**使用IPython有哪些好处**

Python工程师需要快速验证代码运行结果是否符合预期。最快捷方便的做法就是使用Python自带的交互模式，但是这个Python Shell有非常多的弊端：

* 不能在退出时保存历史记录以备未来查询。
* 不支持Tab自动补全。
* 不能快速获得模块/函数/类的信息，如参数、文档、原始代码等。
* 不方便在交互环境下执行Shell命令。

IPython是一个基于Python Shell的交互式解释器，但是有比默认Shell强大得多的编辑和交互功能。笔者在开发中，有时候在IPython交互环境下的时间甚至比使用编辑器的时间还长。14年的时候我就写过2篇博客：

[ipython的一些高级用法(一)](https://link.zhihu.com/?target=http%3A//www.dongwm.com/archives/ipythonde-xie-gao-ji-yong-fa/)

[ipython的一些高级用法(二)](https://link.zhihu.com/?target=http%3A//www.dongwm.com/archives/ipythonde-xie-gao-ji-yong-fa-er/)

时隔三年，我重新梳理一下对于开发者非常有用得几点吧：

1. **获得对象信息**：输入你想要查看的对象，然后加上一个或者两个问号，就能获得多种对象信息。一个问号只是显示对象的签名，文档字符串以及代码文件的位置，二个问号可以直接显示源代码，这样直接节省了用编辑器打开代码文件然后搜索对应代码的时间。

2. **调用系统Shell命令。只需要在命令前加!即可：**

In : !uptime

16:31:11 up 1:41, 3 users, load average: 0.02, 0.06, 0.09

3. **Tab自动补全。**IPython可以自动检查对象的属性，通过object\_name.<TAB>列出全部的子属性，再使用Tab切换到对应的属性上，然后回车就可以了。

4. **历史记录**。IPython把输入的历史记录存放在个人配置目录下的history.sqlite文件中，并且可以结合%rerun、%recall、%macro、%save等Magic函数使用。尤为有意义的是，它把最近的三次执行记录绑定在\_、\_\_和\_\_\_这三个变量上。搜索历史记录时，还支持Ctrl-r、 Ctrl-n和Ctrl-p等快捷键。

**然后第二类是IPython有很多Magic函数**（可以使用%lsmagic获得全部可用的Magic函数），我举例几个我非常常用的：

1. %debug：激活交互的调试器：

In : a = 0

In : b = 1

In : b / a

---------------------------------------------------------------------------

ZeroDivisionError Traceback (most recent call last)

<ipython-input-8-fc1551a96997> in <module>()

----> 1 b / a

ZeroDivisionError: integer division or modulo by zero

In : %debug

> <ipython-input-8-fc1551a96997>(1)<module>()

----> 1 b / a

ipdb> p b

1

ipdb> p a

0

可以直接pdb或者pidb去调试问题了。

2. %hist：%history的别名，查看历史记录。

3. %load：把外部代码加载进来。

4. %rehashx：把$PATH中的可执行命令都更新进别名系统，这样就可以在IPython中不加感叹号而调用了：

In : echo 1

File "<ipython-input-10-334d5669e1fb>", line 1

echo 1

^

SyntaxError: invalid syntax

In : %rehashx

In : echo 1

1

5. %timeit：获得程序执行时间。timeit是Python内置的库，用来测量小代码片的执行时间。

6. %save：把某些历史记录保存到文件中。

7. %logstart/logoff：记录会话。退出IPython后还可以回到之前的状态。

8. %edit：使用编辑器打开，但需要设定EDITOR这个环境变量。假如写了一个很复杂的函数，代码很长，执行后发现不符合预期，用历史记录找到这个函数，然后用鼠标移到对应的位置修改很不方便。其实这时应该使用edit来编辑

9. %macro：把历史记录、文件等封装为宏，以便未来重新执行。

**最后一种好处是IPython的扩展系统。**我介绍最常用的2个：

1. storemagic。storemagic可以持久化宏、变量和别名。可以添加如下配置到ipython\_config.py实现自动保存：

c.StoreMagics.autorestore = True

举个例子，第一次在IPython中执行如下命令：

In : l = ['hello', 10, 'world']

In : %store l

Stored 'l' (list)

In : exit()

这样就把l存储下来了。现在退出IPython后重新进入：

❯ ipython

In : l

Out: ['hello', 10, 'world']

可以看到，l能直接使用。我们还可以用这个功能保存一些重要的资源，这样即使退出IPython也能找回来。

2. autoreload。可以让我们不退出IPython就动态修改代码，在执行代码前IPython会帮我们自动重载改动的模块，这种思想在多种Web框架中都可见其踪影。

先看一个简单的例子：

❯ cat py\_autoreload.py

def a():

return 1

在IPython里面执行它：

In : %load\_ext autoreload

In : %autoreload 2

In : from py\_autoreload import a

In : a()

Out: 1

然后打开另外一个终端修改函数a:

def a():

return 2

在之前的IPython中重新调用a函数：

In : a()

Out: 2

可以看到返回值动态地改变了。

IPython设计的还很容易自定义一个扩展，简直不要太方便哦。

剩下的大家自己挖掘喽 ✧(≖ ◡ ≖✿)