

Underscore Shortcuts and Previous Outputs

The standard Python shell contains just one simple shortcut for accessing previous output; the variable `_` (i.e., a single underscore) is kept updated with the previous output; this works in IPython as well:

```
In [9]: print(_)  
1.0
```

But IPython takes this a bit further—you can use a double underscore to access the second-to-last output, and a triple underscore to access the third-to-last output (skipping any commands with no output):

```
In [10]: print(__)  
-0.4161468365471424
```

```
In [11]: print(___)  
0.9092974268256817
```

IPython stops there: more than three underscores starts to get a bit hard to count, and at that point it's easier to refer to the output by line number.

There is one more shortcut we should mention, however—a shorthand for `Out[X]` is `_X` (i.e., a single underscore followed by the line number):

```
In [12]: Out[2]  
Out[12]: 0.9092974268256817
```

```
In [13]: _2  
Out[13]: 0.9092974268256817
```

Suppressing Output

Sometimes you might wish to suppress the output of a statement (this is perhaps most common with the plotting commands that we'll explore in [Chapter 4](#)). Or maybe the command you're executing produces a result that you'd prefer not to store in your output history, perhaps so that it can be deallocated when other references are removed. The easiest way to suppress the output of a command is to add a semicolon to the end of the line:

```
In [14]: math.sin(2) + math.cos(2);
```

Note that the result is computed silently, and the output is neither displayed on the screen or stored in the `Out` dictionary:

```
In [15]: 14 in Out  
Out[15]: False
```

Related Magic Commands

For accessing a batch of previous inputs at once, the `%history` magic command is very helpful. Here is how you can print the first four inputs:

```
In [16]: %history -n 1-4
1: import math
2: math.sin(2)
3: math.cos(2)
4: print(In)
```

As usual, you can type `%history?` for more information and a description of options available. Other similar magic commands are `%rerun` (which will re-execute some portion of the command history) and `%save` (which saves some set of the command history to a file). For more information, I suggest exploring these using the `? help` functionality discussed in [“Help and Documentation in IPython” on page 3](#).

IPython and Shell Commands

When working interactively with the standard Python interpreter, one of the frustrations you’ll face is the need to switch between multiple windows to access Python tools and system command-line tools. IPython bridges this gap, and gives you a syntax for executing shell commands directly from within the IPython terminal. The magic happens with the exclamation point: anything appearing after `!` on a line will be executed not by the Python kernel, but by the system command line.

The following assumes you’re on a Unix-like system, such as Linux or Mac OS X. Some of the examples that follow will fail on Windows, which uses a different type of shell by default (though with the 2016 announcement of native Bash shells on Windows, soon this may no longer be an issue!). If you’re unfamiliar with shell commands, I’d suggest reviewing the [Shell Tutorial](#) put together by the always excellent Software Carpentry Foundation.

Quick Introduction to the Shell

A full intro to using the shell/terminal/command line is well beyond the scope of this chapter, but for the uninitiated we will offer a quick introduction here. The shell is a way to interact textually with your computer. Ever since the mid-1980s, when Microsoft and Apple introduced the first versions of their now ubiquitous graphical operating systems, most computer users have interacted with their operating system through familiar clicking of menus and drag-and-drop movements. But operating systems existed long before these graphical user interfaces, and were primarily controlled through sequences of text input: at the prompt, the user would type a command, and the computer would do what the user told it to. Those early prompt