10.6 重新加载模块¶

问题¶

你想重新加载已经加载的模块，因为你对其源码进行了修改。

解决方案¶

使用imp.reload()来重新加载先前加载的模块。举个例子：

```
>>> import spam
>>> import imp
>>> imp.reload(spam)
<module 'spam' from './spam.py'>
>>>
```

讨论¶

重新加载模块在开发和调试过程中常常很有用。但在生产环境中的代码使用会不安全，因为它并不总是像您期望的那样工作。

reload()擦除了模块底层字典的内容，并通过重新执行模块的源代码来刷新它。模块对象本身的身份保持不变。因此，该操作在程序中所有已经被导入了的地方更新了模块。

尽管如此，reload()没有更新像”from module import name”这样使用import语句导入的定义。举个例子：

```
# spam.py
def bar():
    print('bar')

def grok():
    print('grok')
```

现在启动交互式会话：

```
>>> import spam
>>> from spam import grok
>>> spam.bar()
bar
>>> grok()
grok
>>>
```

不退出Python修改spam.py的源码，将grok()函数改成这样：

```
def grok():
    print('New grok')
```

现在回到交互式会话，重新加载模块，尝试下这个实验：

```
>>> import imp
>>> imp.reload(spam)
<module 'spam' from './spam.py'>
>>> spam.bar()
bar
>>> grok() # Notice old output
grok
>>> spam.grok() # Notice new output
New grok
>>>
```

在这个例子中，你看到有2个版本的grok()函数被加载。通常来说，这不是你想要的，而是令人头疼的事。

因此，在生产环境中可能需要避免重新加载模块。在交互环境下调试，解释程序并试图弄懂它。