IPython Magic Commands

IPython魔术命令

> The previous two sections showed how IPython lets you use and explore Python efficiently and interactively. Here we'll begin discussing some of the enhancements that IPython adds on top of the normal Python syntax. These are known in IPython as *magic commands*, and are prefixed by the `%` character. These magic commands are designed to succinctly solve various common problems in standard data analysis. Magic commands come in two flavors: *line magics*, which are denoted by a single `%` prefix and operate on a single line of input, and *cell magics*, which are denoted by a double `%%` prefix and operate on multiple lines of input. We'll demonstrate and discuss a few brief examples here, and come back to more focused discussion of several useful magic commands later in the chapter.

前两小节展示了怎样使用IPython，今你在其中执行Python代码更加有效和具有交互性。现在我们要开始讨论一些IPython增强的语言特性。这些特性被称为IPython的*魔术命令*，它们都是以 % 字符开头的。这些魔术命令被设计用来简洁地实现很多通用的标准数据科学问题。魔术命令分成两种模式：*行魔术*，以一个 % 开头，是对于一行的输入进行魔术处理的；另一种是*单元格魔术*，以两个 %% 开头，是对于多行的输入进行魔术处理的。本节我们会展示和讨论一些例子，然后本章后续小节会对部分有用的魔术命令进行详细的讨论。

Pasting Code Blocks: `%paste` and `%cpaste`

粘贴代码块：`%paste` 和 `%cpaste`

> When working in the IPython interpreter, one common gotcha is that pasting multi-line code blocks can lead to unexpected errors, especially when indentation and interpreter markers are involved. A common case is that you find some example code on a website and want to paste it into your interpreter. Consider the following simple function:

当使用IPython解释器时，我们会遇到一个坑，就是粘贴多行代码块是会出现很多意料之外的错误，尤其是当存在缩进和提示符的情况下。其中一个常见的情况就是当你在网上找到一些示例代码，然后想将它们粘贴到你的解释器中。例如下面这个简单的函数：

```
>>> def donothing(x):
...     return x
```

> The code is formatted as it would appear in the Python interpreter, and if you copy and paste this directly into IPython you get an error:

这段代码在Python解释器中就会像上面那样展示，但是如果你采用通常的复制粘贴大法将它们粘贴到IPython的时候，错误就发生了：

```
In [2]: >>> def donothing(x):
   ...:     ...     return x
   ...:
  File "<ipython-input-20-5a66c8964687>", line 2
    ...     return x
            ^
SyntaxError: invalid syntax
```

> In the direct paste, the interpreter is confused by the additional prompt characters. But never fear–IPython's `%paste` magic function is designed to handle this exact type of multi-line, marked-up input:

在直接粘贴的情况下，解释器被额外的提示符号搞蒙了。不怕，IPyton的 `%paste` 魔术命令是专门为了处理这种情况（多行代码块，带提示符号）设计的：

```
In [3]: %paste
>>> def donothing(x):
...     return x

## -- End pasted text --
```

> The `%paste` command both enters and executes the code, so now the function is ready to be used:

`%paste` 命令既输入了多行代码又执行了它们，因此 `donothing` 函数已经可以使用了：

```
In [4]: donothing(10)
Out[4]: 10
```

> A command with a similar intent is `%cpaste`, which opens up an interactive multiline prompt in which you can paste one or more chunks of code to be executed in a batch:

还有一个魔术命令 `%cpaste` 也是类似的作用，它会打开一个交互的多行提示符，允许你粘贴多个代码块然后批量执行它们：

```
In [5]: %cpaste
Pasting code; enter '--' alone on the line to stop or use Ctrl-D.
:>>> def donothing(x):
:...     return x
:--
```

> These magic commands, like others we'll see, make available functionality that would be difficult or impossible in a standard Python interpreter.

这些魔术命令，还有我们马上会看到的其他命令，提供了标准Python解释器很难或无法提供的功能。

Running External Code: `%run`

执行外部代码：`%run`

> As you begin developing more extensive code, you will likely find yourself working in both IPython for interactive exploration, as well as a text editor to store code that you want to reuse. Rather than running this code in a new window, it can be convenient to run it within your IPython session. This can be done with the `%run` magic.

