模块中的数据隐藏：Python中没有C++/C#中public,private的概念。所谓数据的隐藏是一种惯例。

在变量前面加下划线\_，可以在一定程度上（不能完全有效）隐藏数据。当通过from \*语句导入模块时，可以隐藏变量。

例：程序module\_test/hide.py

\_x = 5  
x = 6

程序module\_test/module\_test7.py

# import hide  
# print("x ", hide.x) # ok, 6  
# print("\_x ", hide.\_x) # ok, 5  
  
# from hide import \_x  
# print("\_x ", \_x) # ok, 5  
  
from hide import \*  
print("x ", x) # ok, 6  
# print("\_x ", \_x) # NameError: name '\_x' is not defined

可以看出，变量名前加下划线，只对from\*语句起作用。

\_\_name\_\_属性：

例：程序module\_test/name.py

def tester():  
 print("It's Christmas in Heaven...")  
  
  
# 如果以顶层脚本执行name.py  
# \_\_name\_\_就为\_\_main\_\_  
if \_\_name\_\_ == '\_\_main\_\_':  
 tester()

程序module\_test/module\_test8.py

import name  
  
print('\_\_name\_\_ test')  
name.tester()

输出为：

\_\_name\_\_ test

It's Christmas in Heaven...

从输出可以看出，import name时，并未执行name.py中的tester()函数，name.py中的\_\_name\_\_不是\_\_main\_\_，如果直接执行name.py，则其\_\_name\_\_就是\_\_main\_\_。

import和from的as扩展

import modulename as name

相当于：

import modulename

name = modulename

del modulename

例：程序module\_test/module\_test9.py

import small as sm  
from name import tester as func  
  
# as扩展之后，只能使用as之后的变量名来引用该模块  
# print(small.x) # Error，无法使用small  
print(sm.x)  
  
# tester() # Error  
func()