A Python Tourist Guide

Omar A. Guerrero*

This document provides a short guide through the essentials for setting up the python programming language in your machine. Python is an invaluable research tool, not only because of its closely resemblance to human language, but also because of its wide adoption in the broader scientific community. This means that, in the increasingly interdisciplinary world that we live in, working with python will make a universe of analytical tools from different fields available to you in an instant. This document is part of the guide that I provide to my students when they incursion in the world of computationally-informed sciences. Therefore, I make it available to anyone interested in taking advantage of this tool.

This is not a tutorial on how to use python, but a guide that will refer you to the most relevant tools to learn fast and efficiently. Everything that I will point to in this document is referred to the original source, so you should think of this document as a *tourist guide*. For further information you should use the official python tutorial.

1 Version and Distribution

There are several versions of the Python programming language. I prefer working with version 2.7 since it has the widest support of the scientific community today. Python can be obtained in different ways or "distributions". A distribution is nothing more than the python language together with a bunch of packages (also known as libraries) that the distributor decides to put together. However, you are not limited to those packages because you can download and install other packages pretty easily.

In order to get you started, download the distribution known as Anaconda, which comes with python 2.7 and all the necessary libraries to keep you busy for a while.

*Institute for New Economic Thinking and Sad Business School, University of Oxford

2 IDE

IDE stands for Integrated Development Environment, and it is the application that allows you to program through a graphical interface. For example, the popular programs like Stata and Matlab are graphically integrated by default, so when you run them, you are automatically working in their IDEs.

Here, I will concentrate on two of my favorites IDEs for Python: Spyder and the IPython Notebooks. Both tools come already in the Anaconda distribution, so once you have installed it, fire up the *Launcher* application. The Anaconda Launcher gives you access to both IDEs, so you are free to choose between them. Here I will give you a brief description of their differences in order for you to decide which one is more suitable for your work.

2.1 Spyder

Spyder is very similar to the Matlab interface. If you are doing heavy-duty coding, I advise you to use this IDE since it allows you to keep track of your variables and data structures in a neat and organized manner. Another favorite IDE that is very similar to Spyder is the IEP IDE. Personally, I prefer IEP because it is very robust and fast, although it does not come with all the debugging utilities that Spyder has. Additionally, IEP does not come with Anaconda so you will need to install it separately.

2.2 IPython Notebooks

The IPython notebooks are executable documents. That is, you have a word-type of document that you can format, but with the advantage that you can embed blocks of code and execute it in order to print the output into the document. The advantage of this tool is that it allows other users to play with you code more easily, and keeps your project very neat. In fact, you can export an IPython notebook to a latex in order to generate nice PDF document out of it. This IDE is more suitable if you are not writing insane amounts of code and if your intention is to communicate your program more effectively.

3 Native Elements

The native elements are data types, structures, and functions that are provided by default in python instead of an external library. Python comes

with very useful native elements that will allow you to make a huge progress in a a few steps.

3.1 Data Types

Integers and floats are automatically managed by python. Just remember that if you divide two integers, **you will get an integer back**, which is a common source of error in early python programs. The other data type you should master are strings. Python is extremely good at handling strings. That is the reason why it has become the main tool for text analysis and related fields. Finally, you should know about Boolean types, which are binary variables taking either True or False values.

3.2 Data Structures

Like every programming language, python has several data structures that will help in your data processing. I advise you to get familiar with lists, dictionaries, and sets.

3.3 Control Flow

Programming 101: if-else statements and for-loops; both of these statements are vital in any language.

3.4 Functions

Python comes with many native functions. I recommend to start by mastering abs, all, any, enumerate, eval, filter, float, int, len, list, map, max, min, open, pow, range, reduce, round, set, slice, sorted, str, sum.

3.5 Extra

An extra tool that will come in handy is list comprehension. This is a compact way of constructing and evaluating lists that not only simplifies your code, but speeds it up considerably.

4 Libraries

Now that you are proficient in python, it is time to expand your horizons. There are tons of cool and useful libraries that will help you to perform

different types of analysis. I strongly advise to become familiar with a few: Matplotib, Numpy, Scipy, NetworkX, Scikit-Learn, csv, and pickle.

4.1 Matplotlib

Matplotlib is a plotting library. It is extremely intuitive, comprehensive, and flexible. After using Matplotlib, you will hardly seek to use anything else for your plots.

4.2 Numpy

Numpy is one of the best libraries available for Python. It is a scientific computing library of a similar caliber as Matlab. Numpy is a key library in the Python community since several scientific libraries use it. Its data structures and vectorized operations can speed up your code in orders of magnitude.

If you are familiar with Matlab, transitioning to Numpy is natural, for which I recommend this guide. If you are new to scientific computing, then I recommend studying the official Numpy tutorial. For the purpose of this project, I suggest you focus on sections 2 and 3.

4.3 Scipy

If you are working with me, you will most likely make extensive use of statistical tools. For this purpose, you should learn all about Scipy. Scipy is a collection of statistical distributions and tests that will make your life easier. It is very comprehensive in the families of distributions that they have implemented. Unfortunately, Scipy does not contain fancier econometric procedures like the ones implemented in Stata (time series and panel data analysis). There are other libraries that try to amend this, for example Pandas. However, the econometrics community has not yet jumped into the Python wage, which makes these alternative not as good as Stata. The good news is that almost everything that is implemented in Stata is also implemented in R. Therefore, you can use the useful library RPy2, which is nothing else than an interface between python and R. With RPy2 you can run your code in Python and call R to perform a test on your data and get back the results in your python program.

4.4 NetworkX

If you want to analyze networks, NetworkX provides a comprehensive collection of algorithms for that purpose. The implementation is very elegant, so it allows a lot of flexibility on how you specify a networks (the node can be almost any data type or structure), which is useful if you want to make some fancy simulations.

4.5 Scikit-Learn

If you are fed up with linear regressions, you might want to look into machine learning. For this, python provides Scikit-Learn, a vast library of machine learning algorithm to analyze any kind of data.

4.6 CSV

When you want to import or export data, the most common format is the comma separated value (csv). The library csv provides the tools for that in a very intuitive way.

4.7 Pickle

Another tool to import/export data is pickle. The difference with csv is that pickle is not limited to csv files, but it can take almost any data structure and save it in the hard drive for lated usage. So, if created a fancy network where both nodes and edges have different attributes, you can save it *as it is* with pickle instead of translating it into a text file. Of course, this comes at the cost of loosing some speed in the process, so you have to consider what is more efficient according to your problem.

5 Ready to Go

These are the fundamental tools that, in my experience, you should learn to become proficient in python in a few days. Once you master these tools, you learning curve to use other python libraries will be quite flat, allowing you to focus on more important scientific tasks than reading tutorials. Remember that this is only a collections a pointers intended to help you navigate through the learning process in a more efficient way. For a more in-depth review of different python libraries and tools, I recommend the following books.

- A Byte of Python: An excellent book that covers lots of topics. Best of all, it is free.
- Think Stats: An introductory course to statistics through the lens of python programming. You will learn stats in the most intuitive way, by performing simulated experiments and making sense of strange concepts in a natural way. It is also free!
- Social Network Analysis for Startups: An introductory course to network analysis that teaches yo the essentials, from scraping data from the web to visualizing your networks. All free and all python.
- Quantitative Economics: This is a pretty standard course in numerical methods for economics that uses python. If you are into standard equilibrium economics, this is a must.
- An Introduction to Machine Learning with Scikit-Learn: A great introduction to the field of machine learning, with hands-on experience as you go.