```
$ 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 reference to a string, known as a *docstring*, which in most cases will contain a concise summary of the object and how to use it. Python has a built-in `help()` function that can access this information and print the results. For example, to see the documentation of the built-in `len` function, you can do the following:

```
In [1]: help(len)
Help on built-in function len in module builtins:

len(...)
    len(object) -> integer

    Return the number of items of a sequence or mapping.
```

Depending on your interpreter, this information may be displayed as inline text, or in some separate pop-up window.

Because finding help on an object is so common and useful, IPython introduces the `?` character as a shorthand for accessing this documentation and other relevant information:

```
In [2]: len?
Type:        builtin_function_or_method
String form: <built-in function len>
Namespace:   Python builtin
Docstring:
len(object) -> integer

Return the number of items of a sequence or mapping.
```

This notation works for just about anything, including object methods:

```
In [3]: L = [1, 2, 3]
In [4]: L.insert?
Type:        builtin_function_or_method
String form: <built-in method insert of list object at 0x1024b8ea8>
Docstring:   L.insert(index, object) -- insert object before index
```

or even objects themselves, with the documentation from their type:

```
In [5]: L?
Type:        list
String form: [1, 2, 3]
Length:      3
Docstring:
list() -> new empty list
list(iterable) -> new list initialized from iterable's items
```

Importantly, this will even work for functions or other objects you create yourself! Here we'll define a small function with a docstring:

```
In [6]: def square(a):
  ....:     """Return the square of a."""
```

```
   ...:     return a ** 2
   ...:
```

Note that to create a docstring for our function, we simply placed a string literal in the first line. Because docstrings are usually multiple lines, by convention we used Python's triple-quote notation for multiline strings.

Now we'll use the `?` mark to find this docstring:

```
In [7]: square?
Type:        function
String form: <function square at 0x103713cb0>
Definition:  square(a)
Docstring:   Return the square of a.
```

This quick access to documentation via docstrings is one reason you should get in the habit of always adding such inline documentation to the code you write!

## Accessing Source Code with ??

Because the Python language is so easily readable, you can usually gain another level of insight by reading the source code of the object you're curious about. IPython provides a shortcut to the source code with the double question mark (`??`):

```
In [8]: square??
Type:        function
String form: <function square at 0x103713cb0>
Definition:  square(a)
Source:
def square(a):
    "Return the square of a"
    return a ** 2
```

For simple functions like this, the double question mark can give quick insight into the under-the-hood details.

If you play with this much, you'll notice that sometimes the `??` suffix doesn't display any source code: this is generally because the object in question is not implemented in Python, but in C or some other compiled extension language. If this is the case, the `??` suffix gives the same output as the `?` suffix. You'll find this particularly with many of Python's built-in objects and types, for example `len` from above:

```
In [9]: len??
Type:        builtin_function_or_method
String form: <built-in function len>
Namespace:   Python builtin
Docstring:
len(object) -> integer

Return the number of items of a sequence or mapping.
```