Data Structures and Algorithms with Python - Tutorial 4

September 25, 2016

This workshop will introduce to scientific computing in Python using the versatile `numpy` and `pandas` modules. You'll also learn how to present data in plots, and to apply some basic data analysis tools. Libraries such as these provide many more tools than we're able to cover here - essentially, for most things you can do in specialized languages such as R or MATLAB, you can also find a Python implementation.

But before going into scientific computing, let us introduce another important way to interact with Python.

1 Jupyter Notebooks

So far you've been using the Python console and the Spyder IDE to write code. Now you'll learn another convenient way to both write code and present it: the Jupyter Notebook. Briefly, the notebook is an interactive computing environment that allows you to include "live" Python code that the user can run, combined with explanations, narrative text, equations, etc. Notebooks can also easily be exported to html and PDF formats. Due to this flexibility, they have become popular in various contexts: for example, they are often used in analytics teams to share and present ideas. Indeed, this document and all the other tutorials are Jupyter Notebooks.

The main visible component of the notebook is a web application, which is your interface with the notebook. It allows you to both create and manage documents and write and run Python code. All of this happens in your browser. The web application uses a process called Python kernel to run your code - essentially there is an IPython shell running in the background. The documents have the extension `.ipynb`.

The Jupyter notebook is actually not limited to Python, but can be used with other languages including R using the corresponding kernel.

Jupyter Notebook comes with your Anaconda distribution. In Windows, you should be able to find it by tapping the Windows key and starting to type Jupyter. Opening the program starts both a log window and a new tab in your web browser.

The browser window gives you a folder view, by default of the directory `C:\Users\USERNAME`. You can change this directory later, but for now download the notebook file for this tutorial and place it within the directory. Navigate to the file in the jupyter browser window, and open it. You'll see this same text, but later also snippets of Python code that you can run and edit. Indeed, you can also edit the entire document to make comments etc. You can also try creating your own documents - it is a valuable tool to know.

A notebook is organized in cells. Some cells contain text and other code You can run a cell containing Python code by selecting it and pressing the "play" button in the toolbar or using `Shift + Enter`. To edit a cell's contents, double-click on it.

From now on, we'll provide the tutorials as both notebook files and PDFs. You may choose whether to work in the notebook environment or continue working on Spyder. Both are useful coding environments to learn.

2 Scientific computing with numpy

2.1 Why numpy

Consider the following simple matrix:

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

You could implement it in Python as a list of lists:

```
>>> m = [[1, 2, 3],
...      [4, 5, 6],
...      [7, 8, 9]]
>>> m
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Imagine you want to add 2 to every element. Ideally, we would want to do something like `m + 2`. However, if you try you will get this:

```
>>> m + 2
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-d5811c0a075a> in <module>()
----> 1 m + 2

TypeError: can only concatenate list (not "int") to list
```

You can do it but it becomes more involved, not very readable and not very general either:

```
>>> [[i + 2 for i in j] for j in m]
[[3, 4, 5], [6, 7, 8], [9, 10, 11]]
```

That is where Numpy comes in. Numpy is a library providing a powerful representation for fast manipulation of multi dimensional arrays. In particular it is particularly appropriate for scientific computing.

In addition to being much more adapted, it is also much faster:

```
>>> import numpy as np
>>> a = np.array([1, 2, 3, 4])
>>> a
a = np.array([1, 2, 3, 4])
```

```
In [1]: %timeit l = [i**2 for i in range(10000)]
100 loops, best of 3: 4.43 ms per loop

In [2]: %timeit l = np.arange(10000)**2
100000 loops, best of 3: 14.4 µs per loop
```

3 Basic operations

First import numpy. The convention is to import as follows:

```
>>> import numpy as np
```

This means that everytime you want to use a function in numpy you access it as `np.function`. For instance you create a numpy array as:

```
>>> np.array([1, 2, 3])
array([1, 2, 3])
```

The Scipy lectures notes are a great source of information for numpy and scientific computing with Python in general and this tutorial is based on these.

The most obvious way to create a numpy array is using a list with the syntax `np.array(l)` where l is your list.

3.1 1-D array

```
In [27]: a = np.array([1, 2, 3])
```

You can check the number of dimension of an array

```
In [28]: a.ndim

Out[28]: 1
```

The shape gives you the size of each dimension (in this case one dimension of size 3)

```
In [29]: a.shape

Out[29]: (3,)
```

3.2 2-D array

The syntax for 2D arrays in natural and similar to that of nested lists.

