模块概念

- 模块是一个包含Python定义和语句的文件，文件名就是模块名后跟文件后缀 .py

模块导入

- 模块可以被别的程序引入，以使用该模块中定义的变量和函数等功能
- 习惯上（但不强制要求）把所有 import 语句放在模块的开头
- 一个模块被另一个程序导入时，会执行该模块
- 一个模块只会被另一个程序导入一次

模块导入
import module1[, module2[,... moduleN]]
import module1 as alias1[, module2 as alias2[,... moduleN as aliasN]]
from module import item1[, item2[,... itemN]]
from module import item1 as alias1[, item2 as alias2[,... itemN as aliasN]]
from module import *

注意：请慎用 from module import * ，很容易出现名称重复的情况，导致出现一些意外的问题

模块搜索路径

- sys 模块的 path 变量包含了 Python 解释器自动查找所需模块的路径的列表
- 如果这些路径都找不到，则会报错：ModuleNotFoundError: No module named 'xxx'

```
import sys

print(sys.path)
```

__name__属性

- 每个模块都有一个__name__属性，当其值是 '__main__' 时，说明该模块自身在运行，否则说明该模块被导入，其值为模块名
- 在完成一个模块的编写之前，我们一般会对模块中的功能进行测试，看看各项功能是否正常运行。对于这些测试的代码，我们希望只在直接运行这个py文件的时候执行，而在用其他的程序导入这个模块的时候不要执行。这个时候就可以借助__name__属性来实现

```
print('在该模块自身运行时会执行')
print('在该模块被导入时也会执行')


if __name__ == '__main__':
    print('在if语句下的程序块仅在该模块自身运行时才执行')
else:
    print('在else子句下的程序块在该模块自身运行时不会执行')
    print('但是在被导入时，会执行')
```

包的概念

- Python包实际上就是一个文件夹，只是该文件夹里面一定包含 __init__.py 模块
- 和文件夹一样，包里面还可以装其他的包

包的作用

- 避免相同命名冲突： 如果在同一个包里，是不允许两个模块命名相同的，但是如果不在同一个包里，是可以的
- 模块分区： 把不同功能的模块归类到不同的包里，方便查询和修改。在比较大型的项目中常常需要编写大量的模块，此时我们可以使用包来对这些模块进行管理

包的导入

包的导入
import package1[, package2[,... packageN]]
import package1 as alias1[, package2 as alias2[,... packageN as aliasN]]
from package import module1[, module2[,... moduleN]]

包的导入
from package import module1 as alias1[, module2 as alias2[,... moduleN as aliasN]
from package.module import item1[, item2[,... itemN]]
from package.module import item1 as alias1[, item2 as alias2[,... itemN as aliasN]]