dizon_ipythonexercise_part4.2

February 12, 2015

Simple Arrays 1-dimensional array:

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

In [2]: a = np.array([1,2,3,4,5])
        a

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

In [3]: aa = np.array([7,6,5,4,3])
        aa

Out[3]: array([7, 6, 5, 4, 3])
```

2-dimensional array:

```
In [5]: b = np.array([[0,1,2],[3,4,5]])
        b

Out[5]: array([[0, 1, 2],
               [3, 4, 5]])

In [6]: bb = np.array([[9,8,7],[6,5,4]])
        bb

Out[6]: array([[9, 8, 7],
               [6, 5, 4]])

In [7]: len(bb)

Out[7]: 2

In [8]: len(a)

Out[8]: 5

In [10]: bb.shape

Out[10]: (2, 3)

In [11]: aa.ndim

Out[11]: 1

In [12]: c = np.arange(20)
         c
```

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

In [14]: d = np.arange(1,21,3)
         d

Out[14]: array([ 1,  4,  7, 10, 13, 16, 19])

In [16]: e = np.linspace(0,1,6)
         e

Out[16]: array([ 0. ,  0.2,  0.4,  0.6,  0.8,  1. ])

In [17]: f = np.linspace (0,1,5,endpoint=False)
         f

Out[17]: array([ 0. ,  0.2,  0.4,  0.6,  0.8])

In [19]: g = np.ones((3,3))
         g

Out[19]: array([[ 1.,  1.,  1.],
                [ 1.,  1.,  1.],
                [ 1.,  1.,  1.]])

In [20]: h = np.zeros((3,3))
         h

Out[20]: array([[ 0.,  0.,  0.],
                [ 0.,  0.,  0.],
                [ 0.,  0.,  0.]])

In [21]: i = np.eye(4)
         i

Out[21]: array([[ 1.,  0.,  0.,  0.],
                [ 0.,  1.,  0.,  0.],
                [ 0.,  0.,  1.,  0.],
                [ 0.,  0.,  0.,  1.]])
```

Creating arrays with random numbers

```
In [22]: j = np.diag(np.array([1,2,3,4,5]))
         j

Out[22]: 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, 5]])

In [23]: a = np.random.rand(5)
         a

Out[23]: array([ 0.69886585,  0.03206655,  0.89755273,  0.60252693,  0.3620176 ])

In [24]: b = np.random.rand(5,1)
         b
```

```
Out[24]: array([[ 0.43755986],
                [ 0.36614097],
                [ 0.5723815 ],
                [ 0.48360474],
                [ 0.06225226]])
```

np.empty creates garbage values; it is faster than utilizing zeros for your array's initial values.

```
In [25]: %pylab inline

Populating the interactive namespace from numpy and matplotlib

WARNING: pylab import has clobbered these variables: ['e', 'f']
`%matplotlib` prevents importing * from pylab and numpy

In [26]: import matplotlib.pyplot as plt

In [27]: x = np.linspace (0,3,20)
         y = np.linspace (0,9,20)
         plt.plot(x,y)

Out[27]: [<matplotlib.lines.Line2D at 0x39b6610>]
```

max size=0.90.9dizon$_i$*pythonexercise*$_p$*art*4.2$_f$*iles*/*dizon*$_i$*pythonexercise*$_p$*art*4.2$_2$5$_1$.*png*

```
In [28]: plt.plot(x,y,'o')

Out[28]: [<matplotlib.lines.Line2D at 0x3adba50>]
```

max size=0.90.9dizon$_i$*pythonexercise*$_p$*art*4.2$_f$*iles*/*dizon*$_i$*pythonexercise*$_p$*art*4.2$_2$6$_1$.*png*

```
In [29]: image = np.random.rand(30, 30)
         plt.imshow(image, cmap=plt.cm.gray)
         plt.colorbar()

Out[29]: <matplotlib.colorbar.Colorbar instance at 0x3abb8c0>
```

max size=0.90.9dizon$_i$*pythonexercise*$_p$*art*4.2$_f$*iles*/*dizon*$_i$*pythonexercise*$_p$*art*4.2$_2$7$_1$.*png*

Indexing and Slicing

```
In [31]: a = np.arange(30)
         a

Out[31]: 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, 24, 25, 26, 27, 28, 29])

In [32]: a[4:25:3]

Out[32]: array([ 4,  7, 10, 13, 16, 19, 22])

In [33]: a[:6]

Out[33]: array([0, 1, 2, 3, 4, 5])

In [34]: a[2:5]
```

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

In [35]: a[::6]

Out[35]: array([ 0,  6, 12, 18, 24])

In [36]: a[7:]

Out[36]: array([ 7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,
                24, 25, 26, 27, 28, 29])

In [38]: c = np.arange(6) + np.arange(0,51,10) [:,np.newaxis]
         c

Out[38]: array([[ 0,  1,  2,  3,  4,  5],
                [10, 11, 12, 13, 14, 15],
                [20, 21, 22, 23, 24, 25],
                [30, 31, 32, 33, 34, 35],
                [40, 41, 42, 43, 44, 45],
                [50, 51, 52, 53, 54, 55]])

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

Out[39]: array([[1, 2, 3],
                [4, 5, 6]])

In [40]: b = np.arange(6)
         c = np.arange(0,51,10)
         d = np.array([[b],[c]])
         d

Out[40]: array([[[ 0,  1,  2,  3,  4,  5]],

                [[ 0, 10, 20, 30, 40, 50]]])

In [41]: a = d
         a

Out[41]: array([[[ 0,  1,  2,  3,  4,  5]],

                [[ 0, 10, 20, 30, 40, 50]]])

In [42]: y = np.arange(10)
         y

Out[42]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [45]: z = np.arange(5)
         z[5:] = x[::-2]
         z


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

         ValueError                                Traceback (most recent call last)

         <ipython-input-45-fd0b44f2f121> in <module>()
```

```
      1 z = np.arange(5)
----> 2 z[5:] = x[::-2]
      3 z


ValueError: operands could not be broadcast together with shapes (0) (10)
```

Using a step of -2 in the reversal idiom will raise a ValueError, as shown above.
Array Creation

```
In [47]: a = np.array([[1,1,1,1], [1,1,1,1], [1,1,1,2], [1,6,1,1]])
         a

Out[47]: array([[1, 1, 1, 1],
                [1, 1, 1, 1],
                [1, 1, 1, 2],
                [1, 6, 1, 1]])

In [48]: b = np.array([[0.,0., 0., 0., 0.], [2., 0., 0., 0., 0.], [0., 3., 0., 0., 0], [0., 0., 4., 0.,
         b

Out[48]: array([[ 0.,  0.,  0.,  0.,  0.],
                [ 2.,  0.,  0.,  0.,  0.],
                [ 0.,  3.,  0.,  0.,  0.],
                [ 0.,  0.,  4.,  0.,  0.],
                [ 0.,  0.,  0.,  5.,  0.],
                [ 0.,  0.,  0.,  0.,  6.]])
```

Tiling for array creation

```
In [49]: a = np.array([[4,3], [2,1]])
         np.tile(a,(2,3))

Out[49]: array([[4, 3, 4, 3, 4, 3],
                [2, 1, 2, 1, 2, 1],
                [4, 3, 4, 3, 4, 3],
                [2, 1, 2, 1, 2, 1]])
```

Fancy indexing

```
In [50]: a = np.arange(10)
         a

Out[50]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [51]: a[[2,8,4]] = np.array([500,600,700])
         a

Out[51]: array([  0,   1, 500,   3, 700,   5,   6,   7, 600,   9])

In [ ]:
```