当你使用IPython开发更多代码之后，你会发现你可能需要两个环境，在IPython中交互式的进行探索和快速验证，使用文本编辑器保存那些以后你需要重用的代码。当你需要在IPython中运行你已经保存好的Pythonn代码文件时，你不需要打开一个新的进程执行它们，也不需要将它们的代码粘贴进来，你可以使用 `%run` 魔术。

> For example, imagine you've created a `myscript.py` file with the following contents:

例如，你创建了一个 `myscript.py` 文件，里面的内容是：

```
#-------------------------------------
# file: myscript.py

def square(x):
    """square a number"""
    return x ** 2

for N in range(1, 4):
    print(N, "squared is", square(N))
```

> You can execute this from your IPython session as follows:

你可以在你的IPython shell中这样执行这个Pythonf代码文件：

```
In [6]: %run myscript.py
1 squared is 1
2 squared is 4
3 squared is 9
```

> Note also that after you've run this script, any functions defined within it are available for use in your IPython session:

你应该注意到了，当你执行完这个脚本文件之后，任何定义了的函数也可以在你当前的IPython会话中使用了。

```
In [7]: square(5)
Out[7]: 25
```

> There are several options to fine-tune how your code is run; you can see the documentation in the normal way, by typing **`%run?`** in the IPython interpreter.

还有一些参数可以精细控制你的代码文件如何执行；你可以像之前介绍的那样查看它的文档，只需要在IPython shell中输入 `%run?` 即可。

Timing Code Execution: `%timeit`

代码执行计时：`%timeit`

> Another example of a useful magic function is `%timeit`, which will automatically determine the execution time of the single-line Python statement that follows it. For example, we may want to check the performance of a list comprehension:

下面介绍的魔术命令是 `%timeit`，它会自动测试统计紧跟之后的单行Python语句的执行性能（时间）。例如我们需要测试列表解析的性能：

```
In [8]: %timeit L = [n ** 2 for n in range(1000)]
1000 loops, best of 3: 325 µs per loop
```

> The benefit of `%timeit` is that for short commands it will automatically perform multiple runs in order to attain more robust results. For multi line statements, adding a second `%` sign will turn this into a cell magic that can handle multiple lines of input. For example, here's the equivalent construction with a `for`-loop:

使用 `%timeit` 的时候，它会自动执行多次，以获取更有效的结果。对于多行的代码来说，增加一个 `%` 号，会将本魔术命令变成单元格模式，因此它能测试多行输入的性能。例如，下面是一段相同功能的列表初始化，使用的 `for` 循环：

```
In [9]: %%timeit
   ...: L = []
   ...: for n in range(1000):
   ...:     L.append(n ** 2)
   ...:
1000 loops, best of 3: 373 µs per loop
```

> We can immediately see that list comprehensions are about 10% faster than the equivalent `for`-loop construction in this case. We'll explore `%timeit` and other approaches to timing and profiling code in Profiling and Timing Code.

从上面的结果可以看出来，使用列表解析能比使用 `for` 循环的方式提升10%的运行速度。我们将在性能测算和计时中更加详细的讨论它。

Help on Magic Functions: `?`, `%magic`, and `%lsmagic`

魔术命令帮助：`?`、`%magic` 和 `%lsmagic`

> Like normal Python functions, IPython magic functions have docstrings, and this useful documentation can be accessed in the standard manner. So, for example, to read the documentation of the `%timeit` magic simply type this:

就像普通的Python对象，IPython魔术命令也有docstring，这些文档可以按照我们之前的方式简单的获取到。举个例子，想要查阅 `%timeit` 的文档，仅需输入：

```
In [10]: %timeit?
```

> Documentation for other functions can be accessed similarly. To access a general description of available magic functions, including some examples, you can type this:

其他魔术命令和文档也可以类似获得。要获得魔术命令的通用描述以及它们的例子，你可以输入：

```
In [11]: %magic
```

> For a quick and simple list of all available magic functions, type this:

如果想要快速简单地列出所有可用的魔术命令，输入：

```
In [12]: %lsmagic
```

> Finally, I'll mention that it is quite straightforward to define your own magic functions if you wish. We won't discuss it here, but if you are interested, see the references listed in More IPython Resources.

最后，你可以了解自定义魔术命令的有关知识。但是本书不会讨论这个方面，如果读者感兴趣，请参见更多IPython资源。