day10 Python基础

**第一集 模块后续**

A、\_\_all\_\_的作用

在昨天的模块部分，如果B.py调用了A.py，那么A.py中的所有东西都将被B.py调用；这很不安全。使用\_\_all\_\_变量就能让A.py限制自己哪些东西可以被使用哪些东西不能被使用。

A.py中如下：

\_\_all\_\_=[“test”,”Test”]

#在列表中写上要被外界使用的方法名，全局变量名或者类名等；表示这些东西能被外界使

#用，没有写的东西不能被外界使用。(视频中说的是通过from A import \*这种方式才能限制，自行测试import A的方式能不能被限制)

def test():

print(“test”)

def test1():

print(“test1”)

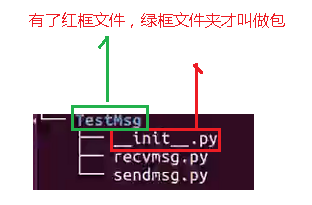
num=100

class Test:

pass

B、包

把几个.py文件放进某个文件夹下，此时这个文件夹就是一个包。但是此时python解释器是不认这个文件夹为包的，还需要在这个文件夹下创建一个\_\_init\_\_.py的文件才认识；也就是说\_\_init\_\_.py文件让这个文件夹变成了解释器能认识的包。此文件在外界导入其所在的包时执行。



\_\_init\_\_.py文件内容如下：

\_\_all\_\_=[“sendmsg”] #在这里指定其他文件能用这个包中的哪些模块

在其他文件中如下(此文件与TestMsg同路径)：

from TestMsg import \*

#表示导入TestMsg 包中的除了\_\_init\_\_.py限制之外的所有模块。

recvmsg.test1()

#recvmsg模块中有test1方法。不能使用，因为\_\_init\_\_.py中没让外界使用recvmsg方法

sendmsg.test2()

#sendmsg模块中有test2方法。可以使用，因为\_\_init\_\_.py中让外界使用sendmsg方法

此外\_\_init\_\_.py文件加入：

from . import sendmsg #点表示相同文件夹下

那么其它文件中就可以如下直接调用模块中的方法：

TestMsg.sendmsg.test2() #包名.模块名.方法名