Kato_ipythonexercise_part4.2

February 12, 2015

1 Answers for IPython Exercise Part 4.2

1.0.1 Simple Arrays

1-D Array

```
In [2]: import numpy as np
```

```
In [3]: onedarray = np.array([11, 12, 13, 14, 15])
        onedarray
```

```
Out[3]: array([11, 12, 13, 14, 15])
```

2-D Array:

```
In [4]: twodarray = np.array([[10,9,8], [7,6,5]])
        twodarray
```

```
Out[4]: array([[10,  9,  8],
               [ 7,  6,  5]])
```

```
In [5]: len(onedarray)
```

```
Out[5]: 5
```

```
In [6]: twodarray.shape
```

```
Out[6]: (2, 3)
```

```
In [7]: onedarray.ndim
```

```
Out[7]: 1
```

1.0.2 Creating Arrays Using Functions

```
In [8]: a = np.arange(10)
        a
```

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

```
In [9]: b = np.arange(1,9,2)
        b
```

```
Out[9]: array([1, 3, 5, 7])
```

```
In [11]: c = np.linspace(0,1,6)
         c
```

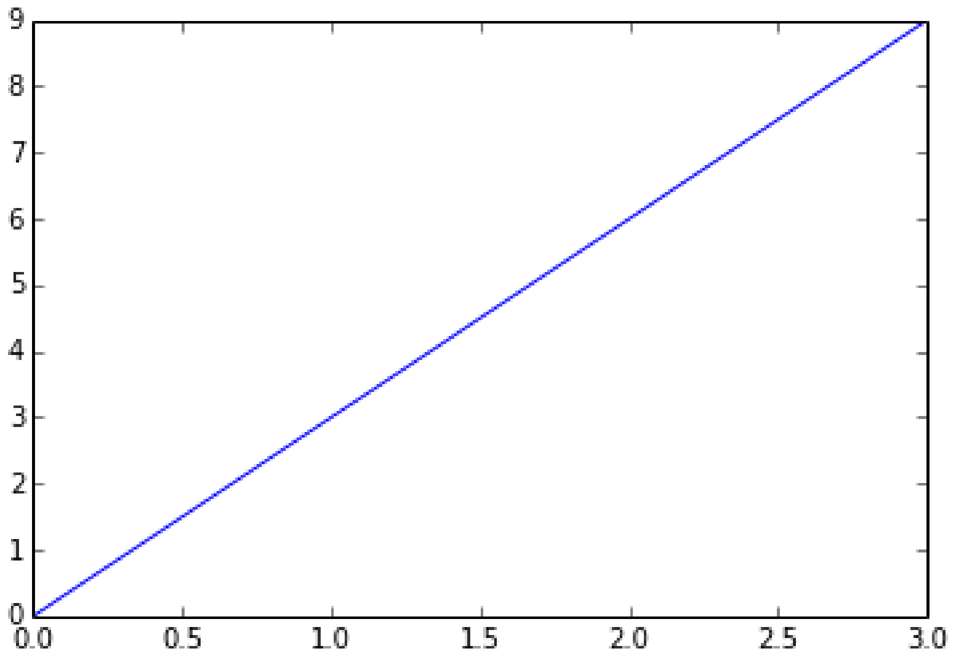
```
Out[11]: array([ 0. ,  0.2,  0.4,  0.6,  0.8,  1. ])

In [13]: d = np.linspace(0, 1, 5, endpoint=False)
         d

Out[13]: array([ 0. ,  0.2,  0.4,  0.6,  0.8])

In [14]: a = np.ones((3,3))
         a

Out[14]: array([[ 1.,  1.,  1.],
                [ 1.,  1.,  1.],
                [ 1.,  1.,  1.]])

In [15]: b = np.zeros((2,2))
         b

Out[15]: array([[ 0.,  0.],
                [ 0.,  0.]])

In [16]: c = np.eye(3)
         c

Out[16]: array([[ 1.,  0.,  0.],
                [ 0.,  1.,  0.],
                [ 0.,  0.,  1.]])

In [17]: d = np.diag(np.array([1,2,3,4]))
         d

Out[17]: array([[1, 0, 0, 0],
                [0, 2, 0, 0],
                [0, 0, 3, 0],
                [0, 0, 0, 4]])

In [18]: a = np.random.rand(4)
         a

Out[18]: array([ 0.86748968,  0.29152335,  0.52104699,  0.99721789])

In [19]: b = np.random.rand(4)
         b

Out[19]: array([ 0.07411529,  0.22565108,  0.64906644,  0.22481595])
```

