([[12,  6, 14,  9],
       [ 3, 12, 10,  8],
       [ 8,  3, 10, 11]])
```

In [37]:
```
np.fliplr(ac)
```

Out[37]:
```
array([[ 8,  3, 12],
       [ 3, 12,  6],
       [10, 10, 14],
       [11,  8,  9]])
```

In [38]:
```
ac
```

Out[38]:
```
array([[12,  3,  8],
       [ 6, 12,  3],
       [14, 10, 10],
       [ 9,  8, 11]])
```

In [39]:
```
ac.astype('float')
```

Out[39]:
```
array([[12.,  3.,  8.],
       [ 6., 12.,  3.],
       [14., 10., 10.],
       [ 9.,  8., 11.]])
```

questions

In [45]:
```
# Q1) create an array with multiples of 2 from 2 to 20
print(np.arange(2,21,2))
```

```
[ 2  4  6  8 10 12 14 16 18 20]
```

In [48]:
```
# Q2) convert a 3D array into 1D array
arr4.ravel()
```

Out[48]:
```
array([2, 3, 4, 5, 6, 7, 8, 9])
```

In [49]:
```
# Q) create an array from  51 to 100 and the shape is 5,2,5
np.arange(51,101).reshape(5,2,5)
```

Out[49]:
```
array([[[ 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]]])
```

In [50]:
```
#Q) Generate an array 50 to 41 by using arange function
np.arange(50,40,-1)
```

Out[50]:
```
array([50, 49, 48, 47, 46, 45, 44, 43, 42, 41])
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js