

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

```
In [2]: np1 = np.linspace(1,10,7)      # Linearly spaced
```

```
In [3]: np1
```

```
Out[3]: array([ 1. ,  2.5,  4. ,  5.5,  7. ,  8.5, 10. ])
```

```
In [4]: np.ones(3)
```

```
Out[4]: array([1., 1., 1.])
```

```
In [5]: np.ones((3,3))
```

```
Out[5]: array([[1., 1., 1.],
               [1., 1., 1.],
               [1., 1., 1.]])
```

```
In [6]: np.zeros((4,4))*2
```

```
Out[6]: array([[0., 0., 0., 0.],
               [0., 0., 0., 0.],
               [0., 0., 0., 0.],
               [0., 0., 0., 0.]])
```

```
In [7]: np.eye(3)
```

```
Out[7]: array([[1., 0., 0.],
               [0., 1., 0.],
               [0., 0., 1.]])
```

```
In [8]: np.eye(5)
```

```
Out[8]: array([[1., 0., 0., 0., 0.],
               [0., 1., 0., 0., 0.],
               [0., 0., 1., 0., 0.],
               [0., 0., 0., 1., 0.],
               [0., 0., 0., 0., 1.]])
```

```
In [9]: np.eye(3,4)
```

```
Out[9]: array([[1., 0., 0., 0.],
               [0., 1., 0., 0.],
               [0., 0., 1., 0.]])
```

```
In [10]: np.eye(4,3)
```

```
Out[10]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.],
                [0., 0., 0.]])
```

```
In [11]: np.diag([3,6,9])
```

```
Out[11]: array([[3, 0, 0],
                [0, 6, 0],
                [0, 0, 9]])
```

```
In [12]: np.diag([3,6,9,12])**3
```

```
Out[12]: array([[ 27,   0,   0,   0],
                [  0, 216,   0,   0],
```

```
[ 0, 0, 729, 0],  
[ 0, 0, 0, 1728]], dtype=int32)
```

```
In [13]: np.diag([3,6,9,12,19])//3
```

```
Out[13]: array([[1, 0, 0, 0, 0],  
               [0, 2, 0, 0, 0],  
               [0, 0, 3, 0, 0],  
               [0, 0, 0, 4, 0],  
               [0, 0, 0, 0, 6]], dtype=int32)
```

```
In [14]: np1
```

```
Out[14]: array([ 1. ,  2.5,  4. ,  5.5,  7. ,  8.5, 10. ])
```

```
In [15]: id(np1)
```

```
Out[15]: 1748499350912
```

```
In [16]: np2 = np1
```

```
In [17]: np2
```

```
Out[17]: array([ 1. ,  2.5,  4. ,  5.5,  7. ,  8.5, 10. ])
```

```
In [18]: id(np2)
```

```
Out[18]: 1748499350912
```

```
In [19]: np.shares_memory(np1,np2)
```

```
Out[19]: True
```

```
In [20]: np3 = np2.copy()
```

```
In [21]: id(np2),id(np3)
```

```
Out[21]: (1748499350912, 1748499489616)
```

```
In [22]: np.shares_memory(np2,np3)
```

```
Out[22]: False
```

```
In [23]: fours = np.ones((5,5))*4
```

```
In [24]: fours
```

```
Out[24]: array([[4., 4., 4., 4., 4.],  
               [4., 4., 4., 4., 4.],  
               [4., 4., 4., 4., 4.],  
               [4., 4., 4., 4., 4.],  
               [4., 4., 4., 4., 4.]])
```

```
In [25]: fours[2,4] = 54
```

```
In [26]: fours
```

```
Out[26]: array([[ 4.,  4.,  4.,  4.,  4.],  
               [ 4.,  4.,  4.,  4.,  4.],  
               [ 4.,  4.,  4.,  4., 54.]])
```

```
[ 4.,  4.,  4.,  4.,  4.],  
[ 4.,  4.,  4.,  4.,  4.]])
```

```
In [27]: fours[2,4]           # Only returns the specific element
```

```
Out[27]: 54.0
```

```
In [28]: fours[[2,4]]        # Returns the 2nd and 4th row of a matrix
```

```
Out[28]: array([[ 4.,  4.,  4.,  4., 54.],  
                [ 4.,  4.,  4.,  4.,  4.]])
```

```
In [29]: diag_mat = np.diag([1,2,3,4,5,6])
```

```
In [30]: diag_mat[[2,5]]
```

```
Out[30]: array([[0, 0, 3, 0, 0, 0],  
                [0, 0, 0, 0, 0, 6]])
```

```
In [31]: np.random.rand(1,5)
```

```
Out[31]: array([[0.86487613, 0.98354167, 0.31389564, 0.80327425, 0.60180532]])
```

```
In [32]: np.random.randn(1,5)
```

```
Out[32]: array([[ 0.87165648,  1.23125057,  0.97498759, -0.84334351,  0.55134073]])
```

```
In [33]: A = np.array([[1,2,3],[4,5,6],[1,1,1]])
```

```
In [34]: A
```

```
Out[34]: array([[1, 2, 3],  
                [4, 5, 6],  
                [1, 1, 1]])
```

```
In [35]: A.shape
```

```
Out[35]: (3, 3)
```

```
In [36]: A.ndim
```

```
Out[36]: 2
```

```
In [37]: type(A)
```

```
Out[37]: numpy.ndarray
```

```
In [38]: A.diagonal()
```

```
Out[38]: array([1, 5, 1])
```

```
In [39]: B = np.array([[1,2,3],[4,5,6],[1,1,1]])
```

```
In [40]: id(A)
```

```
Out[40]: 1748499583584
```

```
In [41]: id(B)
```

