CHAPTER 1

IPython: Beyond Normal Python

There are many options for development environments for Python, and I'm often asked which one I use in my own work. My answer sometimes surprises people: my preferred environment is IPython plus a text editor (in my case, Emacs or Atom depending on my mood). IPython (short for *Interactive Python*) was started in 2001 by Fernando Perez as an enhanced Python interpreter, and has since grown into a project aiming to provide, in Perez's words, "Tools for the entire lifecycle of research computing." If Python is the engine of our data science task, you might think of IPython as the interactive control panel.

As well as being a useful interactive interface to Python, IPython also provides a number of useful syntactic additions to the language; we'll cover the most useful of these additions here. In addition, IPython is closely tied with the Jupyter project, which provides a browser-based notebook that is useful for development, collaboration, sharing, and even publication of data science results. The IPython notebook is actually a special case of the broader Jupyter notebook structure, which encompasses notebooks for Julia, R, and other programming languages. As an example of the usefulness of the notebook format, look no further than the page you are reading: the entire manuscript for this book was composed as a set of IPython notebooks.

IPython is about using Python effectively for interactive scientific and data-intensive computing. This chapter will start by stepping through some of the IPython features that are useful to the practice of data science, focusing especially on the syntax it offers beyond the standard features of Python. Next, we will go into a bit more depth on some of the more useful "magic commands" that can speed up common tasks in creating and using data science code. Finally, we will touch on some of the features of the notebook that make it useful in understanding data and sharing results.

Shell or Notebook?

There are two primary means of using IPython that we'll discuss in this chapter: the IPython shell and the IPython notebook. The bulk of the material in this chapter is relevant to both, and the examples will switch between them depending on what is most convenient. In the few sections that are relevant to just one or the other, I will explicitly state that fact. Before we start, some words on how to launch the IPython shell and IPython notebook.

Launching the IPython Shell

This chapter, like most of this book, is not designed to be absorbed passively. I recommend that as you read through it, you follow along and experiment with the tools and syntax we cover: the muscle-memory you build through doing this will be far more useful than the simple act of reading about it. Start by launching the IPython interpreter by typing **`ipython`** on the command line; alternatively, if you've installed a distribution like Anaconda or EPD, there may be a launcher specific to your system (we'll discuss this more fully in "Help and Documentation in IPython" on page 3).

Once you do this, you should see a prompt like the following:

```
IPython 4.0.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
In [1]:
```

With that, you're ready to follow along.

Launching the Jupyter Notebook

The Jupyter notebook is a browser-based graphical interface to the IPython shell, and builds on it a rich set of dynamic display capabilities. As well as executing Python/IPython statements, the notebook allows the user to include formatted text, static and dynamic visualizations, mathematical equations, JavaScript widgets, and much more. Furthermore, these documents can be saved in a way that lets other people open them and execute the code on their own systems.

Though the IPython notebook is viewed and edited through your web browser window, it must connect to a running Python process in order to execute code. To start this process (known as a "kernel"), run the following command in your system shell:

```
$ jupyter notebook
```

This command will launch a local web server that will be visible to your browser. It immediately spits out a log showing what it is doing; that log will look something like this: