In [1]:

```
import numpy as np
```

In [2]:

```
l=[1,2,3,5]
```

In [3]:

```
np_a=np.array(l)
```

In [4]:

```
np_a
```

Out[4]:

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

In [5]:

```
np_a.shape
```

Out[5]:

```
(4,)
```

In [6]:

```
np_a.size
```

Out[6]:

```
4
```

In [7]:

```
np.arange(10) #to get np arry for the given range
```

Out[7]:

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

In [11]:

```
np_a2=np.arange(start=10,step=3,stop=34,dtype=np.float64) #To get for given condtions
np_a2
```

Out[11]:

```
array([10., 13., 16., 19., 22., 25., 28., 31.])
```

In [12]:

```
np_a2.shape
```

Out[12]:

```
(8,)
```

In [13]:

```
np_a2.reshape(2,4) ## re shape array
```

Out[13]:

```
array([[10., 13., 16., 19.],
       [22., 25., 28., 31.]])
```

In [14]:

```
np_a2.reshape(-500,4)
```

Out[14]:

```
array([[10., 13., 16., 19.],
       [22., 25., 28., 31.]])
```

In [15]:

```
np_a2.reshape(4,-1000)
```

Out[15]:

```
array([[10., 13.],
       [16., 19.],
       [22., 25.],
       [28., 31.]])
```

In [16]:

```
np.zeros((3,5))
```

Out[16]:

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

In [17]:

```
d3=np.ones((2,3,4),dtype=np.int16)
d3
```

Out[17]:

```
array([[[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]],

       [[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]]], dtype=int16)
```

In [18]:

```
d3.shape
```

Out[18]:

```
(2, 3, 4)
```

In [19]:

```
np.empty((2,2)) #gives Random numbers
```

Out[19]:

```
array([[6.79973727e-312, 0.00000000e+000],
       [            nan, 2.86558075e-322]])
```

In [20]:

```
#eye() the function eye() generate identiy matrix
np.eye((5))
```

Out[20]:

```
array([[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.]])
```

In [22]:

```
#linspace() this method return evenly spaced numbers over a specified interval
np.linspace(2.0,3.0,num=10,endpoint=True)
```

Out[22]:

```
array([2.        , 2.11111111, 2.22222222, 2.33333333, 2.44444444,
       2.55555556, 2.66666667, 2.77777778, 2.88888889, 3.        ])
```

In [23]:

```
np.linspace(2.0,3.0,num=30,endpoint=False,retstep=True)
```

Out[23]:

```
(array([2.        , 2.03333333, 2.06666667, 2.1       , 2.13333333,
        2.16666667, 2.2       , 2.23333333, 2.26666667, 2.3       ,
        2.33333333, 2.36666667, 2.4       , 2.43333333, 2.46666667,
        2.5       , 2.53333333, 2.56666667, 2.6       , 2.63333333,
        2.66666667, 2.7       , 2.73333333, 2.76666667, 2.8       ,
        2.83333333, 2.86666667, 2.9       , 2.93333333, 2.96666667]),
 0.03333333333333333)
```

In [24]:

```
type(np.linspace(2.0,3.0,num=30,endpoint=False)) #To find type
```

Out[24]:

```
numpy.ndarray
```

Basic Operations

In [25]:

```
a=np.array([10,20,30,40])
b=np.arange(4)
```

In [26]:

```
print(a,b)
```

```
[10 20 30 40] [0 1 2 3]
```

In [27]:

```
c=a-b
c
```

Out[27]:

```
array([10, 19, 28, 37])
```

In [28]:

```
10*a
```

Out[28]:

```
array([100, 200, 300, 400])
```

In [29]:

```
a+10
```

Out[29]:

```
array([20, 30, 40, 50])
```

In [30]:

```
f=np.array([a,b])
f
```

Out[30]:

```
array([[10, 20, 30, 40],
       [ 0,  1,  2,  3]])
```

In [31]:

```
f.shape
```

Out[31]:

```
(2, 4)
```

In [32]:

```
f*a
```

Out[32]:

```
array([[ 100,  400,  900, 1600],
       [   0,   20,   60,  120]])
```

In [33]:

```
f+=10
f
```

Out[33]:

```
array([[20, 30, 40, 50],
       [10, 11, 12, 13]])
```

In [34]:

```
e=np.eye(4)
```

In [35]:

```
e
```

Out[35]:

