Basic numpy operations

In [3]:

```
import numpy as np
```

In [2]:

```
l = [1,2,3,4,5,6]
```

In [3]:

```
type(l)
```

Out[3]:

```
list
```

In [5]:

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

In [6]:

```
type(a)
```

Out[6]:

```
numpy.ndarray
```

In [7]:

```
a
```

Out[7]:

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

In [8]:

```
a1 = np.array([[1,2,3] , [3,4,5]])
```

In [9]:

```
a1
```

Out[9]:

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

In [10]:

```
a2 = np.array([[[1,2,3] , [4,.5,6]]])
```

In [11]:

```
a2
```

Out[11]:

```
array([[[1. , 2. , 3. ],
        [4. , 0.5, 6. ]]])
```

In [12]:

```
a
```

Out[12]:

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

In [13]:

```
a1
```

Out[13]:

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

In [14]:

```
a2
```

Out[14]:

```
array([[[1. , 2. , 3. ],
        [4. , 0.5, 6. ]]])
```

In [15]:

```
a.ndim
```

Out[15]:

```
1
```

In [16]:

```
a1.ndim
```

Out[16]:

```
2
```

In [17]:

```
a2.ndim
```

Out[17]:

```
3
```

In [18]:

```
a3 = np.array([[[1,2,3] , [4,.5,6]],[[1,2,3] , [4,.5,6]],[[1,2,3] , [4,.5,6]]])
```

In [19]:

```
a3
```

Out[19]:

```
array([[[1. , 2. , 3. ],
        [4. , 0.5, 6. ]],

       [[1. , 2. , 3. ],
        [4. , 0.5, 6. ]],

       [[1. , 2. , 3. ],
        [4. , 0.5, 6. ]]])
```

In [20]:

```
a3.ndim
```

Out[20]:

```
3
```

In [21]:

```
l = [1,2,3,4]
```

In [23]:

```
# array will have atleast 0 (or) 1 dimensions

a = np.asarray(l)
```

In [25]:

```
# Matrix should have atleast 2 dimensions

m = np.matrix(l)
```

In [26]:

```
# If we pass the list into this function it will automatically converts into the array.

np.asanyarray(l)
```

Out[26]:

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

In [27]:

```
# If we pass an array means it will keep as it is...

np.asanyarray(a)
```

Out[27]:

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

In [28]:

```
# If we pass matrix also it will keep as it is nochanges

np.asanyarray(m)
```

Out[28]:

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

In [29]:

```
a
```

Out[29]:

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

In [30]:

```
a1 = a
```

In [31]:

```
a1
```

Out[31]:

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

In [32]:

```
a
```

Out[32]:

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

In [33]:

```
a[0]
```

Out[33]:

```
1
```

In [34]:

```
# i am reassigning the value

a[0] = 10
```

In [35]:

```
a
```

Out[35]:

```
array([10,  2,  3,  4])
```

In [36]:

```
a1
```

Out[36]:

```
array([10,  2,  3,  4])
```

In [37]:

```
a1[3] = 50
```

In [38]:

```
a1
```

Out[38]:

```
array([10,  2,  3, 50])
```

In [39]:

```
a
```

Out[39]:

```
array([10,  2,  3, 50])
```

In [40]:

```
# 1) By using this copy function it will reflect any changes in 'a'. this is called as d
# 2) It will just copy the data from 'a' and then what ever the changes we do in 'a2' wi
#       in 'a'.

a2 = np.copy(a)
```

In [41]:

```
a2
```

Out[41]:

```
array([10,  2,  3, 50])
```

In [42]:

```
a2[2] = 30
```

In [43]:

```
a2
```

Out[43]:

```
array([10,  2, 30, 50])
```

In [44]:

```
a
```

Out[44]:

```
array([10,  2,  3, 50])
```

In [45]:

```
np.fromfunction(lambda i,j : i==j, (3,4))
```

Out[45]:

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

In [46]:

```
# It will create a random dataset based on the given dimensions.

np.random.rand(2,3,2)
```

Out[46]:

```
array([[[0.56771449, 0.53303746],
        [0.01856296, 0.11779739],
        [0.26107495, 0.18596443]],

       [[0.30285662, 0.57393164],
        [0.0470844 , 0.04477806],
        [0.61124887, 0.68137033]]])
```

In [47]:

```
np.random.rand(2,3,2,2)
```

Out[47]:

```
array([[[[0.26372678, 0.47165035],
         [0.50479555, 0.26090485]],

        [[0.45086957, 0.9562422 ],
         [0.84367176, 0.01693441]],

        [[0.87072555, 0.51671993],
         [0.45152065, 0.2702615 ]]],


       [[[0.71230577, 0.71887381],
         [0.37683494, 0.47644885]],

        [[0.02898636, 0.64569581],
         [0.89925595, 0.25327072]],

        [[0.47653614, 0.9849151 ],
         [0.6153431 , 0.22804771]]]])
```