Out[41]: 1748499608640

In [42]: `C = A`

In [43]: `id(A)`

Out[43]: 1748499583584

In [44]: `id(C)`

Out[44]: 1748499583584

In [45]: `A == B`

Out[45]: `array([[ True, True, True],  
 [ True, True, True],  
 [ True, True, True]])`

In [46]: `A == C`

Out[46]: `array([[ True, True, True],  
 [ True, True, True],  
 [ True, True, True]])`

In [47]: `C == B`

Out[47]: `array([[ True, True, True],  
 [ True, True, True],  
 [ True, True, True]])`

In [48]: `C = C * 2`

In [49]: `C == B`

Out[49]: `array([[False, False, False],  
 [False, False, False],  
 [False, False, False]])`

In [50]: `C`

Out[50]: `array([[ 2, 4, 6],  
 [ 8, 10, 12],  
 [ 2, 2, 2]])`

In [51]: `type(C)`

Out[51]: `numpy.ndarray`

In [52]: `isinstance(C,np.int)`

Out[52]: `False`

In [53]: `isinstance(C,np.ndarray)`

Out[53]: `True`

In [54]: `np.asarray(((4,5,6),(6,7,8)))`

Out[54]: `array([[4, 5, 6],  
 [6, 7, 8]])`

```
In [55]: np.array_equal(C, B)
```

```
Out[55]: False
```

```
In [56]: np.logical_or(A,B)
```

```
Out[56]: array([[ True,  True,  True],
               [ True,  True,  True],
               [ True,  True,  True]])
```

```
In [57]: np.logical_and(A,B)
```

```
Out[57]: array([[ True,  True,  True],
               [ True,  True,  True],
               [ True,  True,  True]])
```

```
In [58]: B = B * 0
```

```
In [59]: A
```

```
Out[59]: array([[1, 2, 3],
               [4, 5, 6],
               [1, 1, 1]])
```

```
In [60]: B
```

```
Out[60]: array([[0, 0, 0],
               [0, 0, 0],
               [0, 0, 0]])
```

```
In [61]: np.logical_or(A,B)
```

```
Out[61]: array([[ True,  True,  True],
               [ True,  True,  True],
               [ True,  True,  True]])
```

```
In [62]: np.logical_and(A,B)
```

```
Out[62]: array([[False, False, False],
               [False, False, False],
               [False, False, False]])
```

```
In [63]: np.sin(np.arange(6))
```

```
Out[63]: array([ 0.          ,  0.84147098,  0.90929743,  0.14112001, -0.7568025 ,
               -0.95892427])
```

```
In [64]: np.log(5)
```

```
Out[64]: 1.6094379124341003
```

```
In [65]: np.exp(np.arange(1,10,2))
```

```
Out[65]: array([2.71828183e+00, 2.00855369e+01, 1.48413159e+02, 1.09663316e+03,
               8.10308393e+03])
```

```
In [66]: np.square(np.arange(1,10,2))
```

```
Out[66]: array([ 1,  9, 25, 49, 81], dtype=int32)
```

```
In [67]: A
```

```
Out[67]: array([[1, 2, 3],
               [4, 5, 6],
```

```
[1, 1, 1]])
```

```
In [68]: flat_A = A.flatten()
```

```
In [69]: flat_A
```

```
Out[69]: array([1, 2, 3, 4, 5, 6, 1, 1, 1])
```

```
In [70]: flat_A.ndim
```

