Reshaping

A NumPy array can be restructured to take on a different shape. Let's convert a 1-D array to a $m \times n$ matrix.

```
# make 20 elements evenly spaced between 0 and 5
a = np.linspace(0,5,20)
a
'Output':
array([ 0.        ,  0.26315789,  0.52631579,  0.78947368,  1.05263158,
        1.31578947,  1.57894737,  1.84210526,  2.10526316,  2.36842105,
        2.63157895,  2.89473684,  3.15789474,  3.42105263,  3.68421053,
        3.94736842,  4.21052632,  4.47368421,  4.73684211,  5.        ])
# observe that a is a 1-D array
a.shape
'Output': (20,)
# reshape into a 5 x 4 matrix
A = a.reshape(5, 4)
A
'Output':
array([[ 0.        ,  0.26315789,  0.52631579,  0.78947368],
       [ 1.05263158,  1.31578947,  1.57894737,  1.84210526],
       [ 2.10526316,  2.36842105,  2.63157895,  2.89473684],
       [ 3.15789474,  3.42105263,  3.68421053,  3.94736842],
       [ 4.21052632,  4.47368421,  4.73684211,  5.        ]])
# The vector a has been reshaped into a 5 by 4 matrix A
A.shape
'Output': (5, 4)
```

Reshape vs. Resize Method

NumPy has the **np.reshape** and **np.resize** methods. The reshape method returns an ndarray with a modified shape without changing the original array, whereas the resize method changes the original array. Let's see an example.

```
# generate 9 elements evenly spaced between 0 and 5
a = np.linspace(0,5,9)
a
'Output': array([ 0.   ,  0.625,  1.25 ,  1.875,  2.5  ,  3.125,  3.75 ,
4.375,  5.   ])
# the original shape
a.shape
'Output': (9,)
# call the reshape method
a.reshape(3,3)
'Output':
array([[ 0.   ,  0.625,  1.25 ],
       [ 1.875,  2.5  ,  3.125],
       [ 3.75 ,  4.375,  5.   ]])
# the original array maintained its shape
a.shape
'Output': (9,)
# call the resize method - resize does not return an array
a.resize(3,3)
# the resize method has changed the shape of the original array
a.shape
'Output': (3, 3)
```

Stacking Arrays

NumPy has methods for concatenating arrays – also called stacking. The methods hstack and vstack are used to stack several arrays along the horizontal and vertical axis, respectively.

```
# create a 2x2 matrix of random integers in the range of 1 to 20
A = np.random.randint(1, 50, size=[3,3])
B = np.random.randint(1, 50, size=[3,3])
# print out the arrays
A
```