```
In [30]: a = np.array([[1, 2, 3],
                       [4, 5, 6],
                       [7, 8, 9]])

In [31]: a

Out[31]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])
```

We created a matrix, naturally, it has two dimensions:

```
In [32]: a.ndim
```

```
Out[32]: 2
```

Each of its dimension have size 3

```
In [33]: a.shape

Out[33]: (3, 3)

In [34]: b = np.array([[1, 2, 3],
                       [4, 5, 6],
                       [7, 8, 9],
                       [10, 11, 12]])

In [35]: b.ndim

Out[35]: 2

In [36]: b.shape

Out[36]: (4, 3)

In [38]: len(b) # size of the first dimension

Out[38]: 4
```

In practice we rarily define arrays by hand. There are several convenience functions to create arrays in numpy:

The equivalent of `range`:

```
In [39]: np.arange(10)

Out[39]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Equally spaced numbers:

```
In [40]: np.linspace(2, 20, 10)

Out[40]: array([  2.,   4.,   6.,   8.,  10.,  12.,  14.,  16.,  18.,  20.])
```

Identity matrix:

```
In [41]: np.eye(4)

Out[41]: array([[ 1.,  0.,  0.,  0.],
                [ 0.,  1.,  0.,  0.],
                [ 0.,  0.,  1.,  0.],
                [ 0.,  0.,  0.,  1.]])

In [42]: np.zeros((2, 2))

Out[42]: array([[ 0.,  0.],
                [ 0.,  0.]])
```

```
In [43]: np.ones((3, 4))

Out[43]: array([[ 1.,  1.,  1.,  1.],
                [ 1.,  1.,  1.,  1.],
                [ 1.,  1.,  1.,  1.]])
```

Random array:

```
In [99]: np.random.rand(2, 3)

Out[99]: array([[ 0.76495025,  0.67881293,  0.87448005],
                [ 0.74538693,  0.71711443,  0.78492205]])
```

3.3 Operations on numpy arrays

```
In [44]: a = np.array([[1, 2, 3],
                       [4, 5, 6],
                       [7, 8, 9]])
```

Most operations are by default **element-wise**, i.e. they are applied to each element of the array:

```
In [46]: a + 2

Out[46]: array([[ 3,  4,  5],
                [ 6,  7,  8],
                [ 9, 10, 11]])

In [47]: a * 3

Out[47]: array([[ 3,  6,  9],
                [12, 15, 18],
                [21, 24, 27]])

In [48]: a**2

Out[48]: array([[ 1,  4,  9],
                [16, 25, 36],
                [49, 64, 81]])

In [49]: a/2

Out[49]: array([[ 0.5,  1. ,  1.5],
                [ 2. ,  2.5,  3. ],
                [ 3.5,  4. ,  4.5]])

In [53]: a+a # same as 2*a

Out[53]: array([[ 2,  4,  6],
                [ 8, 10, 12],
                [14, 16, 18]])
```

```
In [55]: a*a # same as a**2

Out[55]: array([[ 1,  4,  9],
                [16, 25, 36],
                [49, 64, 81]])
```

Numpy even comes in with more complex functions:

```
In [68]: np.cos(a)

Out[68]: array([[ 0.54030231, -0.41614684, -0.9899925 ],
                [-0.65364362,  0.28366219,  0.96017029],
                [ 0.75390225, -0.14550003, -0.91113026]])

In [69]: np.exp(a)

Out[69]: array([[  2.71828183e+00,   7.38905610e+00,   2.00855369e+01],
                [  5.45981500e+01,   1.48413159e+02,   4.03428793e+02],
                [  1.09663316e+03,   2.98095799e+03,   8.10308393e+03]])

In [70]: np.exp([1, 0])

Out[70]: array([ 2.71828183,  1.        ])

In [73]: np.cos([0, 1, 3.1415])

Out[73]: array([ 1.        ,  0.54030231, -1.        ])
```