```
Out[70]: 1
```

```
In [71]: flat_A.shape
```

```
Out[71]: (9,)
```

```
In [72]: A
```

```
Out[72]: array([[1, 2, 3],
               [4, 5, 6],
               [1, 1, 1]])
```

```
In [73]: A.ravel('F')
```

```
Out[73]: array([1, 4, 1, 2, 5, 1, 3, 6, 1])
```

```
In [74]: A[2:4] = [2,3,4]
```

```
In [75]: A
```

```
Out[75]: array([[1, 2, 3],
               [4, 5, 6],
               [2, 3, 4]])
```

```
In [76]: A = np.append(A, [[6,7,8]],axis=0)
```

```
In [77]: A
```

```
Out[77]: array([[1, 2, 3],
               [4, 5, 6],
               [2, 3, 4],
               [6, 7, 8]])
```

```
In [78]: A = np.insert(A,2,[[0,9,8]],axis=0)
A
```

```
Out[78]: array([[1, 2, 3],
               [4, 5, 6],
               [0, 9, 8],
               [2, 3, 4],
               [6, 7, 8]])
```

```
In [79]: %%timeit
np.insert(A,3,[[1,1,1,1,1]],axis=1)
```

99 µs ± 20.3 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)

```
In [80]: A
```

```
Out[80]: array([[1, 2, 3],
               [4, 5, 6],
               [0, 9, 8],
```

```
[2, 3, 4],  
[6, 7, 8]])
```

```
In [81]: A = np.insert(A,3,[[1,1,1,1,1]],axis=1)  
A
```

```
Out[81]: array([[1, 2, 3, 1],  
               [4, 5, 6, 1],  
               [0, 9, 8, 1],  
               [2, 3, 4, 1],  
               [6, 7, 8, 1]])
```

```
In [82]: A
```

```
Out[82]: array([[1, 2, 3, 1],  
               [4, 5, 6, 1],  
               [0, 9, 8, 1],  
               [2, 3, 4, 1],  
               [6, 7, 8, 1]])
```

```
In [83]: A.sum()
```

```
Out[83]: 73
```

```
In [84]: A.sum(axis=0)
```

```
Out[84]: array([13, 26, 29,  5])
```

```
In [85]: A.sum(axis=1)
```

```
Out[85]: array([ 7, 16, 18, 10, 22])
```

```
In [86]: A.min()
```

```
Out[86]: 0
```

```
In [87]: A.min(axis=1)
```

```
Out[87]: array([1, 1, 0, 1, 1])
```

```
In [88]: A.min(axis=0)
```

```
Out[88]: array([0, 2, 3, 1])
```

```
In [89]: A
```

```
Out[89]: array([[1, 2, 3, 1],  
               [4, 5, 6, 1],  
               [0, 9, 8, 1],  
               [2, 3, 4, 1],  
               [6, 7, 8, 1]])
```

```
In [90]: A.argmin(0)
```

```
Out[90]: array([2, 0, 0, 0], dtype=int64)
```

```
In [91]: A.argmax()
```

```
Out[91]: 9
```

```
In [92]: np.all([True, True, False])
```

Out[92]: False

In [93]: `np.any([True, False, False])`

Out[93]: True

In [94]: `np.all([1,1,1,1,1, 0])`

Out[94]: False

In [95]: `np.any([1,1,1,1,1])`

Out[95]: True

In [96]: `np.any([1,1,1,1,0])`

Out[96]: True

In [97]: `np.any([0,1,0,0,0,0])`

Out[97]: True

In [98]: A

Out[98]: `array([[1, 2, 3, 1],  
[4, 5, 6, 1],  
[0, 9, 8, 1],  
[2, 3, 4, 1],  
[6, 7, 8, 1]])`

