1. 6) NumPy Ndarray Manipulation Routines.

1) **reshape().**

Returns an array containing the same data with a new shape.

In [1]:
```python
import numpy as np

print('1----------------\n')
x = np.arange(6)
print(x)
print('\n---------------\n')

print('2----------------\n')
y = x.reshape((3, 2))
print(y)
print('\n---------------\n')
```

```
1----------------

[0 1 2 3 4 5]

---------------

2----------------

[[0 1]
 [2 3]
 [4 5]]

---------------
```

There is also an equivalent function to this ----->

In [2]:
```python
print('1----------------\n')
x = np.array([[0, 1, 3], [4, 5, 6]])
print(x)
print('\n---------------\n')

print('2----------------\n')
y = np.reshape(x, 6)
print(y)
print('\n---------------\n')
```

```
1----------------

[[0 1 3]
 [4 5 6]]

---------------

2----------------

[0 1 3 4 5 6]

---------------
```

2) **ravel().**

Returns a contiguous flattened array.

In [3]:
```python
print('1----------------\n')
x = np.array([[0, 1, 3], [4, 5, 6]])
print(x)
print('\n---------------\n')

print('2----------------\n')
y = np.ravel(x)
print(y)
print('\n---------------\n')
```

```
1----------------

[[0 1 3]
 [4 5 6]]

---------------

2----------------

[0 1 3 4 5 6]

---------------
```

3) **flatten().**

Return a copy of the array collapsed into one dimension.

In [4]:
```python
print('1----------------\n')
x = np.array([[0, 1, 3], [4, 5, 6]])
print(x)
print('\n---------------\n')

print('2----------------\n')
# By defualt in row order.
y = x.flatten()
print(y)
print('\n---------------\n')
```

```
1----------------

[[0 1 3]
 [4 5 6]]

---------------

2----------------

[0 1 3 4 5 6]

---------------
```

We can also change the way the array flattend to by passing an optional parameter.

‘C’ means to flatten in row-major (C-style) order.

In [5]:
```
print('1----------------\n')
x = np.array([[0, 1, 3], [4, 5, 6]])
print(x)
print('\n-----------------\n')

print('2----------------\n')
y = x.flatten('C')
print(y)
print('\n-----------------\n')
```

```
1----------------

[[0 1 3]
 [4 5 6]]

-----------------

2----------------

[0 1 3 4 5 6]

-----------------
```

‘F’ means to flatten in column-major (Fortran- style) order.

In [6]:
```
print('1----------------\n')
x = np.array([[0, 1, 3], [4, 5, 6]])
print(x)
print('\n-----------------\n')

print('2----------------\n')
y = x.flatten('F')
print(y)
print('\n-----------------\n')
```

```
1----------------

[[0 1 3]
 [4 5 6]]

-----------------

2----------------

[0 4 1 5 3 6]

-----------------
```

4) **stack().**

Joins a sequence of arrays along a new axis.

In [7]:
```python
# Joining Multiple arrays.
x = np.array([1, 2, 3], dtype = np.uint8)
y = np.array([4, 5, 6], dtype = np.uint8)

print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
print(y)
print('\n-----------------\n')

print('3----------------\n')
z = np.stack((x, y))
print(z)
print('\n-----------------\n')
```

```
1----------------

[1 2 3]

-----------------

2----------------

[4 5 6]

-----------------

3----------------

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

-----------------
```

The axis parameter specifies the index of the new axis in the dimensions of the result. For example, if axis=0 it will be the first dimens
and if axis=-1 it will be the last dimension.

In [8]:
```python
print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
print(y)
print('\n-----------------\n')

print('3----------------\n')
z = np.stack((x, y), axis = -1)
print(z)
print('\n-----------------\n')
```

```
1----------------

[1 2 3]

-----------------

2----------------

[4 5 6]

-----------------

3----------------

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

-----------------
```

In [9]:
```
print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
print(y)
print('\n-----------------\n')

print('3----------------\n')
z = np.stack((x, y), axis = 1)
print(z)
print('\n-----------------\n')
```

```
1----------------

[1 2 3]

-----------------

2----------------

[4 5 6]

-----------------

3----------------

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

-----------------
```

5) **vstack().**

Stacks arrays in sequence vertically (row wise).

