In [1]:

```
# Array Shape
import numpy as np
```

In [2]:

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

Out[2]:

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

In [3]:

```
a.shape
```

Out[3]:

```
(2, 4)
```

In [3]:

```
# Reshaping arrays
# Reshaping means changing the shape of an array.
```

In [4]:

```
# Reshape From 1-D to 2-D
```

In [4]:

```
b = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

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

Out[4]:

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

In [ ]:

```

```

In [ ]:

```
# Reshape From 1-D to 3-D
```

In [5]:

```
c = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

cc = c.reshape(2, 3, 2)
cc
```

Out[5]:

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

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

In [ ]:

```

```

In [6]:

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

dd = d.reshape(3, 3)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-6-81bd4dd19db6> in <module>
      2 d = np.array([1, 2, 3, 4, 5, 6, 7, 8])
      3
----> 4 dd = d.reshape(3, 3)

ValueError: cannot reshape array of size 8 into shape (3,3)
```

In [7]:

```
# returns the view of array
a1 = np.array([1, 2, 3, 4, 5, 6, 7, 8])

a1.reshape(2, 4)
```

Out[7]:

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

In [8]:

```
a1
```

Out[8]:

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

In [9]:

```
a1.reshape(2, 4).base
```

Out[9]:

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

In [10]:

```
a1
```

Out[10]:

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

In [ ]:

```

```

In [11]:

```
# Convert 1D array with 8 elements to 3D array with 2x2 elements:
d1 = np.array([1, 2, 3, 4, 5, 6, 7, 8])

d2 = d1.reshape(2, 2, -1)
d2
```

Out[11]:

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

       [[5, 6],
        [7, 8]]])
```

In [12]:

```
# Flattening the arrays
# Flattening array means converting a multidimensional array into a 1D array.
e1 = np.array([[1, 2, 3], [4, 5, 6]])

ee = e1.reshape(-1)
ee
```

Out[12]:

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

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
# Iterating Arrays
```

In [13]:

```
# iterate on a 1-D array
a1 = np.array([1, 2, 3])

for x in a1:
  print(x)
```

```
1
2
3
```

In [14]:

```
# Iterating 2-D Arrays
a2 = np.array([[1, 2, 3], [4, 5, 6]])

for x in a2:
  print(x)
```

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

In [15]:

```
# to Get Complete Value is
a3 = np.array([[1, 2, 3], [4, 5, 6]])

for x in a3:
  for y in x:
    print(y)
  print()
```

```
1
2
3

4
5
6
```

In [ ]:

```

```

In [16]:

```
# Iterating 3-D Arrays

a4 = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

for x in a4:
  print(x)
```

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

In [17]:

```
a5 = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

for x in a5:
  for y in x:
    for z in y:
      print(z)
    print()
```

```
1
2
3

4
5
6

7
8
9

10
11
12
```

In [18]:

```
# Iterating Arrays Using nditer()
a6 = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

for x in np.nditer(a6):
  print(x)
```

```
1
2
3
4
5
6
7
8
```

In [19]:

```
# Iterating Array With Different Data Types
# op_dtypes argument to change the datatype of elements while iterating.
# NumPy does not change the data type of the element in-place (where the element is in arra
# so it needs some other space to perform this action, that extra space is called buffer,
# and in order to enable it in nditer() we pass flags=['buffered'].

a7 = np.array([1, 2, 3])

for x in np.nditer(a7, flags=['buffered'], op_dtypes=['S']):
  print(x)
```

```
b'1'
b'2'
b'3'
```

In [ ]:

```

```

In [20]:

```
# Iterating With Different Step Size
a8 = np.array([[1, 2, 3, 4, 5], [10, 20, 30, 40, 50]])

for x in np.nditer(a8[:, ::2]):
  print(x)
```

```
1
3
5
10
30
50
```

In [ ]:

```

```

In [21]:

```
# Enumerated Iteration Using ndenumerate()
# Enumeration means mentioning sequence number of somethings one by one.

a9 = np.array([1, 2, 3, 4, 5, 6, 7])

for idx, x in np.ndenumerate(a9):
  print(idx, x)
```

```
(0,) 1
(1,) 2
(2,) 3
(3,) 4
(4,) 5
(5,) 6
(6,) 7
```

In [22]:

```
# Enumerate on following 2D array's elements:
a10 = np.array([[1, 2, 3, 4, 5], [10, 20, 30, 40, 50]])

for indexID, value in np.ndenumerate(a10):
  print(indexID, value)
```

```
(0, 0) 1
(0, 1) 2
(0, 2) 3
(0, 3) 4
(0, 4) 5
(1, 0) 10
(1, 1) 20
(1, 2) 30
(1, 3) 40
(1, 4) 50
```

In [ ]:

```

```

In [ ]:

```

```