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

In [36]:

```
e[0,0]=5
e[1,1]=10
e[2,2]=20
e[3,3]=30
```

In [37]:

```
e
```

Out[37]:

```
array([[ 5.,  0.,  0.,  0.],
       [ 0., 10.,  0.,  0.],
       [ 0.,  0., 20.,  0.],
       [ 0.,  0.,  0., 30.]])
```

In [38]:

```
e[2,1]
```

Out[38]:

```
0.0
```

In [39]:

```
10*np.sin(a)
```

Out[39]:

```
array([-5.44021111,  9.12945251, -9.88031624,  7.4511316 ])
```

In [40]:

```
a<35
```

Out[40]:

```
array([ True,  True,  True, False])
```

In [41]:

```
b=np.random.random((2,3))
b
```

Out[41]:

```
array([[0.27951738, 0.89256953, 0.49672197],
       [0.60999874, 0.80095198, 0.91546422]])
```

In [42]:

```
b
```

Out[42]:

```
array([[0.27951738, 0.89256953, 0.49672197],
       [0.60999874, 0.80095198, 0.91546422]])
```

In [43]:

```
b=np.arange(12)
```

In [44]:

```
b=b.reshape(3,4)
b
```

Out[44]:

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

In [45]:

```
b.sum()
```

Out[45]:

```
66
```

In [46]:

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

Out[46]:

```
array([12, 15, 18, 21])
```

In [47]:

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

Out[47]:

```
array([ 6, 22, 38])
```

In [48]:

```
b.sum(axis=1,keepdims=True)
```

Out[48]:

```
array([[ 6],
       [22],
       [38]])
```

In [49]:

```
b.dtype
```

Out[49]:

```
dtype('int32')
```

In [50]:

```
#Cumilative sum
b.cumsum(axis=1)
```

Out[50]:

```
array([[ 0,  1,  3,  6],
       [ 4,  9, 15, 22],
       [ 8, 17, 27, 38]], dtype=int32)
```

In [51]:

```
b
```

Out[51]:

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

In [52]:

```
b.max()
```

Out[52]:

```
11
```

In [53]:

```
b.min()
```

Out[53]:

```
0
```

In [54]:

```
b.max(axis=1)
```

Out[54]:

```
array([ 3,  7, 11])
```

In [55]:

```
b.mean()
```

Out[55]:

```
5.5
```

In [56]:

```
b.mean(axis=1)
```

Out[56]:

```
array([1.5, 5.5, 9.5])
```

In [57]:

```
b.mean(axis=0)
```

Out[57]:

```
array([4., 5., 6., 7.])
```

In [58]:

```
b.argmin()
```

Out[58]:

```
0
```

In [59]:

```
b.argmax()
```

Out[59]:

```
11
```

In [60]:

```
c=np.arange(3)
```

In [61]:

```
c
```

Out[61]:

```
array([0, 1, 2])
```

In [62]:

```
np.exp(c)
```

Out[62]:

```
array([1.        , 2.71828183, 7.3890561 ])
```

In [63]:

```
e=np.arange(10)**3
e
```

Out[63]:

```
array([  0,   1,   8,  27,  64, 125, 216, 343, 512, 729], dtype=int32)
```

In [64]:

```
type(e)
```

Out[64]:

```
numpy.ndarray
```

In [65]:

```
e[2]
```

Out[65]:

```
8
```

In [66]:

```
e[[5,3,7]]
```

Out[66]:

```
array([125,  27, 343], dtype=int32)
```

In [67]:

```
e[2:5]
```

Out[67]:

```
array([ 8, 27, 64], dtype=int32)
```

In [68]:

```
e[0:-4]
```

Out[68]:

```
array([  0,   1,   8,  27,  64, 125], dtype=int32)
```

In [69]:

```
e
```

Out[69]:

```
array([  0,   1,   8,  27,  64, 125, 216, 343, 512, 729], dtype=int32)
```

In [70]:

```
e[:]
```

Out[70]:

```
array([  0,   1,   8,  27,  64, 125, 216, 343, 512, 729], dtype=int32)
```

In [71]:

```
e[0:6:2]
```

Out[71]:

```
array([ 0,  8, 64], dtype=int32)
```

In [72]:

```
e[0:6:2]=-20
```

In [73]:

```
e
```

Out[73]:

```
array([-20,   1, -20,  27, -20, 125, 216, 343, 512, 729], dtype=int32)
```

In [74]:

```
e[::3]
```

Out[74]:

```
array([-20,  27, 216, 729], dtype=int32)
```

In [75]:

```
e[::-1]
```

Out[75]:

```
array([729, 512, 343, 216, 125, -20,  27, -20,   1, -20], dtype=int32)
```

In [76]:

```
b
```

Out[76]:

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

In [77]:

```
x=b[[1,2],1:3]
```

In [78]:

```
x
```

Out[78]:

```
array([[ 5,  6],
       [ 9, 10]])
```

In [79]:

```
x=b[[1,2]]
```

In [81]:

```
x
```

Out[81]:

```
array([[ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [83]:

```
y=b[[1,2],1:3]
y
```

Out[83]:

```
array([[ 5,  6],
       [ 9, 10]])
```

In [84]:

```
def f(x,y):
    return 3*x+2*y
g=np.fromfunction(f,(2,3),dtype=int) #position based fill up (x,y)=(0,0),(0,1)..
g
```

Out[84]:

```
array([[0, 2, 4],
       [3, 5, 7]])
```

In [85]:

```
g.shape
```

Out[85]:

```
(2, 3)
```

In [86]:

```
g=np.fromfunction(lambda x,y:3*x+2*y,(4,5),dtype=int)
g
```

Out[86]:

```
array([[ 0,  2,  4,  6,  8],
       [ 3,  5,  7,  9, 11],
       [ 6,  8, 10, 12, 14],
       [ 9, 11, 13, 15, 17]])
```

In [87]:

```
for x in g:
    print(x)
```

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

In [89]:

```
np.max(g)
```

Out[89]:

```
17
```

In [90]:

```
for row in g:
    for i in row:
        print(i)
```

```
0
2
4
6
8
3
5
7
9
11
6
8
10
12
14
9
11
13
15
17
```

In [91]:

```
np.sort(g)
```

Out[91]:

```
array([[ 0,  2,  4,  6,  8],
       [ 3,  5,  7,  9, 11],
       [ 6,  8, 10, 12, 14],
       [ 9, 11, 13, 15, 17]])
```

In [93]:

```
-g
```

Out[93]:

```
array([[  0,  -2,  -4,  -6,  -8],
       [ -3,  -5,  -7,  -9, -11],
       [ -6,  -8, -10, -12, -14],
       [ -9, -11, -13, -15, -17]])
```

In [94]:

```
g.T
```

Out[94]:

```
array([[ 0,  3,  6,  9],
       [ 2,  5,  8, 11],
       [ 4,  7, 10, 13],
       [ 6,  9, 12, 15],
       [ 8, 11, 14, 17]])
```

In [97]:

```
g.resize(4,5)
g
```

Out[97]:

```
array([[ 0,  2,  4,  6,  8],
       [ 3,  5,  7,  9, 11],
       [ 6,  8, 10, 12, 14],
       [ 9, 11, 13, 15, 17]])
```

In [99]:

```
k=np.array([[8,8],[20,15]])
k
l=np.array([[2,4],[3,9]])
l
u=np.array([[9,7],[6,5]])
u
```

Out[99]:

```
array([[9, 7],
       [6, 5]])
```

In [100]:

```
np.vstack((k,l,u))
```

Out[100]:

```
array([[ 8,  8],
       [20, 15],
       [ 2,  4],
       [ 3,  9],
       [ 9,  7],
       [ 6,  5]])
```

In [101]:

```
np.hstack((k,l,u))
```

Out[101]:

```
array([[ 8,  8,  2,  4,  9,  7],
       [20, 15,  3,  9,  6,  5]])
```

In [103]:

```
countries=np.array(['india','usa','germany','mexico','germany','russia'])
countries
```

Out[103]:

```
array(['india', 'usa', 'germany', 'mexico', 'germany', 'russia'],
      dtype='<U7')
```

In [106]:

```
np.unique(countries)
```

Out[106]:

```
array(['germany', 'india', 'mexico', 'russia', 'usa'], dtype='<U7')
```

In [109]:

```
np.in1d((['france','usa','india']),countries)
```

Out[109]:

```
array([False,  True,  True])
```

In [111]:

```
junk_data=np.array(['india',20,True,'30.33'])
junk_data
```

Out[111]:

```
array(['india', '20', 'True', '30.33'], dtype='<U5')
```

In [ ]:

```

```