In [48]:

```
np.random.rand(1,4,4)
```

Out[48]:

```
array([[[0.73678356, 0.97487006, 0.27164748, 0.00923442],
        [0.11797212, 0.15425921, 0.71654653, 0.05554336],
        [0.7311371 , 0.86954941, 0.36108718, 0.7439734 ],
        [0.43315169, 0.34214684, 0.85885389, 0.33866309]]])
```

In [50]:

```
np.random.rand(2,4,4)
```

Out[50]:

```
array([[[0.71727827, 0.11165903, 0.88000271, 0.18004964],
        [0.60618678, 0.63848832, 0.52574878, 0.36507022],
        [0.99916362, 0.53662626, 0.91718647, 0.344858  ],
        [0.63037934, 0.4928798 , 0.86526033, 0.60009306]],

       [[0.69823222, 0.87538781, 0.74430076, 0.72578646],
        [0.56886909, 0.80965831, 0.81271994, 0.43286542],
        [0.84554105, 0.60902897, 0.04546384, 0.06845917],
        [0.8969521 , 0.10471964, 0.20355443, 0.57613833]]])
```

In [51]:

```
np.random.rand(3,4,4)
```

Out[51]:

```
array([[[0.31309825, 0.58530948, 0.33635523, 0.49633055],
        [0.47267002, 0.95377643, 0.66773942, 0.09348168],
        [0.73969539, 0.87253835, 0.02242645, 0.5190232 ],
        [0.79522891, 0.02079045, 0.83570849, 0.16411594]],

       [[0.63685495, 0.65352757, 0.46922357, 0.78573048],
        [0.35155634, 0.83225147, 0.38645397, 0.35075367],
        [0.63564374, 0.88926135, 0.03438776, 0.48047707],
        [0.82723936, 0.20748226, 0.02804363, 0.36289786]],

       [[0.53592529, 0.67984999, 0.37190491, 0.57156621],
        [0.18565573, 0.31073413, 0.85448432, 0.35745435],
        [0.65281879, 0.77167405, 0.7625496 , 0.83174549],
        [0.96307248, 0.42810495, 0.06684046, 0.91923208]]])
```

In [53]:

```
# 'randn' will try to generate the data only based on standard Normal distribution.
# 'rand' will try to generate the data only based on Normal distribution.

np.random.randn(2,4,4)
```

Out[53]:

```
array([[[ 0.75223986,  1.48376294, -1.22956701,  0.30770647],
        [ 0.44893225,  0.56569257, -1.25301764, -0.3385101 ],
        [-2.27863714,  1.23646555,  0.48835334,  1.09209586],
        [-0.78929473,  0.73846969,  0.03850178, -0.36042931]],

       [[ 1.61718583,  1.03761927, -0.90300769,  1.11122625],
        [ 0.80684852, -0.88874333, -1.72238444,  1.05488419],
        [-1.34869127,  0.46931046, -0.12250104, -0.93563045],
        [-2.06636593, -0.58137759, -0.05485826,  1.77819716]]])
```

In [55]:

```
d1 = np.random.randint(0,10,(4,4,2))
```

In [56]:

```
d1
```

Out[56]:

```
array([[[6, 5],
        [0, 1],
        [3, 1],
        [9, 2]],

       [[4, 9],
        [6, 4],
        [9, 6],
        [2, 4]],

       [[9, 6],
        [5, 6],
        [8, 7],
        [8, 6]],

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

In [57]:

```
d1.reshape(16,2)
```

Out[57]:

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

In [58]:

```
d1.reshape(8,4)
```

Out[58]:

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

In [59]:

```
# If we don't know the another dimension, simply we can keep any negative value,
#    then automatically it will take required number.

d1.reshape(8,-1)
```

Out[59]:

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

In [61]:

```
import pandas as pd
```

In [62]:

```
# we do this reshape because to represent the data in a dataframe.
#  DataFrame will only take 2-dimensional data.

pd.DataFrame(d1.reshape(32,-4))
```

Out[62]:

	0
0	6
1	5
2	0
3	1
4	3
5	1
6	9
7	2
8	4
9	9
10	6
11	4
12	9
13	6
14	2
15	4
16	9
17	6
18	5
19	6
20	8
21	7
22	8
23	6
24	4
25	8
26	8
27	5
28	6
29	8
30	9
31	3