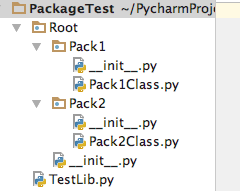
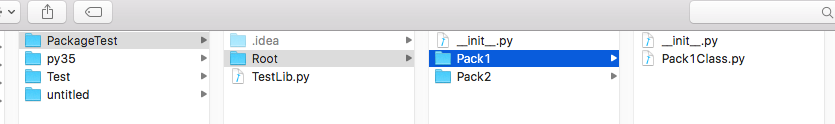
[**Python包中\_\_init\_\_.py作用**](https://www.cnblogs.com/AlwinXu/p/5598543.html)

在创建python包的过程中，IDE都会在包根目录下创建一个\_\_init\_\_.py文件，该Python文件默认是空的.目录结构如下：

Pycharm下的package树结构：



在Finder中的目录结构：



从Finder中的目录就可以看出来，每个package实际上是一个目录（Directory),那么IDE是怎么识别它为package呢？没错，\_\_init\_\_.py的第一个作用就是package的标识，如果没有该文件，该目录就不会认为是package。

Python中的包和模块有两种导入方式：精确导入和模糊导入：

精确导入：

from Root.Pack1 import Pack1Class

import Root.Pack1.Pack1Class

模糊导入：

from Root.Pack1 import \*

模糊导入中的\*中的模块是由\_\_all\_\_来定义的，\_\_init\_\_.py的另外一个作用就是定义package中的\_\_all\_\_，用来模糊导入，如\_\_init\_\_.py：

\_\_all\_\_ = ["Pack1Class","Pack1Class1"]

在包外部调用：

from Root.Pack1 import \*

a = Pack1Class.Pack1\_AA("Alvin")

a.PrintName()

\_\_init\_\_.py首先是一个python文件，所有还可以用来写python模块，但是不建议这么写，尽量保证\_\_init\_\_.py足够轻：

\_\_init\_\_.py:

[复制代码](javascript:void(0);)

\_\_all\_\_ = ["Pack1Class","Pack1Class1","Init\_AA"]

class Init\_AA:

def \_\_init\_\_(self,name):

self.name = name

def Greeting(self):

print("Hello ",self.name)

[复制代码](javascript:void(0);)

在测试中调用：

from Root.Pack1 import \*

b = Init\_AA("test")

b.Greeting()

总结：

从上边的例子可以看出，\_\_init\_\_.py的主要作用是：

1. Python中package的标识，不能删除

2. 定义\_\_all\_\_用来模糊导入

3. 编写Python代码(不建议在\_\_init\_\_中写python模块，可以在包中在创建另外的模块来写，尽量保证\_\_init\_\_.py简单）