3.4 Types

Numpy arrays can be of different types:

```
In [61]: np.array(['a', 'b', 'c'])

Out[61]: array(['a', 'b', 'c'],
               dtype='<U1')
```

... but you cannot mix any type of object like you would do in Python:

```
In [62]: np.array([1, [1], 2])

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

<ipython-input-62-50dde5b36add> in <module>()
----> 1 np.array([1, [1], 2])


ValueError: setting an array element with a sequence.
```

3.5 Indexing and slicing

You can keep only part of an array with the same syntax as lists (array[begin:end:step]):

```
In [100]: a = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

In [101]: a[:5] # all elements up to the fifth

Out[101]: array([1, 2, 3, 4, 5])

In [102]: a[-1] # Last element@

Out[102]: 10

In [103]: a[5:]

Out[103]: array([ 6,  7,  8,  9, 10])

In [105]: a[2:-2]

Out[105]: array([3, 4, 5, 6, 7, 8])

In [106]: a[::2]

Out[106]: array([1, 3, 5, 7, 9])
```

You can also have a negative step in which case the element are read from the end:

```
In [107]: a[::-1]

Out[107]: array([10,  9,  8,  7,  6,  5,  4,  3,  2,  1])
```

3.6 Reshaping

You can **reshape** an array by reading the elements differently: For instance, if you read a vector of 12 elements by batches of three, you get a matrix of size 4 by 3:

```
In [162]: a = np.arange(12)

In [163]: a.reshape((4, 3))

Out[163]: array([[ 0,  1,  2],
                 [ 3,  4,  5],
                 [ 6,  7,  8],
                 [ 9, 10, 11]])

In [164]: a.reshape((3, 4))

Out[164]: array([[ 0,  1,  2,  3],
                 [ 4,  5,  6,  7],
                 [ 8,  9, 10, 11]])
```

You can transpose an array. I assume you already know the transpose operator, at least for vectors and matrices:

```
In [165]: a.reshape((4, 3)).T

Out[165]: array([[ 0,  3,  6,  9],
                 [ 1,  4,  7, 10],
                 [ 2,  5,  8, 11]])
```

4 Plotting

First import the plotting library, `matplotlib`:

```
In [64]: import matplotlib.pyplot as plt
```

If you are in the notebook, you want to specify that you want the plots to appear in the notebook:

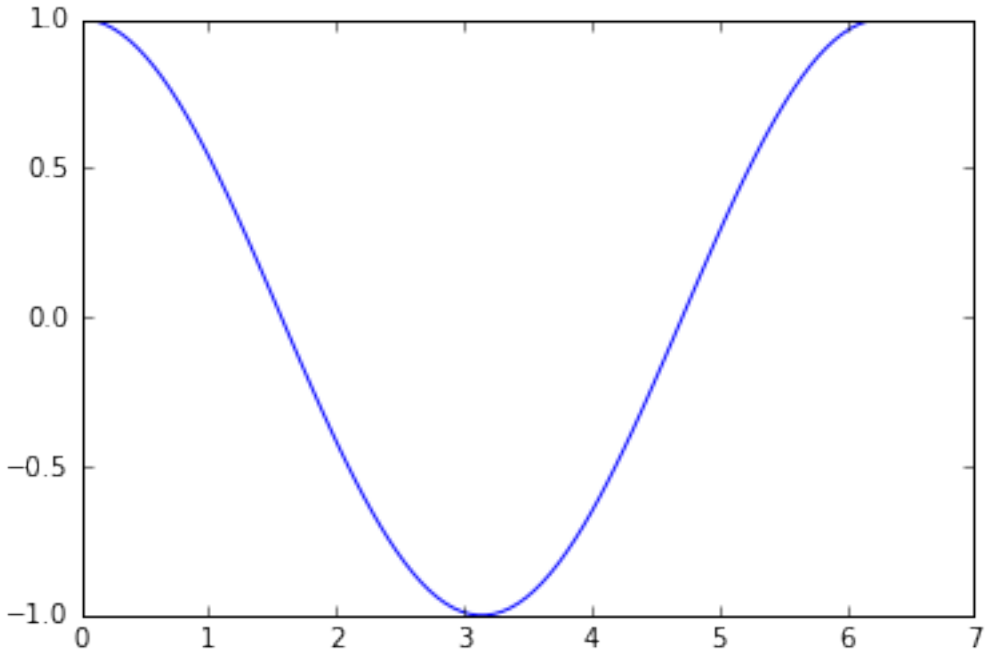
```
In [65]: %matplotlib inline
```

```
In [75]: x = np.linspace(0, 2*3.1415, 100)
```

```
In [76]: y = np.cos(x)
```