In [99]: B

Out[99]: `array([[0, 0, 0],  
[0, 0, 0],  
[0, 0, 0]])`

In [100... `B = np.append(B, [[5,6,7],[7,8,9],[9,9,9]], axis=0)`

In [101... `B = np.delete(A,[3,4,5],axis=1)`

c:\users\rajsh\appdata\local\programs\python\python36\lib\site-packages\ipykernel\_launcher.py:1: DeprecationWarning: in the future out of bounds indices will raise an error instead of being ignored by `numpy.delete`.  
 """Entry point for launching an IPython kernel.

In [102... B

Out[102... `array([[1, 2, 3],  
[4, 5, 6],  
[0, 9, 8],  
[2, 3, 4],  
[6, 7, 8]])`

In [103... `A = np.delete(A,[3,4,5],axis=1)`

c:\users\rajsh\appdata\local\programs\python\python36\lib\site-packages\ipykernel\_launcher.py:1: DeprecationWarning: in the future out of bounds indices will raise an error instead of being ignored by `numpy.delete`.  
 """Entry point for launching an IPython kernel.

In [104... A



```
Out[104...] array([[1, 2, 3],
                  [4, 5, 6],
                  [0, 9, 8],
                  [2, 3, 4],
                  [6, 7, 8]])
```

```
In [105...] A * B
```

```
Out[105...] array([[ 1,  4,  9],
                  [16, 25, 36],
                  [ 0, 81, 64],
                  [ 4,  9, 16],
                  [36, 49, 64]])
```

```
In [106...] B = B.reshape(3,5)
            A.dot(B)
```

```
Out[106...] array([[ 22,  14,  39,  41,  33],
                  [ 52,  32,  93,  98,  78],
                  [ 78,  32, 129, 128,  82],
                  [ 32,  20,  57,  60,  48],
                  [ 72,  44, 129, 136, 108]])
```

### NUMPY Broadcasting

```
In [107...] A
```

```
Out[107...] array([[1, 2, 3],
                  [4, 5, 6],
                  [0, 9, 8],
                  [2, 3, 4],
                  [6, 7, 8]])
```

```
In [108...] B
```

```
Out[108...] array([[1, 2, 3, 4, 5],
                  [6, 0, 9, 8, 2],
                  [3, 4, 6, 7, 8]])
```

```
In [109...] A.T
```

```
Out[109...] array([[1, 4, 0, 2, 6],
                  [2, 5, 9, 3, 7],
                  [3, 6, 8, 4, 8]])
```

```
In [110...] B.T
```

```
Out[110...] array([[1, 6, 3],
                  [2, 0, 4],
                  [3, 9, 6],
                  [4, 8, 7],
                  [5, 2, 8]])
```

```
In [111...] np.transpose([A[:,1]])
```

```
Out[111...] array([[2],
                  [5],
                  [9],
                  [3],
                  [7]])
```

```
In [112...] D = np.tile(np.transpose([A[:,1]]),(1,3))
```

```
In [113...] D
```

```
Out[113...] array([[2, 2, 2],
                  [5, 5, 5],
```

```
[9, 9, 9],
[3, 3, 3],
[7, 7, 7]])
```

In [114... D + A

```
Out[114... array([[ 3,  4,  5],
          [ 9, 10, 11],
          [ 9, 18, 17],
          [ 5,  6,  7],
          [13, 14, 15]])
```

In [115... A

```
Out[115... array([[1, 2, 3],
          [4, 5, 6],
          [0, 9, 8],
          [2, 3, 4],
          [6, 7, 8]])
```

```
In [116]: A.shape
```

```
Out[116]: (5, 3)
```

```
In [117... np.tile(A,(10,3)).shape
```

```
Out[117]: (50, 9)
```

```
In [118... np.tile(A,(10,3))
```

[illegible]

```
[0, 9, 8, 0, 9, 8, 0, 9, 8],
[2, 3, 4, 2, 3, 4, 2, 3, 4],
[6, 7, 8, 6, 7, 8, 6, 7, 8],
[1, 2, 3, 1, 2, 3, 1, 2, 3],
[4, 5, 6, 4, 5, 6, 4, 5, 6],
[0, 9, 8, 0, 9, 8, 0, 9, 8],
[2, 3, 4, 2, 3, 4, 2, 3, 4],
[6, 7, 8, 6, 7, 8, 6, 7, 8],
[1, 2, 3, 1, 2, 3, 1, 2, 3],
[4, 5, 6, 4, 5, 6, 4, 5, 6],
[0, 9, 8, 0, 9, 8, 0, 9, 8],
[2, 3, 4, 2, 3, 4, 2, 3, 4],
[6, 7, 8, 6, 7, 8, 6, 7, 8]])
```

```
In [119... AA = np.tile(A,(2,1))
```

```
In [120... AA
```

```
Out[120... array([[1, 2, 3],
         [4, 5, 6],
         [0, 9, 8],
         [2, 3, 4],
         [6, 7, 8],
         [1, 2, 3],
         [4, 5, 6],
         [0, 9, 8],
         [2, 3, 4],
         [6, 7, 8]])
```

```
In [121... XX = np.arange(0,10,2)
```

```
In [122... XX
```

```
Out[122... array([0, 2, 4, 6, 8])
```

```
In [123... XX.ndim
```

```
Out[123... 1
```

```
In [124... XX.shape
```

```
Out[124... (5,)
```

```
In [125... new_XX = XX[np.newaxis,:]
```

```
In [126... new_XX
```

```
Out[126... array([[0, 2, 4, 6, 8]])
```

```
In [127... new_XX.ndim
```

```
Out[127... 2
```

```
In [128... new_XX.shape
```

```
Out[128... (1, 5)
```

```
In [129... new_XX1 = XX[:,np.newaxis]
```

```
In [130... new_XX1
```

```
Out[130...] array([[0],  
                [2],  
                [4],  
                [6],  
                [8]])
```

```
In [131...] new_XX1.shape
```

```
Out[131...] (5, 1)
```

```
In [132...] new_XX1.ndim
```

```
Out[132...] 2
```

```
In [133...] new_XX2 = new_XX1[:,np.newaxis]
```

```
In [134...] new_XX2
```

```
Out[134...] array([[0]],  
                [[2]],  
                [[4]],  
                [[6]],  
                [[8]])
```

```
In [135...] new_XX2.shape
```

```
Out[135...] (5, 1, 1)
```

```
In [136...] new_XX2.ndim
```

```
Out[136...] 3
```

```
In [137...] YY = np.arange(4*3*2).reshape(4,3,2)
```

```
In [138...] YY
```

```
Out[138...] array([[ 0,  1],  
                  [ 2,  3],  
                  [ 4,  5]],  
                  [[ 6,  7],  
                  [ 8,  9],  
                  [10, 11]],  
                  [[12, 13],  
                  [14, 15],  
                  [16, 17]],  
                  [[18, 19],  
                  [20, 21],  
                  [22, 23]])
```

```
In [139...] YY.ndim
```

```
Out[139...] 3
```

```
In [140...] YY.shape
```

```
Out[140...] (4, 3, 2)
```

```
In [141... YY[0]
```

```
Out[141... array([[0, 1],  
          [2, 3],  
          [4, 5]])
```

```
In [142... YY[0][1][1] = 9
```

```
In [143... YY
```

```
Out[143... array([[ 0,  1],  
          [ 2,  9],  
          [ 4,  5]],  
  
          [[ 6,  7],  
          [ 8,  9],  
          [10, 11]],  
  
          [[12, 13],  
          [14, 15],  
          [16, 17]],  
  
          [[18, 19],  
          [20, 21],  
          [22, 23]]])
```

```
In [144... YY[0,1,1]
```

```
Out[144... 9
```

```
In [145... pp = np.arange(1,10)
```

```
In [146... pp
```

```
Out[146... array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [147... qq = pp
```

```
In [148... pp = pp.reshape(3,3)
```

```
In [149... qq
```

```
Out[149... array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [150... pp
```

```
Out[150... array([[1, 2, 3],  
          [4, 5, 6],  
          [7, 8, 9]])
```

```
In [151... rr = pp
```

```
In [152... pp.resize(1,9)
```

```
In [153... pp
```

```
Out[153... array([[1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

```
In [154... qq
```

```
Out[154...] array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [155...] pp.resize(3,3)
```

```
In [156...] pp.resize((9,0))
```

```
-----  
ValueError                                Traceback (most recent call last)  
<ipython-input-156-7d505926ffe5> in <module>  
----> 1 pp.resize((9,0))
```

```
ValueError: cannot resize this array: it does not own its data
```

```
In [157...] pp
```

```
Out[157...] array([[1, 2, 3],  
                  [4, 5, 6],  
                  [7, 8, 9]])
```

```
In [158...] qq
```

```
Out[158...] array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [159...] id(pp), id(qq)
```

```
Out[159...] (1748499875920, 1748499873920)
```

```
In [160...] tt = pp
```

```
In [161...] id(pp), id(tt)
```

```
Out[161...] (1748499875920, 1748499875920)
```

```
In [162...] pp.shape, qq.shape, tt.shape
```

```
Out[162...] ((3, 3), (9,), (3, 3))
```

```
In [163...] pp.ndim, qq.ndim, tt.ndim
```

```
Out[163...] (2, 1, 2)
```

```
In [164...] pp.resize(9,1)
```

```
In [165...] pp
```

```
Out[165...] array([[1],  
                  [2],  
                  [3],  
                  [4],  
                  [5],  
                  [6],  
                  [7],  
                  [8],  
                  [9]])
```

```
In [166...] tt
```

```
Out[166...] array([[1],  
                  [2],  
                  [3],  
                  [4],  
                  [5],
```

```
[6],  
[7],  
[8],  
[9]])
```

```
In [167... id(pp), id(tt)
```

```
Out[167... (1748499875920, 1748499875920)
```

```
In [168... pp.resize((3,3))
```

```
In [169... pp
```

```
Out[169... array([[1, 2, 3],  
        [4, 5, 6],  
        [7, 8, 9]])
```

```
In [170... tt
```

```
Out[170... array([[1, 2, 3],  
        [4, 5, 6],  
        [7, 8, 9]])
```

```
In [171... pp,tt
```

```
Out[171... (array([[1, 2, 3],  
        [4, 5, 6],  
        [7, 8, 9]]),  
        array([[1, 2, 3],  
        [4, 5, 6],  
        [7, 8, 9]]))
```

```
In [172... pp = pp.resize((9,1))
```

```
In [173... tt
```

```
Out[173... array([[1],  
        [2],  
        [3],  
        [4],  
        [5],  
        [6],  
        [7],  
        [8],  
        [9]])
```

```
In [174... arr1 = np.arange(1,16)
```

```
In [175... arr1
```

```
Out[175... array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

```
In [176... arr1.resize((5,3))
```

```
In [177... import numpy as np
```

```
In [178... a = np.array([0,1,2,3,0,0,0,0])
```

```
In [179... a
```

```
Out[179... array([0, 1, 2, 3, 0, 0, 0, 0])
```

```
In [180... b = a
```

```
In [181... id(a), id(b)
```

```
Out[181... (1748499874560, 1748499874560)
```

```
In [182... a.resize((4,2))
```

```
In [183... a
```

```
Out[183... array([[0, 1],  
        [2, 3],  
        [0, 0],  
        [0, 0]])
```

```
In [184... id(a), id(b)
```

```
Out[184... (1748499874560, 1748499874560)
```

```
In [185... a , b
```

```
Out[185... (array([[0, 1],  
        [2, 3],  
        [0, 0],  
        [0, 0]]),  
array([[0, 1],  
        [2, 3],  
        [0, 0],  
        [0, 0]]))
```

```
In [186... a.resize((4,0),refcheck=False)
```

```
In [187... b
```

```
Out[187... array([], shape=(4, 0), dtype=int32)
```

```
In [188... kk = np.array([1,2,3,4])
```

```
In [189... kk.resize((2,))
```

```
In [190... kk
```

```
Out[190... array([1, 2])
```

```
In [191... oo = kk
```

```
In [192... oo.resize((1,))
```

```
-----  
ValueError                                Traceback (most recent call last)  
<ipython-input-192-6d993e236f8e> in <module>  
----> 1 oo.resize((1,))
```

```
ValueError: cannot resize an array that references or is referenced  
by another array in this way.  
Use the np.resize function or refcheck=False
```

```
In [193... tt.shape
```

```
Out[193... (9, 1)
```



```
In [194...] tt.resize((3,3))
```

```
In [195...] tt
```

```
Out[195...] array([[1, 2, 3],  
          [4, 5, 6],  
          [7, 8, 9]])
```

```
In [196...] tt.ndim, tt.shape
```

```
Out[196...] (2, (3, 3))
```

```
In [222...] idx_tt = tt.argsort(axis=0)
```

```
In [223...] idx_tt
```

```
Out[223...] array([[0, 0, 0],  
          [1, 1, 1],  
          [2, 2, 2]], dtype=int64)
```

```
In [199...] idx_tt.shape, idx_tt.ndim
```

```
Out[199...] ((3, 3), 2)
```

```
In [200...] new_tt = tt[idx_tt]
```

```
In [201...] new_tt.shape, new_tt.ndim
```

```
Out[201...] ((3, 3, 3), 3)
```

```
In [202...] new_tt
```

```
Out[202...] array([[1, 2, 3],  
          [1, 2, 3],  
          [1, 2, 3]],  
  
          [[4, 5, 6],  
          [4, 5, 6],  
          [4, 5, 6]],  
  
          [[7, 8, 9],  
          [7, 8, 9],  
          [7, 8, 9]])
```

```
In [203...] tt = tt*2
```

```
In [228...] import os
```

```
In [229...] os.getcwd()
```

```
Out[229...] 'E:\\STUDY\\PROJECTS\\AAIC_Practice\\MODULES\\Module_1\\Module_1_All_about_Python'
```

```
In [245...] np.loadtxt('../..\\CC_Before_train_test_split.csv', skiprows=1, dtype=np.str, delimiter=
```

```
Out[245...] array([[ '18', '4.0', '15.0', ..., '0.0', '0', '0'],  
          [ '15', '1.0', '14.0', ..., '0.0', '0', '0'],  
          [ '34', '1.0', '?', ..., '0.0', '0', '0'],  
          ...,  
          [ '25', '2.0', '17.0', ..., '0.0', '0', '0'],  
          [ '33', '2.0', '24.0', ..., '0.0', '0', '0'],  
          [ '29', '2.0', '20.0', ..., '0.0', '0', '0']], dtype='<U12')
```

```
In [218... tt
```

```
Out[218... array([[ 2,  4,  6],
          [ 8, 10, 12],
          [14, 16, 18]])
```

```
In [224... tt.argsort(axis=0), tt.argsort(axis=1)
```

```
Out[224... (array([[0, 0, 0],
          [1, 1, 1],
          [2, 2, 2]], dtype=int64),
          array([[0, 1, 2],
          [0, 1, 2],
          [0, 1, 2]], dtype=int64))
```

```
In [246... np.take_along_axis(tt,tt.argsort(axis=1),axis=1), np.take_along_axis(tt,tt.argsort(a
```

```
Out[246... (array([[ 2,  4,  6],
          [ 8, 10, 12],
          [14, 16, 18]]),
          array([[ 2,  2,  2],
          [10, 10, 10],
          [18, 18, 18]]))
```

```
In [207... ttt = np.arange(1,10)
```

```
In [208... ttt
```

```
Out[208... array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [209... ttt.resize((3,3))
```

```
In [210... ttt[2][2] = 2
```

```
In [211... ttt
```

```
Out[211... array([[1, 2, 3],
          [4, 5, 6],
          [7, 8, 2]])
```

```
In [212... ttt.sort(axis=0)
```

```
In [213... ttt
```

```
Out[213... array([[1, 2, 2],
          [4, 5, 3],
          [7, 8, 6]])
```

```
In [214... ttt.sort(axis=1)
```

```
In [215... ttt
```

```
Out[215... array([[1, 2, 2],
          [3, 4, 5],
          [6, 7, 8]])
```

```
In [216... ttt.sort(axis=-1)
```

```
In [217... ttt
```

```
Out[217... array([[1, 2, 2],
          [3, 4, 5],
          [6, 7, 8]])
```

In [ ]:

In [ ]: