16.模块和包

一、模块的概念

当代码量变得相当大的时候，我们最好把代码分成一些有组织的代码段，前提是保证他们的彼此交互。这些代码片段相互间有一定的联系，可能是一个包含数据成员和方法的类，也可能是一组相关但彼此独立的操作函数。这些代码段是共享的，所以python允许“调入”一个模块，允许使用其他模块的属性来利用之前的工作成果，实现代码重用。这个把其他模块中属性附加到你的模块中的操作叫做导入（import）。那些自我包含并且有组织的代码片段就是模块（module）。
简单的来讲，其实一个python文件就是一个模块。

1. 模块的命名空间（名称空间）
   要使用一个模块中的属性和方法，那么就必须在属性和方法的前面使用模块名加点（.）操作符来使用它们。比如之前用到过的random模块，要使用random模块的randint方法，我们需要使用random.randint()的方式来使用。
2. 模块的搜索路径
   当我们导入一个模块的时候，python首先会搜索当前路径（导入模块的路径）下有没有该模块，如果没有的话那么就会去系统的环境变量中搜索，如果都搜索不到，就会报ImportError异常。如果当前路径下有模块名跟系统的模块名重名，那么当前的模块会覆盖掉系统的模块（所以尽量不要跟系统的模块重名，否则就无法使用系统的模块的功能）。

```
# 查看系统环境变量方法
import sys
print sys.path
# 往系统环境变量里面追加路径
sys.path.append("C:\\Python27\\Scripts")
```

二、导入模块

1. import语句

```
# 一行导入一个模块
import urllib
import re
# 一行导入多个模块
import urllib2, sys, random
```

2. python导入模块的顺序
    - python标准库模块
    - python第三方模块
    - 自定义模块
    - 每个部分用一个空行分开
3. python模块作用域
   解析器执行到这条语句（import），如果在搜索路径中找到了指定模块，就会加载它。该过程遵循作用域原则，如果在一个模块的顶层导入，那么它的作用域就是全局的；如果在函数中导入，它的作用域是局部的。

4. from-import语句
   你可以在你的模块里面导入指定的模块属性。也就是把指定名称导入到当前作用域。使用from-import语句可以实现，它的语法是：from module import name1[,name2[,name3...]]
   例如，我们如果只使用到random模块的randint方法，那么我们可以这么做：

```
from random import randint
# 这样我们就可以直接使用randint方法了，而不需要在前面加上random
num = randint(10, 100)
# 如果我们需要使用random模块里面的所有属性和方法
# 但是又不想在前面添加random，那么可以使用
from random import *
# 上面的语句就会将random模块里面所以可以导出的属性和方法都加载到当前模块的作用域
# 但是一般不推荐这样子写，因为有可能会导致名称冲突。
# 比如其他的模块里面也有一个randint方法，这样的话就会导致名称冲突了
```

5. 扩展的import语句（as）
   有时候你导入的模块或者是模块属性名称已经在你的程序中使用了，或者你不想使用导入的名字。可能是他太长不变输入什么的，总之你不喜欢他。想给他换个名字，那么就可以使用as语句来给模块或属性方法等起别名，然后在程序中使用这个别名即可。

```
# 把random模块重命名为rd
import random as rd
num = rd.randint(1, 100)
#把randint重命名为rdn
from random import randint as rdn
num = rdn(1, 100)
```

6. reload()重新导入一个模块
   reload()内建函数
   可以重新导入一个已经导入的模块。它的语法如下：
   reload(module)
   module是你想要重新导入的模块。使用reload的时候有一些标准。首先模块必

须是全部导入（不是使用from-import），而且它必须被成功导入。另外reload函数的参数必须是模块自身而不是包含模块名的字符串，也就是说类似reload(sys)而不是reload("sys")。
一般用法，就是改变文件中的字符编码时用到：

```
import sys
reload(sys)
sys.setdefaultencoding("utf-8")
```

三、包相关知识

包是一个有层次的文件目录结构，它定义了一个由模块和子包组成的python应用程序执行环境。包主要用来帮助解决如下问题：

1. 为平坦名称空间假如有层次的组织结构；
2. 允许程序员把有联系的模块组合到一起；
3. 允许分发者使用目录结构而不是一大堆混乱的文件；
4. 帮助解决有冲突的模块名称。
   与类和模块相同，包也使用句点属性标识符（点号）来访问他们的元素。使用标准的import和from-import语句导入包中的模块。
   包的目录结构
   包其实就是一个目录里面多个一个**init**.py文件，当我们导入包的时候会自动的执行**init**.py里面得代码。一般情况下，如果我们不需要在包导入的时候就执行代码的话，那么**init**.py文件留空即可。

```
mypackage/
    __init__.py
    a.py
    b.py
    subpackage/
        __init__.py
        suba.py
        subb.py
        ......
```

作业

1. 理解模块和包的概念
2. 知道如何导入模块和包
3. 知道如何使用模块和包中的属性和方法