In [12]:
```python
print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
print(y)
print('\n-----------------\n')

print('3----------------\n')
z = np.vstack((x, y))
print(z)
print('\nThe shape of the array: ', z.shape)
print('\n-----------------\n')
```

```
1----------------

[1 2 3]

-----------------

2----------------

[4 5 6]

-----------------

3----------------

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

The shape of the array:  (2, 3)

-----------------
```

6) **hstack().**

Stacks arrays in sequence horizontally (column wise).

In [13]:
```python
print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
print(y)
print('\n-----------------\n')

print('3----------------\n')
z = np.hstack((x, y))
print(z)
print('\nThe shape of the array: ', z.shape)
print('\n-----------------\n')
```

```
1----------------

[1 2 3]

-----------------

2----------------

[4 5 6]

-----------------

3----------------

[1 2 3 4 5 6]

The shape of the array:  (6,)

-----------------
```

7) **dstack().**

Stacks arrays in sequence depth wise (along third axis).

In [15]:
```python
print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
print(y)
print('\n-----------------\n')

print('3----------------\n')
z = np.dstack((x, y))
print(z)
print('\nThe shape of the array: ', z.shape)
print('\n-----------------\n')
```

```
1----------------

[1 2 3]

-----------------

2----------------

[4 5 6]

-----------------

3----------------

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

The shape of the array:  (1, 3, 2)

-----------------
```

8) **split().**

Splits an array into multiple sub-arrays as views into ary.

In [17]:

```
x = np.arange(9)

print('1----------------\n')
print(x)
print('\n-----------------\n')

a, b, c = np.split(x, 3)
print('2----------------\n')
print(a)
print('\n-----------------\n')

print('3----------------\n')
print(b)
print('\n-----------------\n')

print('3----------------\n')
print(c)
print('\n-----------------\n')
```

```
1----------------

[0 1 2 3 4 5 6 7 8]

-----------------

2----------------

[0 1 2]

-----------------

3----------------

[3 4 5]

-----------------

3----------------

[6 7 8]

-----------------
```

9) **dsplit().**

Splits array into multiple sub-arrays along the 3rd axis (depth).

In [31]:
```
x =  np.array([[[1, 2], [4, 5]],
               [[10, 11], [13, 14]]])

print('1----------------\n')
print(x)
print('\nThe shape of the array: ', x.shape)
print('\n-----------------\n')

y, z = np.dsplit(x, 2)
print('2----------------\n')
print(y)
print('\nThe shape of the array: ', y.shape)
print('\n-----------------\n')

print('3----------------\n')
print(z)
print('\nThe shape of the array: ', z.shape)
print('\n-----------------\n')
```

```
1----------------

[[[ 1  2]
  [ 4  5]]

 [[10 11]
  [13 14]]]

The shape of the array:  (2, 2, 2)

-----------------

2----------------

[[[ 1]
  [ 4]]

 [[10]
  [13]]]

The shape of the array:  (2, 2, 1)

-----------------
```

```
3----------------

[[[ 2]
  [ 5]]

 [[11]
  [14]]]

The shape of the array:  (2, 2, 1)

-----------------
```

10) **hsplit().**

Splits an array into multiple sub-arrays horizontally (column-wise).

In [33]:

```python
x =  np.array([[[1, 2], [4, 5]],
               [[10, 11], [13, 14]]])

print('1----------------\n')
print(x)
print('\nThe shape of the array: ', x.shape)
print('\n-----------------\n')

y, z = np.hsplit(x, 2)
print('2----------------\n')
print(y)
print('\nThe shape of the array: ', y.shape)
print('\n-----------------\n')

print('3----------------\n')
print(z)
print('\nThe shape of the array: ', z.shape)
print('\n-----------------\n')
```

```
1----------------

[[[ 1  2]
  [ 4  5]]

 [[10 11]
  [13 14]]]

The shape of the array:  (2, 2, 2)

-----------------

2----------------

[[[ 1  2]]

 [[10 11]]]

The shape of the array:  (2, 1, 2)

-----------------

3----------------
```

```
[[[ 4  5]]

 [[13 14]]]

The shape of the array:  (2, 1, 2)

-----------------
```

11) **vsplit().**

Splits an array into multiple sub-arrays vertically (row-wise).

In [34]:
```python
x =  np.array([[[1, 2], [4, 5]],
               [[10, 11], [13, 14]]])

print('1----------------\n')
print(x)
print('\nThe shape of the array: ', x.shape)
print('\n-----------------\n')

y, z = np.vsplit(x, 2)
print('2----------------\n')
print(y)
print('\nThe shape of the array: ', y.shape)
print('\n-----------------\n')

print('3----------------\n')
print(z)
print('\nThe shape of the array: ', z.shape)
```

```
1----------------

[[[ 1  2]
  [ 4  5]]

 [[10 11]
  [13 14]]]

The shape of the array:  (2, 2, 2)

-----------------

2----------------

[[[1 2]
  [4 5]]]

The shape of the array:  (1, 2, 2)

-----------------

3----------------

[[[10 11]
  [13 14]]]
```

```
The shape of the array:  (1, 2, 2)
```

12) **flip().**

Reverses the order of elements in an array along the given axis.

In [2]:
```python
import numpy as np

x = np.arange(16).reshape(4, 4)
print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
y = np.flip(x, axis = -1)
print(y)
print('\n-----------------\n')

print('3----------------\n')
y = np.flip(x, axis = 0)
print(y)
print('\n-----------------\n')

print('4----------------\n')
y = np.flip(x, axis = 1)
print(y)
print('\n-----------------\n')
```

```
1----------------

[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

-----------------

2----------------

[[ 3  2  1  0]
 [ 7  6  5  4]
 [11 10  9  8]
 [15 14 13 12]]

-----------------

3----------------

[[12 13 14 15]
```

```
 [ 8  9 10 11]
 [ 4  5  6  7]
 [ 0  1  2  3]]

-----------------

4----------------

[[ 3  2  1  0]
 [ 7  6  5  4]
 [11 10  9  8]
 [15 14 13 12]]

-----------------
```

13) **fliplr().**

Flips array in the left/right direction (horizontally (axis=1)).

In [5]:
```
print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
y = np.fliplr(x)
print(y)
print('\n-----------------\n')
```

```
1----------------

[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

-----------------

2----------------

[[ 3  2  1  0]
 [ 7  6  5  4]
 [11 10  9  8]
 [15 14 13 12]]

-----------------
```

14) **flipud().**

Flips array in the up/down direction (vertically (axis=0)).

In [6]:
```
print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
y = np.flipud(x)
print(y)
print('\n-----------------\n')
```

```
1----------------

[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

-----------------

2----------------

[[12 13 14 15]
 [ 8  9 10 11]
 [ 4  5  6  7]
 [ 0  1  2  3]]

-----------------
```

15) **roll().**

Rolls array elements along a given axis.

In [7]:
```python
print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
y = np.roll(x, 8)
print(y)
print('\n-----------------\n')
```

```
1----------------

[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

-----------------

2----------------

[[ 8  9 10 11]
 [12 13 14 15]
 [ 0  1  2  3]
 [ 4  5  6  7]]

-----------------
```

16) **rot90().**

Rotates an array by 90 degrees in the plane specified by axes.

In [8]:
```python
print('1----------------\n')
print(x)
print('\n-----------------\n')

print('2----------------\n')
y = np.rot90(x)
print(y)
print('\n-----------------\n')
```

```
1----------------

[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

-----------------

2----------------

[[ 3  7 11 15]
 [ 2  6 10 14]
 [ 1  5  9 13]
 [ 0  4  8 12]]

-----------------
```