模块

1

模块

——模块和文件

什么是模块

- 模块支持从逻辑上组织python代码
- 当代码量变得相当大的时候, 最好把代码分成一些有组织的代码段
- 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

模块文件

- 说模块是按照逻辑来组织python代码的方法，文件是物理层上组织模块的方法
- 一个文件被看作是一个独立模块，一个模块也可以被看作是一个文件
- 模块的文件名就是模块的名字加上扩展名.py

名称空间

- 名称空间就是一个从名称到对象的关系映射集合
- 给定一个模块名之后，只可能有一个模块被导入到python解释器中,所以在不同模块间不会出现名称交叉现象
- 每个模块都定义了它自己的唯一的名称空间

```
>>> import foo
>>> import bar
>>> print(foo.hi)          #调用foo模中的hi变量
hello
>>> print(bar.hi)          #调用bar模块中的hi变量
greet
```

2

模块

——导入模块

搜索路径

- 模块的导入需要一个叫做“路径搜索”的过程
- python在文件系统“预定义区域”中查找要调用的模块
- 搜索路径在sys.path中定义

```
>>> import sys
>>> print(sys.path)
['', '/usr/local/lib/python36.zip', '/usr/local/lib/python3.6', '/
usr/local/lib/python3.6/lib-dynload', '/usr/local/lib/
python3.6/site-packages']
```

模块导入方法

- 使用import导入模块
- 可以在一行导入多个模块，但是可读性会下降
- 可以只导入模块的某些属性
- 导入模块时，可以为模块取别名

```
>>> import time, os, sys
>>> from random import choice
>>> import pickle as p
```

导入和加载

- 当导入模块时，模块的顶层代码会被执行
- 一个模块不管被导入（import）多少次，只会被加载（load）一次

```
[root@py01 ~]# cat foo.py
hi = 'hello'
print(hi)
[root@py01 ~]# python3
>>> import foo
Hello                    #第一次导入，执行print语句
>>> import foo           #再次导入，print语句不再执行
>>>
```

3

模块

——包

目录结构

- 包是一个有层次的文件目录结构，为平坦的名称空间加入有层次的组织结构
- 允许程序员把有联系的模块组合到一起
- 包目录下必须有一个__init__.py文件

```
phone/
        __init__.py
        common_util.py
         voicedata/
                    __init__.py
                    post.py
```

绝对导入

- 包的使用越来越广泛，很多情况下导入子包会导致和真正的标准库模块发生冲突
- 因此，所有的导入现在都被认为是绝对的，也就是说这些名字必须通过python路径（sys.path或PYTHONPATH）来访问

相对导入

- 绝对导入特性使得程序员失去了import的自由，为此出现了相对导入
- 因为import语句总是绝对导入的，所以相对导入只应用于from-import语句

```
[root@py01 ~]#  ls -R phone/
phone/:
common_util.py  __init__.py  voicedata
phone/voicedata:
__init__.py  post.py

[root@py01 ~]# cat phone/voicedata/post.py
from .. import common_util
```

更多精彩...

http://bj.linux.tedu.cn/
企业QQ： 86198501