Input and Output History

输入输出历史

> Previously we saw that the IPython shell allows you to access previous commands with the up and down arrow keys, or equivalently the Ctrl-p/Ctrl-n shortcuts. Additionally, in both the shell and the notebook, IPython exposes several ways to obtain the output of previous commands, as well as string versions of the commands themselves. We'll explore those here.

前面我们看到IPython shell能够让你获取到命令的历史，使用向上箭头或者向下箭头，或者等同的Ctrl-p/Ctrl-n快捷键。除此之外，在IPython shell和notebook中，还提供了一些方法可以获得前面命令的输出结果，或者字符串形式的命令本身。本节将讨论它们。

IPython's `In` and `Out` Objects

IPython的 `In` 和 `Out` 对象

> By now I imagine you're quite familiar with the `In [1]:` / `Out[1]:` style prompts used by IPython. But it turns out that these are not just pretty decoration: they give a clue as to how you can access previous inputs and outputs in your current session. Imagine you start a session that looks like this:

阅读到这里，作者认为你已经相当熟悉IPython的 `In [1]:` / `Out[1]:` 风格的提示符了。但是其实这些提示符并不是为了美观而采用的装饰符号：它们会给出你提示，让你可以获取之前的输入和输出。例如你启动了一个IPython会话：

```
In [1]: import math

In [2]: math.sin(2)
Out[2]: 0.9092974268256817

In [3]: math.cos(2)
Out[3]: -0.4161468365471424
```

> We've imported the built-in `math` package, then computed the sine and the cosine of the number 2. These inputs and outputs are displayed in the shell with `In` / `Out` labels, but there's more–IPython actually creates some Python variables called `In` and `Out` that are automatically updated to reflect this history:

我们载入了内建的 `math` 包，然后计算了2的正弦和余弦值。这些输入和输出在IPython shell当中使用 `In` / `Out` 标签打印在屏幕上，但实际上这些标签的作用不限于此，IPython创建了两个Python的变量名叫 `In` 和 `Out` ，在每次输入输出的情况下都会自动更新和相应：

```
In [4]: print(In)
['', 'import math', 'math.sin(2)', 'math.cos(2)', 'print(In)']

In [5]: Out
Out[5]: {2: 0.9092974268256817, 3: -0.4161468365471424}
```

> The `In` object is a list, which keeps track of the commands in order (the first item in the list is a place-holder so that `In[1]` can refer to the first command):

`In` 对象是一个列表，保存着本次IPython会话的所有输入命令（列表中的第一个元素是一个占位符，因此第一条命令是 `In[1]` ）：

```
In [6]: print(In[1])
import math
```

> The `Out` object is not a list but a dictionary mapping input numbers to their outputs (if any):

`Out` 对象是一个字典值，将输入的编号对应到它们相应的输出上面：

```
In [7]: print(Out[2])
0.9092974268256817
```

> Note that not all operations have outputs: for example, `import` statements and `print` statements don't affect the output. The latter may be surprising, but makes sense if you consider that `print` is a function that returns `None` ; for brevity, any command that returns `None` is not added to `Out` .

注意并不是所有的操作都有输出：例如， `import` 和 `print` 语句就不会影响输出内容。然后再深入思考一下，你会发现， `print` 是一个返回值为 `None` 的函数；简而言之，任何指令返回None都不会加入到 `Out` 当中。

> Where this can be useful is if you want to interact with past results. For example, let's check the sum of `sin(2) ** 2` and `cos(2) ** 2` using the previously-computed results:

当你需要用到历史结果时，上面的变量就非常有用。例如，我们检查一下 `sin(2) ** 2` 加上 `cos(2) ** 2` 的和，可以使用前面的结果：

```
In [8]: Out[2] ** 2 + Out[3] ** 2
Out[8]: 1.0
```

> The result is `1.0` as we'd expect from the well-known trigonometric identity. In this case, using these previous results probably is not necessary, but it can become very handy if you execute a very expensive computation and want to reuse the result!

结果是 `1.0` ，和我们了解的三角函数运算得到的一样。在这个例子中，使用历史结果并不是特别需要，但是当你前面进行了非常耗时的运算的时候，重用这个结果是非常方便的。

Underscore Shortcuts and Previous Outputs

下划线变量和之前的输出

> 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:

标准的Python shell包含着一个简单的快捷变量用来获取前一个输出结果；变量 `_` （一个下划线），这个变量会更新为每次前一条语句的输出结果。IPython中也是可以使用的：

```
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):

IPython扩展了这个功能，你可以使用双下划线获取倒数第二个输出结果，使用三下划线获取倒数第三个输出结果（当然会跳过无输出的命令）：

```
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):

三个就打住了，IPython也不支持更多的下划线了，因为多于三个的下划线就变得比较难以数清楚了，在这种情况下，使用输入序号会更加方便一些。

这里还有一个快捷方式需要介绍， `Out[x]` 的快捷写法是 `_x` （一个下划线后面跟着输入序号）：

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

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

Suppressing Output

取消输出

> 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 Introduction to Matplotlib). Or maybe the command you're executing produces a result that you'd prefer not like 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:

有时你可能希望取消一个语句的输出结果（这在我们使用绘图指令时很常见，我们会在Matplotlib简介中详细讨论）。或者你在执行的指令会产生的结果，你并不希望结果被存储在输出历史中，这样的结果就能在其他引用被移除后自动释放资源。取消一个指令的输出结果最简单的方法就是在语句最后加上一个分号：

```
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:

这里结果将会静默的计算出来，输出既不会打印在屏幕上，也不会保存在输出 `Out` 的字典中：

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

Related Magic Commands

相关的魔术命令

> 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:

要想一次性获得批量的输入历史， `%history` 魔术命令是非常有用的。下面例子展示了如何使用它打印出输入历史中头四个指令：

```
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.

当然，你也可以使用 `%history?` 来查阅该魔术命令的文档。其他类似的魔术命令包括 `%rerun` （重新执行输入历史中的某部分指令）和 `%save` （将输入历史中的某部分内容保存成文件）。需要更多的信息，推荐使用 `?` 魔术符号来查阅文档，有关 `?` 号的内容请参见IPython帮助和文档。