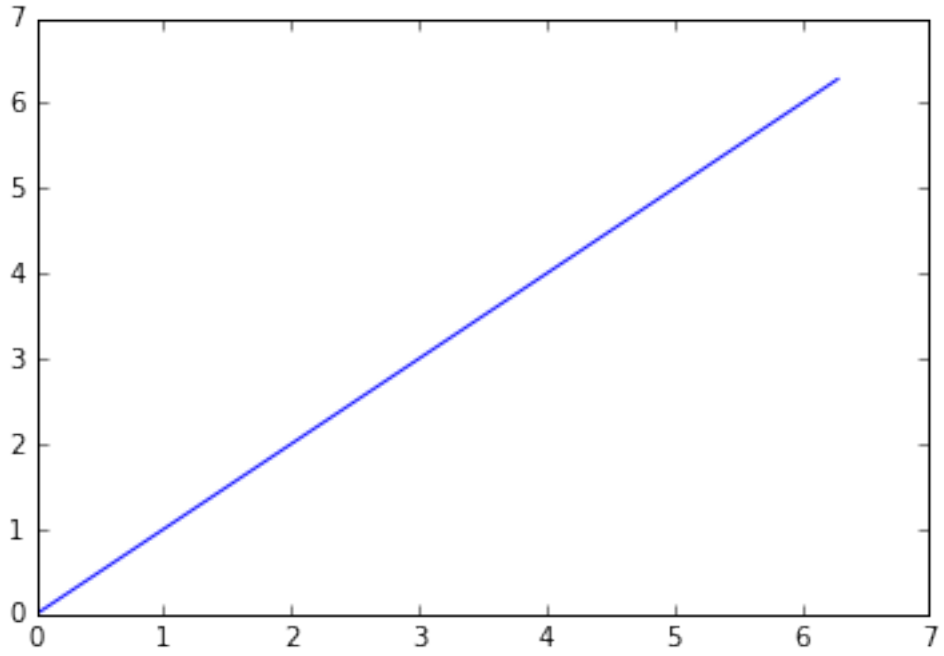
```
In [77]: plt.plot(x, y)
```

```
Out[77]: [<matplotlib.lines.Line2D at 0x10fbbdda0>]
```



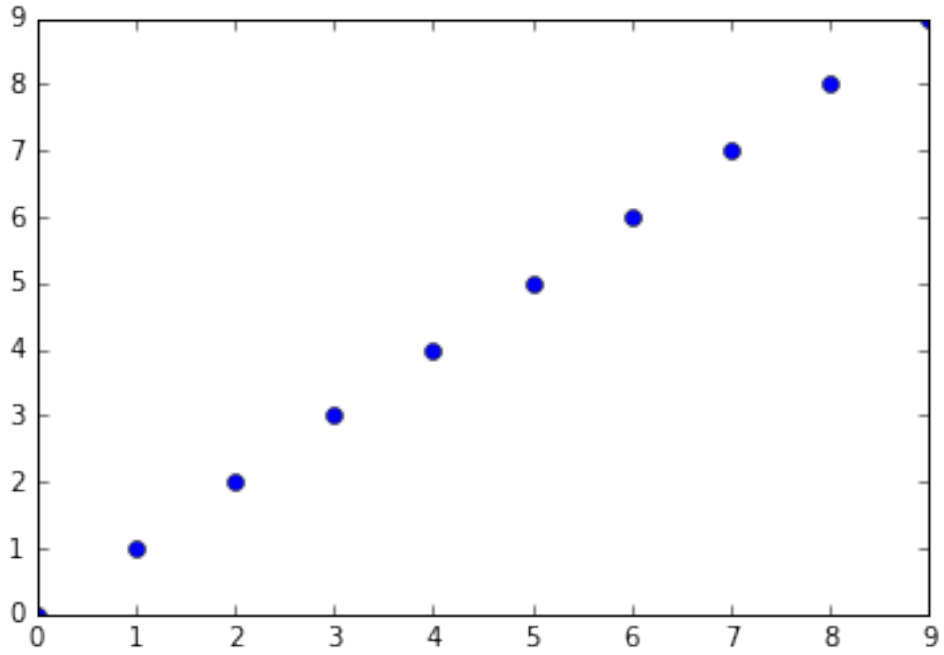

```
In [78]: plt.plot(x, x)
```

```
Out[78]: [<matplotlib.lines.Line2D at 0x10fcf8358>]
```




```
In [80]: x = np.arange(10)
         plt.plot(x, x, 'o')
```

```
Out[80]: [<matplotlib.lines.Line2D at 0x10ff65e80>]
```



