

# 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:

```
$ jupyter notebook
[NotebookApp] Serving notebooks from local directory: /Users/jakevdp/...
[NotebookApp] 0 active kernels
[NotebookApp] The IPython Notebook is running at: http://localhost:8888/
[NotebookApp] Use Control-C to stop this server and shut down all kernels...
```

Upon issuing the command, your default browser should automatically open and navigate to the listed local URL; the exact address will depend on your system. If the browser does not open automatically, you can open a window and manually open this address (<http://localhost:8888/> in this example).

## Help and Documentation in IPython

If you read no other section in this chapter, read this one: I find the tools discussed here to be the most transformative contributions of IPython to my daily workflow.

When a technologically minded person is asked to help a friend, family member, or colleague with a computer problem, most of the time it's less a matter of knowing the answer as much as knowing how to quickly find an unknown answer. In data science it's the same: searchable web resources such as online documentation, mailing-list threads, and Stack Overflow answers contain a wealth of information, even (especially?) if it is a topic you've found yourself searching before. Being an effective practitioner of data science is less about memorizing the tool or command you should use for every possible situation, and more about learning to effectively find the information you don't know, whether through a web search engine or another means.

One of the most useful functions of IPython/Jupyter is to shorten the gap between the user and the type of documentation and search that will help them do their work effectively. While web searches still play a role in answering complicated questions, an amazing amount of information can be found through IPython alone. Some examples of the questions IPython can help answer in a few keystrokes:

- How do I call this function? What arguments and options does it have?
- What does the source code of this Python object look like?
- What is in this package I imported? What attributes or methods does this object have?

Here we'll discuss IPython's tools to quickly access this information, namely the `?` character to explore documentation, the `??` characters to explore source code, and the Tab key for autocompletion.

## Accessing Documentation with ?

The Python language and its data science ecosystem are built with the user in mind, and one big part of that is access to documentation. Every Python object contains the