* Array Shape & Re-Shape :=>

=> Shape of an Array =>

The Shape of an array is the no. of elements in each dimensions.

```
import numpy as np

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

print(arr.shape)
```

This is 2d array.

4 element present in each dimensions.

O/P => 2, 4.

* Reshapping arrays :=>

means changing the Shape of an array.

The shape of an array is the no. of elements in each dimensions.

By reshaping we can add or remove dimension or change no. of elements in each dimension.

* Re Shape From 1-D to 2-D

⇒ Example: ⇒

- Convert the following 1-D array with 12 elements into a 2-D array.

The outermost dimensions will have 4 arrays, each with 3 elements.

```
import numpy as np

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

newarr = arr.reshape(4,3)

print(newarr)

* Can we Reshape Into any Shape?

Yes, as long as the elements required for reshaping are equal in both shapes.

We can reshape an 8 element 1D array into 4 elements in 2 rows 2D array but, we cannot reshape it into a 3 elements 3 rows 2D array as that would require 3×3 = 9 elements.

* Flattening the arrays :-

means converting a multidimensional array into a 1D array.

We can use reshape(-1) to do this.

Import numpy as np

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

newarr = arr.reshape(-1)

print(newarr) ⇒ array([1, 2, 3, 4, 5, 6])

* Example →

(1) import numpy as np

(1) a = np.array([1, 2, 3, 4, 5])
a.shape
O/P ⇒ (5,)

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

b.shape
O/P ⇒ (2, 4)

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

newarr = C.reshape(4, 3)

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

C.shape
O/P ⇒ (12,)

new-arr.shape
O/P ⇒ (4, 3)

④ new-arr2 = C.reshape(2, 3, 2) ⇒ 3D.

new-arr2

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

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

```
new_arr2.shape
(2, 3, 2)
```

=> Indexing