模块

模块概述

模块的英文是modules，可以认为是一盒主题积木，通过它可以拼出某个主题的东西，这个与函数不同，一个函数相当于一个积木，而一个模块包含很多函数，也就是很多积木，所以说模块相当于一盒积木。

在Python中，一个扩展名为'.py'的文件就称为一个模块。

自定义模块

在python中，自定义模块有两个作用，一个是规范代码，让代码易于理解；另一方面方便其他程序使用已经编好的代码，提高开发效率。

1. 创建模块

   创建模块就是创建一个py文件。比如我创建一个tango.py

   ```
   '''自定义模块'''

   def hello(name):
       '''向指定的人打招呼'''
       print("{}你好啊!".format(name))
   ```

   文件名为tango.py

2. 使用import语句导入模块

   如果想用其他模块中的内容，需要先将模块导入,可以使用import导入。

   ```
   import tango #导入tango自定义模块

   # 调用hello方法
   tango.hello("路飞")
   ```

   ```
   PS F:\笔记相关\Python基础第二次分享\day10> python .\demo.py
   路飞你好啊!
   PS F:\笔记相关\Python基础第二次分享\day10>
   ```

   我们并没有在demo.py中写hello这个函数，但是可以看到这个函数执行了。

   如果我们要导入的模块的名字过长，也可个给它起一个别名。

   ```
   import tango as t

   t.hello("娜美")
   ```

```
PS F:\笔记相关\Python基础第二次分享\day10> python .\demo1.py
娜美你好啊！
PS F:\笔记相关\Python基础第二次分享\day10>
```

3. 使用form...import语句导入模块

   如果我们想导入指定模块中指定的方法，可以通过这种方式来实现。

```
from tango import sayBye # 从tango模块中导入sayBye方法

sayBye('乔巴')
```

```
PS F:\笔记相关\Python基础第二次分享\day10> python .\demo2.py
乔巴再见
PS F:\笔记相关\Python基础第二次分享\day10>
```

   我们看到，这时不要通过模块名就可以使用模块中的函数了。

   也可以通过

```
from tango import *
```

   的方式将说有的函数都导入进来。

4. 模块搜素目录

   当使用import语句导入模块是，默认情况下会按照一下顺序查找

   - 在当前目录(即执行的Python脚本所在的目录)下查找
   - 到PYTHONPATH（环境变量）下的每一个目录中查找
   - 到Python的默认安装包下查找

   以上的各个目录的保存位置在标准模块sys的path变量中可以查看到

```
import sys
print(sys.path)
```

```
['F:\\笔记相关\\Python基础第二次分享\\day10', 'D:\\ProgramData\\Anaconda3\\python37.z
ip', 'D:\\ProgramData\\Anaconda3\\DLLs', 'D:\\ProgramData\\Anaconda3\\lib', 'D:\\Prog
ramData\\Anaconda3', 'D:\\ProgramData\\Anaconda3\\lib\\site-packages', 'D:\\ProgramDa
ta\\Anaconda3\\lib\\site-packages\\win32', 'D:\\ProgramData\\Anaconda3\\lib\\site-pac
kages\\win32\\lib', 'D:\\ProgramData\\Anaconda3\\lib\\site-packages\\Pythonwin']
```

   这个是我的目录

以主程序形式执行

如果有多个模块，当你想指定某个模块为主模块时需要在该模块的里面加入如下代码:

```
hello = "你好啊朋友" # 定义一个全局变量
def read():
    '''看书的功能'''
    hello = '你好啊朋友，一起看书吧。'
    print(hello)

if __name__ == "__main__":
    print("我去书店。。。。")
    read()
    print("我回家...")
    hello = "吃饭。。。"
    print(hello)
```

```
PS F:\笔记相关\Python基础第二次分享\day10> python .\demo3.py
我去书店。。。。
你好啊朋友，一起看书吧。
我回家...
吃饭。。。
PS F:\笔记相关\Python基础第二次分享\day10>
```

如果别的模块想导入这个模块，会发生什么呢?

```
import demo3
print(demo3.hello) #全局变量
```

```
PS F:\笔记相关\Python基础第二次分享\day10> python .\demo3_test.py
你好啊朋友
PS F:\笔记相关\Python基础第二次分享\day10>
```

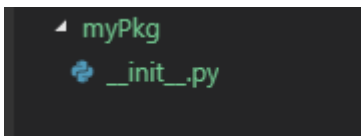
我们看到智能获取到全局变量，而其他的方法是没有运行的，大家可以吧

```
if __name__ == "__main__":
```

去掉，并把缩进往前移动一下，看看结果。

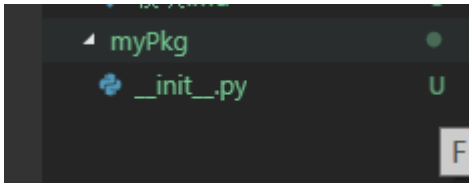
Python中的包

1. Python程序的包结构



2. 创建和使用包

   创建包就是创建一个文件夹，并设置一个 __init__.py的文件，里面什么都不用写



这里的myPkg就是一个python包了。

我们可以通过

```
import 报名.模块
```

的方式来使用包中的内容。

一个包里面可以包含多个模块，这也是为了方便代码的管理。

引用其他模块

1. 导入和使用标准模块

   python自带了很多标准库，在你安装完Python环境的时候，就安装好了，这些标准库可以直接来用，如

   ```
   import random # 导入标准库random，用来生成随机数
   ```

   常用的标准库

模块名	描述
sys	与Python解释器以及环境有关的库
time	提供时间相关的函数
os	提供访问系统的库
calendar	提供与日期相关的各种函数的库
urllib	用来读取来自网站上的数据的库
json	用于使用json序列化和反序列化
re	正则表达式相关
math	数学计算相关
decimal	用于控制运算精度等操作
shutil	高级文件操作，复制，移动等
logging	记录日志文件
tkinter	用于图形界面GUI

2. 第三方模块的下载和安装

   可以通过pip install 包名来安装，指定的包。如：

```
pip install numpy
```