4.1 2-D plots

You can also plot images (an image is nothing more than a 2D array for a gray image or a 3D array for, say, an RGB image).

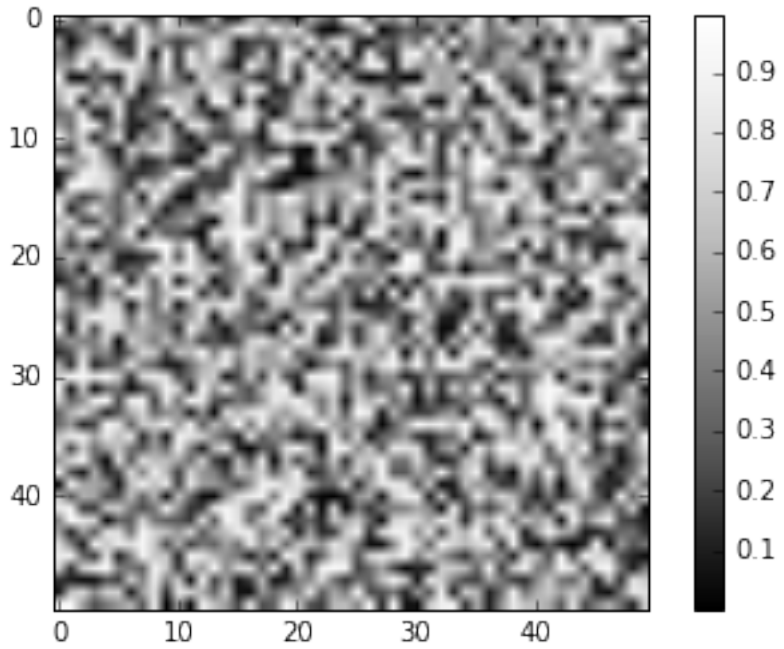
Let us create an "image" with random pixels:

```
In [97]: im = np.random.rand(50, 50)
```

Let's now display our "image":

```
In [98]: plt.imshow(im, cmap=plt.cm.Greys_r)
         plt.colorbar()

Out[98]: <matplotlib.colorbar.Colorbar at 0x1112ac898>
```



