CHAPTER 1

Introduction to Client/Server Networking

This book is about network programming with the Python language: about accomplishing a specific set of tasks that all involve a particular technology—computer networks—using a general-purpose programming language that can do all sorts of things besides the things that you will see illustrated in this book.

We lack the space between the covers of this book to teach you how to program in Python if you have never seen the language before, or never even written a computer program at all. So this book presumes that you have already learned something about Python programming from the many excellent tutorials and books on the subject. We hope that the Python examples in the book are good ones, from which you can learn how to structure and write your own Python programs. But we will be using all sorts of advanced Python features without explanation or apology—though, occasionally, we might point out how we are using a particular technique or construction when we think it is particularly interesting or clever.

On the other hand, this book does not start by assuming that you know any networking! As long as you have ever used a web browser or sent an e-mail, you should know enough to start reading this book at the beginning and learn about computer networking along the way. We will approach networking from the point of view of an application programmer who is either implementing a network-connected service—like a web site, an email server, or a networked computer game—or else writing a client program that is designed to use such a service.

Note that you will not, however, learn how to set up or configure networks from this book, for the simple reason that the Python language is not usually involved when network engineers or system administrators sit down to build and configure their networks. Instead, computer networks are typically assembled from network switches, Ethernet cables, fiber optic strands, and painstakingly configured routers. You will have to learn about devices like those from a book that focuses on creating computer networks in the first place; this book instead will talk about writing programs that use a computer network once it is already set up and running.

The Building Blocks: Stacks and Libraries

As we begin to explore Python network programming, there are two concepts that will appear over and over again:

- The idea of a protocol stack, in which very simple network services are used as a foundation on which to build more sophisticated services.
The fact that you will often be using Python libraries of prepared code—whether from the built-in standard library that ships with Python, or from third-party modules that you download and install—that already know how to speak the network protocol you want to use.

In many cases, network programming simply involves selecting and using a library that already supports the network operations you need to perform. A major purpose of this book is to introduce you to all of the key networking libraries available for Python, and to teach you about the lower-level network services on which those libraries are built—both so that you understand how the libraries work, and so that you will understand what is happening when something at a lower level goes wrong.

Let’s begin with a very simple example. I have here a mailing address, which looks like this:

207 N. Defiance St
Archbold, OH

And I am interested in knowing the latitude and longitude of this physical address. It just so happens that Google provides a “Maps API” that can perform such a conversion. What would I have to do to take advantage of this network service from Python?

When looking at a new network service that you want to use, it is always worthwhile to start by finding out whether someone has already implemented the protocol—in this case, the Google Maps protocol—that your program will need to speak. Start by scrolling through the Python Standard Library documentation, looking for anything having to do with Google Maps:

http://docs.python.org/library/

Do you see anything? No, neither do I. But it is important for a Python programmer to look through the Standard Library’s table of contents pretty frequently, even if you usually do not find what you are looking for, because each reading will make you more familiar with the services that do come included with Python.

Since the Standard Library does not have a package to help us, we can turn to the Python Package Index, an excellent resource for finding all sorts of general-purpose Python packages contributed by other programmers and organizations from across the world. You can also, of course, check the web site of the vendor whose service you will be using to see whether they provide a python library to access it. Or you can do a general Google search for “Python” plus the name of whatever web service you want to use, and see whether any of the first few results link to a package that you might want to try.

In this case, I searched the Python Package Index, which lives at this URL:

http://pypi.python.org/

There, I did a search for Google maps, and immediately found a package that is actually named googlemaps and that provides a clean interface to its features (though, you will note from its description, it is not vendor-provided, but was instead written by someone besides Google):

http://pypi.python.org/pypi/googlemaps/

This is such a common situation—that you find a Python package that sounds like it might already do exactly what you want, and that you want to try it out on your system—that we should pause for a moment and introduce you to the very best Python technology for quickly trying out a new library: virtualenv!

In the old days, installing a Python package was a gruesome and irreversible act that required administrative privileges on your machine and left your system Python install permanently altered. After several months of heavy Python development, your system Python install could become a wasteland of dozens of packages, all installed by hand, and you could even find that the new packages you tried to install would break because they were incompatible with one of the old packages sitting on your hard drive from a project that ended months ago.