Getting Started with Python, Anaconda, Jupyter, and Pandas

What are we even talking about?

- **Python** is a powerful and user-friendly programming language. Python's functionality is enhanced and extended by many useful *packages*. Python itself and almost all of its packages are free and open-source: anyone can download and modify them free of cost and without restrictions.
- **Anaconda** is the software you will download to install Python and the packages we will use. Anaconda "includes more than 400 of the most popular Python packages for science, math, engineering, and data analysis." It also includes `conda`, a *package manager* that makes it easy to install new add-ons.
- **Jupyter** (formerly called IPython) provides a "notebook" environment for programming in Python. Working in Jupyter *notebooks* enables you to easily organize, document, and share your code. Jupyter can also provide similar functionality for a wide range of other programming languages.
- **Pandas** is an essential package for Python data analysis. It enables us to store and manipulate data in *dataframes*, efficient and flexible structures that are like Excel sheets on steroids.

OK, how do I get these things?

The default Anaconda installation includes Python, Jupyter, and Pandas. Some programs are still written in Python 2.7, but for this tutorial we will use Python 3.5. This version represents the future of the language, and has some minor syntax differences compared with Python 2.7.

Visit https://www.continuum.io/downloads and download Anaconda for your operating system. Make sure to download the Python 3.5, 64-bit version.[1] The installer will take a little while to download; use the waiting time to learn your way around the command line (see below). When it's finished, open the file you downloaded and follow the on-screen instructions. The default settings should generally work fine.

I have Anaconda installed. Now how do I start coding?

There are many ways to write and run Python programs. We will be working within the Jupyter notebook environment. To get started, we need to launch Jupyter from the *command line*, within the folder containing your data.

- In Windows, you can access the command line by opening the Start menu and typing `cmd`. To open a command prompt in a specific folder, navigate to that folder in Windows, then click File > Open Command Prompt > Open Command Prompt.
- In Mac OS X, you can access the command line by opening the Terminal app (in Applications > Utilities). To navigate to your project folder, first type `cd ` (including the space) in the Terminal, then drag the desired folder from the Finder into the Terminal window, then hit Enter.

[1] If you're on a very old computer, there is a small chance you will instead need the 32-bit version. To check in Windows 10, go to Settings > System > About > System Type; all Macs released after 2010 are 64-bit.)

To learn more about working in a command line environment, check out http://tinyurl.com/commandline-windows or http://tinyurl.com/commandline-mac.

Before you get started coding, make sure key packages are up to date. Type, one line at a time:

```
conda install pandas
conda install jupyter
pip install geopy
```

Press Enter after typing each line. (The `geopy` package isn't available for installation through the `conda` package manager, so we use `pip` – an alternative package manager – instead.)

Once you have a command line open in your working folder, type `jupyter notebook` and hit Enter. The Jupyter interface will appear in your browser. In the upper right corner, click New > Python 3. This will create and open a new Jupyter notebook. Next to `In [ ]:`, type:

```
print('Hello world!')
```

Then click Cell > Run Cells.[2] Congratulations: you've just written and run a program!

Is everything set up correctly?

To make sure the installation was fully successful, let's *import* the packages we'll use for the boot camp. Importing packages is easy. First, let's create a new notebook *cell*. Click Insert > Insert Cell Below. In the new cell, type the following line:

```
import time
```

Then run the cell. You won't see any output, but you will now have access to the functionality in the `time` package, such as `time.sleep(5)`, which tells the program to wait for five seconds before continuing.

You can give a package a new name – an *alias* – when you import it. This can be a good way to save time by not having to type the full name. For example, in your code, you will frequently be referring to Pandas functionality, so an alias for `pandas` will save a lot of typing. By convention,[3] the `pandas` package is usually imported as `pd`:

```
import pandas as pd
```

You can also import a specific *module* from a given package. This lets you save typing by providing direct access to the functionality you care about. For instance, you can type:

```
from geopy.geocoders import GoogleV3
```

If the preceding two commands run without returning an `ImportError`, your installation is complete and you're ready for the boot camp!

---

[2] See Help > Keyboard Shortcuts for many time-saving key commands.

[3] Other conventions exist: `import numpy as np`, for instance.