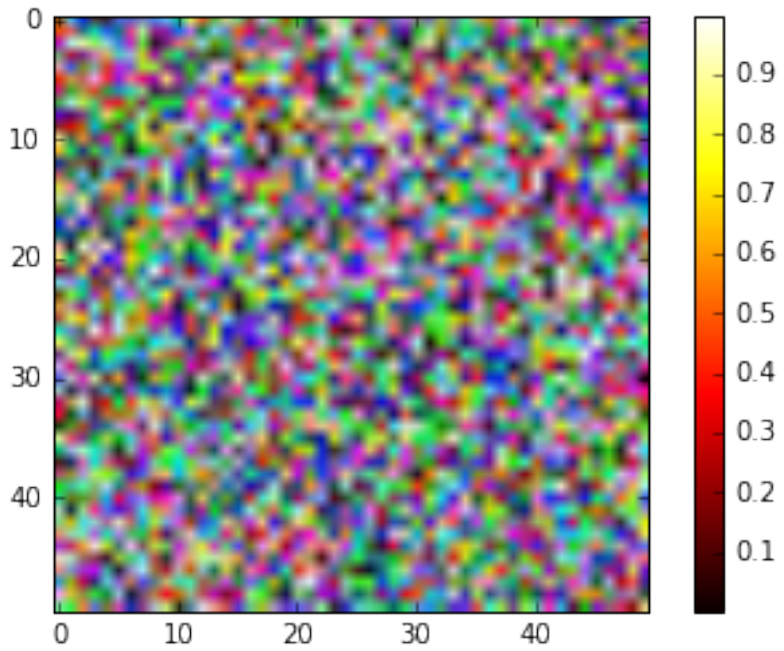
4.2 3D plots

Let's create an 'RGB image':

```
In [95]: im = np.random.rand(50, 50, 3)

In [96]: plt.imshow(im, cmap=plt.cm.hot)
         plt.colorbar()

Out[96]: <matplotlib.colorbar.Colorbar at 0x111030a90>
```



5 Pandas

Pandas builds on top of numpy and matplotlib and provides a data analysis and statistical package in Python.

You can read data easily from pandas. For instance to load data from a ".csv" file:

5.0.1 Getting some data

First let us download an excel file from a famous statistical dataset, the adult dataset, that contains different characteristics of individuals. The goal is to use these to predict if their salary is lower or higher than 50K.

```
In [1]: import requests

        def save_from_url(url, filename):
            response = requests.get(url, stream=True)
            if response.status_code == 200:
                with open(filename, 'wb') as f:
                    for chunk in response:
                        f.write(chunk)

In [2]: url = 'http://www.openmarkov.org/learning/datasets/adult.csv'

In [3]: save_from_url(url, 'adult.csv')
```

5.0.2 Exploring with Pandas

In this part, we will use the Titanic dataset from the Kaggle Getting Started challenge at:
https://www.kaggle.com/c/titanic-gettingStarted
Either log on to the website and dowload the data or get from here:
https://dl.dropboxusercontent.com/u/2140486/data/titanic_train.csv
If you are using linux, you can load the CSV file as a pandas data frame in one line:

```
In [5]: !curl -s https://dl.dropboxusercontent.com/s/2qhkb2wcqr78cp6/titanic_train.
        !curl -s https://dl.dropboxusercontent.com/u/2140486/data/adult_train.csv >
```

Import pandas

```
In [6]: import pandas as pd
```

Loading a dataset in pandas is as easy as Pie(thon):

```
In [15]: data = pd.read_csv('titanic.csv')

In [16]: data.shape

Out[16]: (891, 12)
```

Displaying the 5 first rows:

```
In [17]: data.head(5)

Out[17]:    PassengerId  Survived  Pclass  \
         0            1         0       3
         1            2         1       1
         2            3         1       3
         3            4         1       1
         4            5         0       3

                                                         Name     Sex   Age  SibSp
         0                            Braund, Mr. Owen Harris    male  22.0      1
         1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
         2                             Heikkinen, Miss. Laina  female  26.0      0
         3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
         4                           Allen, Mr. William Henry    male  35.0      0

            Parch            Ticket     Fare Cabin Embarked
         0      0         A/5 21171   7.2500   NaN        S
         1      0          PC 17599  71.2833   C85        C
         2      0  STON/O2. 3101282   7.9250   NaN        S
         3      0            113803  53.1000  C123        S
         4      0            373450   8.0500   NaN        S
```

If you are using the Ipython Notebook, you will get a nice HTML representation.
Get useful information

```
In [18]: data.describe()

/Users/jean/anaconda/lib/python3.5/site-packages/numpy/lib/function_base.py:3823: R
  RuntimeWarning)
```

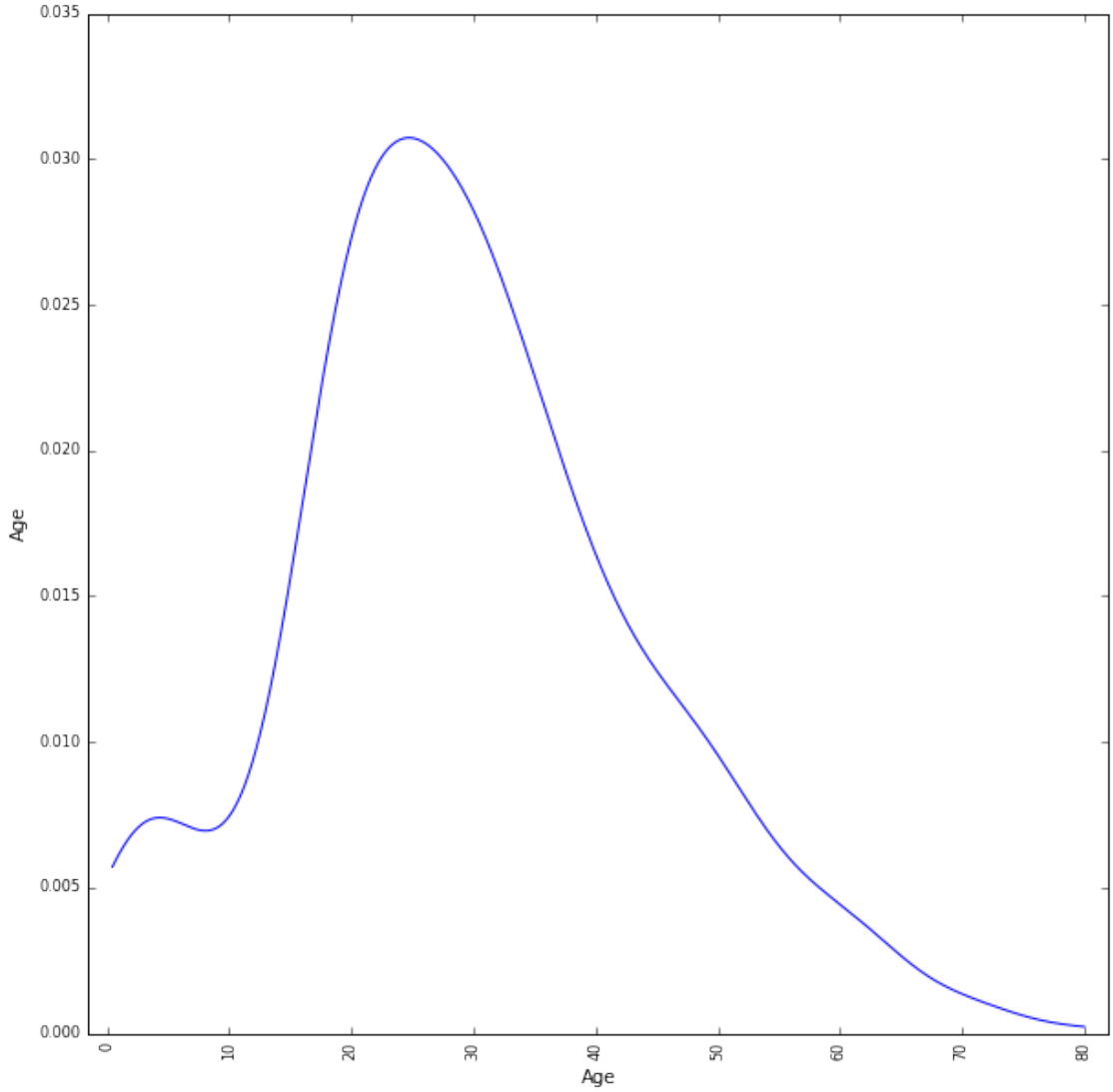
Out[18]:

	PassengerId	Survived	Pclass	Age	SibSp	\
count	891.000000	891.000000	891.000000	714.000000	891.000000	
mean	446.000000	0.383838	2.308642	29.699118	0.523008	
std	257.353842	0.486592	0.836071	14.526497	1.102743	
min	1.000000	0.000000	1.000000	0.420000	0.000000	
25%	223.500000	0.000000	2.000000	NaN	0.000000	
50%	446.000000	0.000000	3.000000	NaN	0.000000	
75%	668.500000	1.000000	3.000000	NaN	1.000000	
max	891.000000	1.000000	3.000000	80.000000	8.000000	

	Parch	Fare
count	891.000000	891.000000
mean	0.381594	32.204208
std	0.806057	49.693429
min	0.000000	0.000000
25%	0.000000	7.910400
50%	0.000000	14.454200
75%	0.000000	31.000000
max	6.000000	512.329200

```
In [24]: pd.tools.plotting.scatter_matrix(data.get(['Age']), alpha=0.2, figsize=(10

Out[24]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x119a487f0>]], d
```



5.0.3 Dropping and replacing variables

```
In [25]: data.head(5)
```

Out[25]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1
2	3	1	3				
3	4	1	1				
4	5	0	3				

```
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

Let's delete the first two rows (remember, indexing starts at 0!)

