PCS100　import ~ 模块及包的使用

概述

Python 语言是通过 import 或 from import 语句来调用模块的。如果程序文件或模块比较多，则会导致整个目录杂乱无章，为了便于管理各个模块，我们把各个模块分门别类存放在不同的文件夹下，并把单独存放模块的文件夹称做包。

应用

模块(modules)

模块（modules）其实就是实现一定具体功能的普通 Python 脚本文件，命名规则是模块名称后加上.py 后缀；主要供其他程序将其引入,以便利用其提供的操作、功能和数据，Python 标准库全部是以模块方式提供的。例如：fibo 模块（fibo.py）是一个实现 Fibonacci 功能的模块。

```
# -*- coding: utf-8 -*-
# Fibonacci 数列模块
# 输出所有小于 n 的 Fibonacci 数
def fib(n):
    a, b = 0, 1
    if n == 1:
        print 1
    while b < n:
        print b,
```

```
        a, b = b, a+b

# 返回所有小于 n 的 Fibonacci 数
def fib2(n):
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

在 Python 解释器中，使用 import fibo 语句导入 fibo 模块，使用 fibo.fib(1000)来调用函式，也可以用 fib = fibo.fib 将模块函式赋值到本地函式。

```
>>> import fibo
>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fibo.fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
>>> fibo.__name__
'fibo'
>>> fib = fibo.fib
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
>>> from fibo import fib,fib2
>>> fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
>>> fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

正如上述代码中，不仅可以通过 import 来实现模块中函式的使用，还可以通过 from ...import 方式来导入模块中的函式。

包(package)

包是采用“.”组织模块命名空间方式的，比如模块名称 A.B 表示是 A 包中的模块 B。这种命名空间的组织方式能够避免不同模块命名的冲突。例如：设计一组模块来处理声音文件和声音数据，即如何组织一个包。由于存在多个不同声音格式的文件，需要一个随时能增加新模块的包来处理新增的声音格式。另外还需要对声音进行各种不同处理（例如混声、加回音、加入平衡、加入人工音效等），所以还需要另写一些模块来作这些处理。如以下组织结构：

```
Sound/                          Top-level package
      __init__.py               Initialize the sound package
```

```
        Formats/                Subpackage for file format conversions
                __init__.py
                wavread.py
                wavwrite.py
                aiffread.py
                aiffwrite.py
                auread.py
                auwrite.py
                ...
        Effects/                Subpackage for sound effects
                __init__.py
                echo.py
                surround.py
                reverse.py
                ...
        Filters/                Subpackage for filters
                __init__.py
                equalizer.py
                vocoder.py
                karaoke.py
                ...
```

__init__.py 是必须的，它帮助 Python 将该目录识别为包。在最简单的例子中，__init__.py 是一个空文件。当然也可以让__init__.py 做一些包的初始化动作或是设定一些变量，如__all__变量。

import

直接导入包中的一个模块或导入模块中定义的一个函式（如模块 fibo 的函式 fib）来使用，比如：

```
        # -*- coding: utf-8 -*-
        import Sound.Effects.echo
        # 使用这个模块,必须使用完整的名字来调用
        Sound.Effects.echo.echofilter(input, output, delay=0.7, atten=4)

        # 另一种替代方法
        from Sound.Effects import echo
        # 不同的是，不需要包前缀
        echo.echofilter(input, output, delay=0.7, atten=4)

        # 另一种直接导入你需要的函式和变量的方法
        from Sound.Effects.echo import echofilter
        # 其使用方法为
```

```
    echofilter(input, output, delay=0.7, atten=4)
```

from ... import

另一种写法 from Sound.Effects import * 会怎么样？ 理想情况下，可能期望它会搜寻整个包目录，然后搜寻所有的模块并且一一导入。但是，在 Mac 及 Windows 平台下，文件的名称大小写不一致，无法保证所有的模块都会被导入。所以唯一的解决方法就是包的作者提供一个明确索引给使用包的人。如果遵守该习惯的话，当使用包的人在导入的时候使用 from Sound.Effects import *，就会查找包中的__init__.py 中的__all__这个 list 变量，该变量就包含所有应该被导入的模块名称。身为包的作者有责任维护更新__init__.py。以 Sounds/Effects/__init__.py 为例：

```
    __all__ = ["echo", "surround", "reverse"]
```

表示当 from Sound.Effects import * 时会 import 这三个 module。如果没有定义 all，from Sound.Effects import *不会保证所有的子模块被导入。所以要么通过 init.py，要么显式地 import 以保证子模块被导入。如下所示：

```
    import Sound.Effects.echo
    import Sound.Effects.surround
    from Sound.Effects import *
```

值得注意的是 import *不被鼓励，因为这样会降低程序的可读性，虽然这样会减少一些打字工作。有些模块在设计时故意只让某些特别的名称可以被使用，如使用 from Package import specific_submodule，这样做没有任何不对，但如果你的模块名称和其他名称冲突，就得使用 as 为冲突模块取个别名：

```
    from Package import specific_submodule as specific_submodule_alias
```

搜寻路径

当你导入 fibo 时，Python 解释器先在当前目录下搜寻 fibo.py 文件，如果没有找到，会依次在$PYTHONPATH 指示的所有路径中搜寻。$PYTHONPATH 的设定方法与$PATH 是一样的，即多个目录路径的字符串。事实上，模块的搜寻路径是依照 sys.path 变量（多个路径组成 的 list 变量）来的。当 Python 解释器启动时，会将当前目录、$PYTHONPATH，以及按照安装时设定的一些目录加入 sys.path 变量中，所以可以修改这些参数来控制搜寻模块的路径。例如：

```
    # -*- coding: utf-8 -*-
    import sys
    # 将 fibo.py 的路径添加到 sys.path
    sys.path.append('/home/shengyan/workspace/')
```

```
from pcs import fibo
```

小结

本文简要介绍了模块、包的相关知识，同时介绍了 import、from import 的使用及相关内容，更多的阅读资料可查看下面的资源链接。

- LpyQLearn-6-model：
  http://wiki.woodpecker.org.cn/moin/ObpLovelyPython/LpyQLearn-6-model
  精巧地址：*http://bit.ly/3ZiH8s*
- Importing Python Modules：*http://effbot.org/zone/import-confusion.htm* ，这是很适合于初学者阅读的一篇文章。
  精巧地址：*http://bit.ly/2xqCpU*
- Modules 文档：*http://docs.python.org/tut/node8.html*
  精巧地址：*http://bit.ly/px6V6*
- import 语句：*http://docs.python.org/ref/import.html*
  精巧地址：*http://bit.ly/3H9UUC*