np.empty creates a "garbahe"(uninitialized values). It is useful because unlike zeros, it does not set the array values to zero and may therefore be marginally faster.

1.0.3 Simple Visualizations

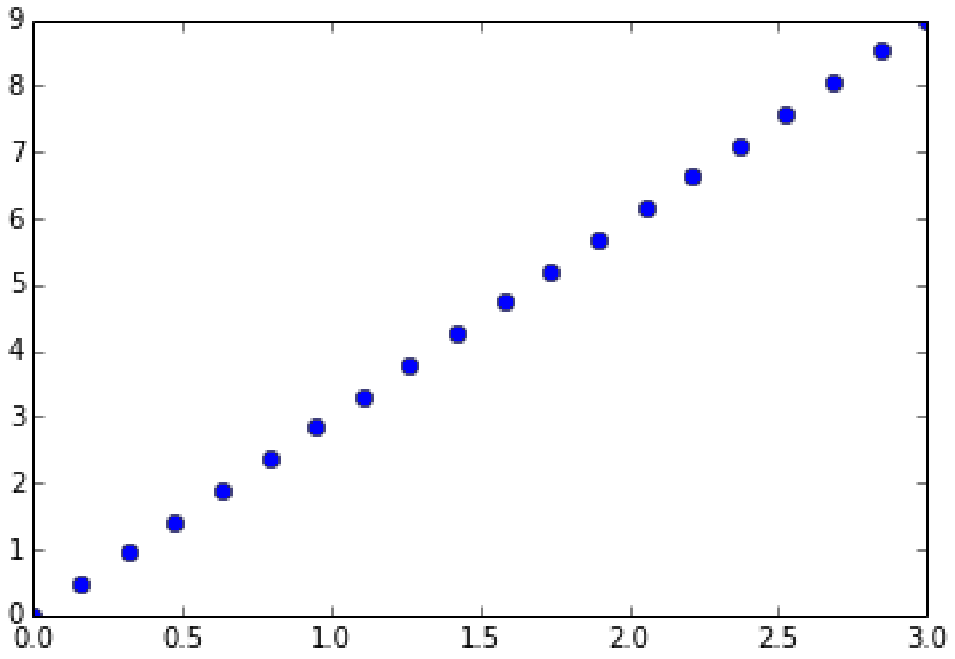
```
In [20]: %pylab inline

Populating the interactive namespace from numpy and matplotlib

In [21]: import matplotlib.pyplot as plt

In [24]: x = np.linspace(0, 3, 20)
         y = np.linspace(0, 9, 20)
         plt.plot(x, y)
```

```
Out[24]: [<matplotlib.lines.Line2D at 0xb15de24c>]
```

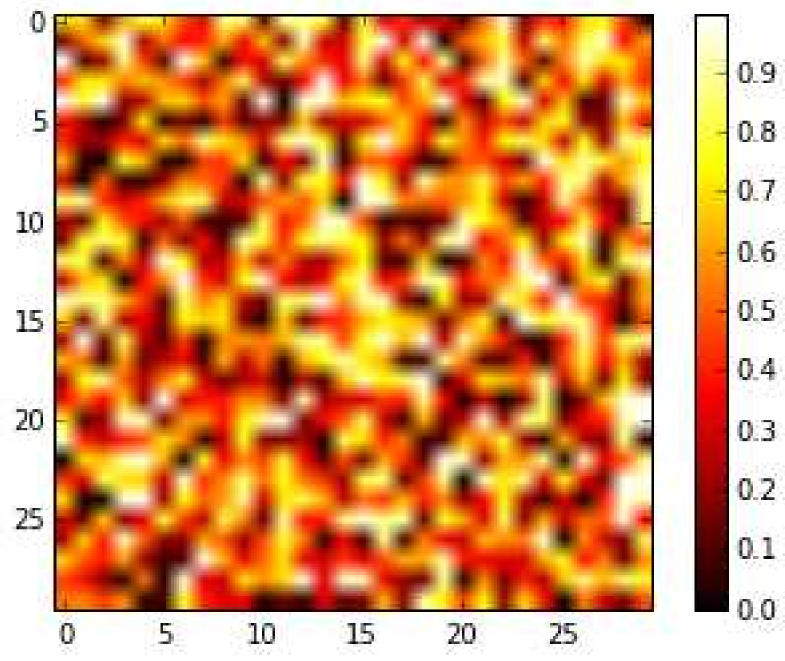


```
In [25]: plt.plot (x, y, 'o')
```

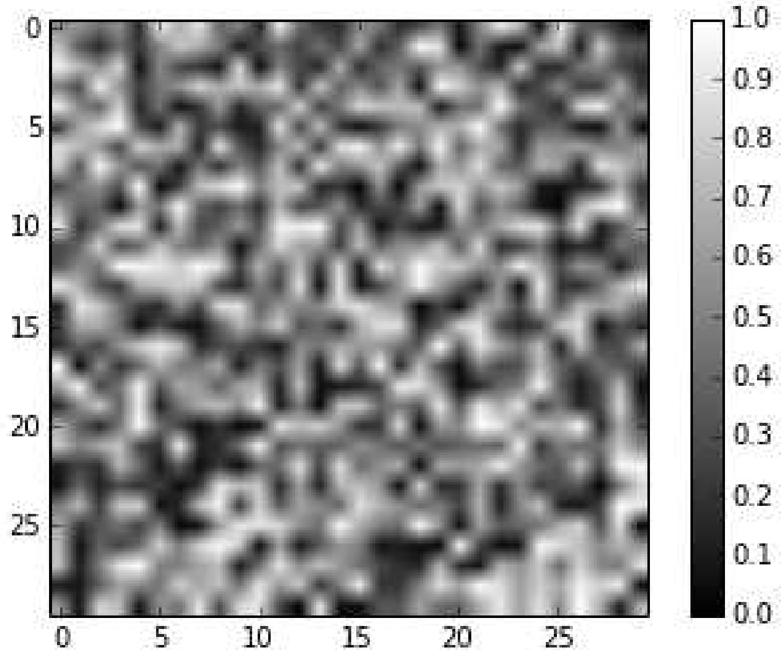
```
Out[25]: [<matplotlib.lines.Line2D at 0xb15952cc>]
```



```
In [26]: image = np.random.rand(30, 30)
         plt.imshow(image, cmap=plt.cm.hot)
         plt.colorbar()

Out[26]: <matplotlib.colorbar.Colorbar instance at 0xb144bfcc>
```



```
In [27]: image = np.random.rand(30, 30)
         plt.imshow(image, cmap=plt.cm.gray)
         plt.colorbar()

Out[27]: <matplotlib.colorbar.Colorbar instance at 0xafb7e80c>
```



1.0.4 Indexing and Slicing

```
In [28]: a = np.arange(20)
         a

Out[28]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
                17, 18, 19])

In [30]: a[4:20:3] #[start:end:step]

Out[30]: array([ 4,  7, 10, 13, 16, 19])

In [32]: a[:7] #end

Out[32]: array([0, 1, 2, 3, 4, 5, 6])

In [34]: a[1:9] #start and end

Out[34]: array([1, 2, 3, 4, 5, 6, 7, 8])

In [35]: a[::2] #steps

Out[35]: array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])

In [36]: a[4:] #start

Out[36]: array([ 4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19])

In [46]: b = np.arange(6) + np.arange(0, 51, 10) [:,np.newaxis]#put the slicing syntax [start:end:step]
         b
```

```
Out[46]: 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 [81]: a = np.array([[1,2,3], [4,5,6]])
         a

Out[81]: array([[1, 2, 3],
               [4, 5, 6]])

In [82]: s1 = np.arange(6)
         s2 = np.arange(0, 51, 10)
         b = np.array([[s1], [s2]])
         b

Out[82]: array([[[ 0,  1,  2,  3,  4,  5]],

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

In [83]: a = b
         a

Out[83]: array([[[ 0,  1,  2,  3,  4,  5]],

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

In [3]: z = np.arange(10)
        z

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

In [86]: x = np.arange(5)
         z[5:] = x[::-1]
         z


        ---------------------------------------------------------------------------
    ValueError                                Traceback (most recent call last)

        <ipython-input-86-229169b7cdde> in <module>()
          1 x = np.arange(5)
    ----> 2 z[5:] = x[::-2]
          3 z


        ValueError: could not broadcast input array from shape (3) into shape (5)
```

If you use step -2 in the above reversal idiom it will raise a ValueError: could not broadcast input array from shape(3) into shape(5).

1.0.5 Array Creation

```
In [17]: a = np.array([[1,1,1,1], [1,1,1,1], [1,1,1,2], [1,6,1,1]])
         a

Out[17]: array([[1, 1, 1, 1],
                [1, 1, 1, 1],
                [1, 1, 1, 2],
                [1, 6, 1, 1]])
```

```
In [16]: b = np.array([[0.,0., 0., 0., 0.], [2., 0., 0., 0., 0.], [0., 3., 0., 0., 0], [0., 0., 4., 0.,
         b

Out[16]: 2.0
```

1.0.6 Tiling For Array Creation

```
In [88]: a = np.array([[4,3], [2,1]])
         np.tile(a,(2,3))

Out[88]: 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]])
```

1.0.7 Fancy Indexing

```
In [89]: from IPython.display import Image
         Image(filename='images/numpy_fancy_indexing.png')
```

```
        ---------------------------------------------------------------------------
    IOError                                   Traceback (most recent call last)

        <ipython-input-89-beb616f120d9> in <module>()
          1 from IPython.display import Image
    ----> 2 Image(filename='images/numpy_fancy_indexing.png')


        /usr/lib/python2.7/dist-packages/IPython/core/display.pyc in __init__(self, data, url, filename,
        599         self.height = height
        600         self.retina = retina
    --> 601         super(Image, self).__init__(data=data, url=url, filename=filename)
        602 
        603         if retina:


        /usr/lib/python2.7/dist-packages/IPython/core/display.pyc in __init__(self, data, url, filename)
        303         self.filename = None if filename is None else unicode(filename)
        304 
    --> 305         self.reload()
        306 
        307     def reload(self):


        /usr/lib/python2.7/dist-packages/IPython/core/display.pyc in reload(self)
        621         """Reload the raw data from file or URL."""
        622         if self.embed:
    --> 623             super(Image,self).reload()
        624             if self.retina:
        625                 self._retina_shape()


        /usr/lib/python2.7/dist-packages/IPython/core/display.pyc in reload(self)
        308         """Reload the raw data from file or URL."""
        309         if self.filename is not None:
    --> 310             with open(self.filename, self._read_flags) as f:
        311                 self.data = f.read()
        312         elif self.url is not None:


        IOError: [Errno 2] No such file or directory: u'images/numpy_fancy_indexing.png'
```

```
In [102]: a = np.arange(10)
          a

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

In [104]: a[[9, 7, 3]] = np.array([1000, 2000, 3000])
          a

Out[104]: array([   0,    1,    2, 3000,    4,    5,    6, 2000,    8, 1000])
```