10.9 将文件夹加入到sys.path¶

问题¶

你无法导入你的Python代码因为它所在的目录不在sys.path里。你想将添加新目录到Python路径，但是不想硬链接到你的代码。

解决方案¶

有两种常用的方式将新目录添加到sys.path。第一种，你可以使用PYTHONPATH环境变量来添加。例如：

```
bash % env PYTHONPATH=/some/dir:/other/dir python3
Python 3.3.0 (default, Oct 4 2012, 10:17:33)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.path
['', '/some/dir', '/other/dir', ...]
>>>
```

在自定义应用程序中，这样的环境变量可在程序启动时设置或通过shell脚本。

第二种方法是创建一个.pth文件，将目录列举出来，像这样：

```
# myapplication.pth
/some/dir
/other/dir
```

这个.pth文件需要放在某个Python的site-packages目录，通常位于/usr/local/lib/python3.3/site-packages 或者 ~/.local/lib/python3.3/sitepackages。当解释器启动时，.pth文件里列举出来的存在于文件系统的目录将被添加到sys.path。安装一个.pth文件可能需要管理员权限，如果它被添加到系统级的Python解释器。

讨论¶

比起费力地找文件，你可能会倾向于写一个代码手动调节sys.path的值。例如:

```
import sys
sys.path.insert(0, '/some/dir')
sys.path.insert(0, '/other/dir')
```

虽然这能“工作”，但是在实践中极为脆弱，应尽量避免使用。这种方法的问题是，它将目录名硬编码到了你的源代码。如果你的代码被移到一个新的位置，这会导致维护问题。更好的做法是在不修改源代码的情况下，将path配置到其他地方。如果您使用模块级的变量来精心构造一个适当的绝对路径，有时你可以解决硬编码目录的问题，比如__file__。举个例子：

```
import sys
from os.path import abspath, join, dirname
sys.path.insert(0, join(abspath(dirname(__file__)), 'src'))
```

这将src目录添加到path里，和执行插入步骤的代码在同一个目录里。

site-packages目录是第三方包和模块安装的目录。如果你手动安装你的代码，它将被安装到site-packages目录。虽然用

site-packages目录是第三方包和模块安装的目录。如果你手动安装你的代码，它将被安装到site-packages目录。虽然用于配置path的.pth文件必须放置在site-packages里，但它配置的路径可以是系统上任何你希望的目录。因此，你可以把你的代码放在一系列不同的目录，只要那些目录包含在.pth文件里。