```
In [26]: data.drop([0, 1], axis=0).head(5)

Out[26]:    PassengerId  Survived  Pclass  \
2            3         1       3
3            4         1       1
4            5         0       3
5            6         0       3
6            7         0       1

                                          Name     Sex   Age  SibSp  Parc
2                       Heikkinen, Miss. Laina  female  26.0      0
3  Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                     Allen, Mr. William Henry    male  35.0      0
5                             Moran, Mr. James    male   NaN      0
6                      McCarthy, Mr. Timothy J    male  54.0      0

             Ticket     Fare Cabin Embarked
2  STON/O2. 3101282   7.9250   NaN        S
3            113803  53.1000  C123        S
4            373450   8.0500   NaN        S
5            330877   8.4583   NaN        Q
6             17463  51.8625   E46        S
```

You can also delete whole columns:

```
In [27]: data.drop(['PassengerId', 'Survived'], axis=1).head(5)

Out[27]:    Pclass                                                Name     Sex   Age
0       3                             Braund, Mr. Owen Harris    male  22.0
1       1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0
2       3                             Heikkinen, Miss. Laina  female  26.0
3       1       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0
4       3                           Allen, Mr. William Henry    male  35.0

   SibSp  Parch            Ticket     Fare Cabin Embarked
0      1      0         A/5 21171   7.2500   NaN        S
```

```
1      1      0           PC 17599  71.2833   C85        C
2      0      0  STON/O2. 3101282   7.9250   NaN        S
3      1      0            113803  53.1000  C123        S
4      0      0            373450   8.0500   NaN        S
```

note: by default, drop is not inplace: in other words, the function returns you a *copy* while the original is untouched: you can check that it hasn't been modified:

```
In [28]: data.head(5)

Out[28]:    PassengerId  Survived  Pclass  \
         0            1         0       3
         1            2         1       1
         2            3         1       3
         3            4         1       1
         4            5         0       3

                                                        Name     Sex   Age  SibSp
         0                            Braund, Mr. Owen Harris    male  22.0      1
         1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
         2                             Heikkinen, Miss. Laina  female  26.0      0
         3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
         4                           Allen, Mr. William Henry    male  35.0      0

            Parch            Ticket     Fare Cabin Embarked
         0      0         A/5 21171   7.2500   NaN        S
         1      0          PC 17599  71.2833   C85        C
         2      0  STON/O2. 3101282   7.9250   NaN        S
         3      0            113803  53.1000  C123        S
         4      0            373450   8.0500   NaN        S
```

To remove duplicates (there are none in this dataset...)

```
In [29]: aduplicates_free = data.drop_duplicates()
```

You can also replace values by new ones. For instance, strings, such as 'male' or 'female' are not really useful for statistical analysis. We usually prefere to replace them with numerical values:

```
In [30]: data.replace('male', 1).head(5)

Out[30]:    PassengerId  Survived  Pclass  \
         0            1         0       3
         1            2         1       1
         2            3         1       3
         3            4         1       1
         4            5         0       3

                                                        Name     Sex   Age  SibSp
         0                            Braund, Mr. Owen Harris       1  22.0      1
```

1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	
2	Heikkinen, Miss. Laina	female	26.0	0	
3	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	
4	Allen, Mr. William Henry	1	35.0	0	

	Parch	Ticket	Fare	Cabin	Embarked
0	0	A/5 21171	7.2500	NaN	S
1	0	PC 17599	71.2833	C85	C
2	0	STON/O2. 3101282	7.9250	NaN	S
3	0	113803	53.1000	C123	S
4	0	373450	8.0500	NaN	S

In [ ]: