We can reshape an array to another dimension using the reshape() function.

Lets consider a program in which ‘arr’ is a rank 1 array and after reshape it’s converted to a rank 2 array ‘arr1’.

In [1]:

```
import numpy as np
list=[1,2,3,4,5]
arr= np.array(list)
print("The NumPy Array is:", arr)
arr1=arr.reshape(1,-1)
print("The Reshape NumPy array is:", arr1)
```

```
The NumPy Array is: [1 2 3 4 5]
The Reshape NumPy array is: [[1 2 3 4 5]]
```

we call the reshape() function with two arguments. The first 1 indicates that we want to convert it into rank 2 array with 1 row, and the -1 indicates that we will leave it to the reshape() function to create the correct number of columns. Of course, in this program, it is clear that after reshaping there will be five columns, so we can call the reshape() function as reshape(1,5). In more complex cases, however, it is always convenient to be able to use -1 to let the function decide on the number of rows or columns to create.

Let’s consider another program to reshape ‘arr1’ (which is a rank 2 array) to rank 1.

In [2]:

```
import numpy as np
list=[1,2,3,4,5]
arr1= np.array([list])
print("The NumPy Array is:", arr1)
arr=arr1.reshape(-1,)
print("The Reshape NumPy array is:", arr)
```

```
The NumPy Array is: [[1 2 3 4 5]]
The Reshape NumPy array is: [1 2 3 4 5]
```

Here, -1 indicates that the function decide how many rows to create as long as the end result is a rank 1 array.

To convert a rank 2 array to a rank 1 array, we can also use the flatten() or ravel() functions. The flatten() function always returns a copy of the array, while the ravel() and reshape() functions return a view (reference) of the original array

In [ ]:

```

```