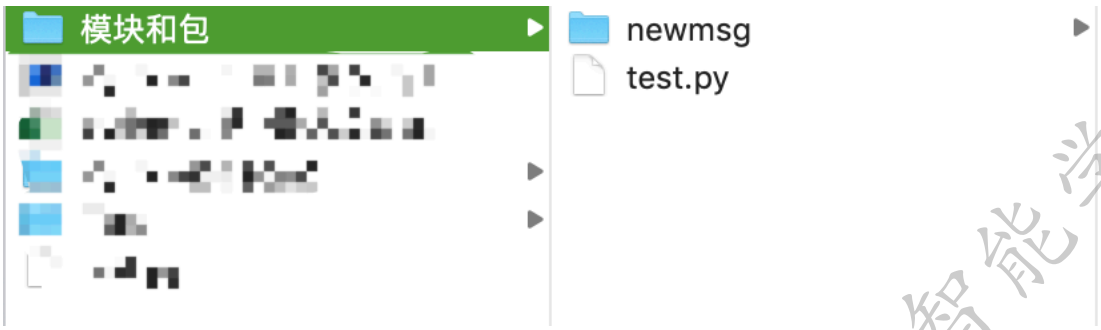
包的使用

一个模块就是一个 py 文件，在 Python 里为了对模块分类管理，就需要划分不同的文件夹。多个有联系的模块可以将其放到同一个文件夹下，为了称呼方便，一般把 Python 里的**一个代码文件夹称为一个包。**

1. 导入包的方式

现有以下包 `newmsg` ,包里由两个模块，分别是 `sendmsg.py` 、 `recvmsg.py` 文件。在包的上级文件夹里，有一个 `test.py` 文件，目标是在 `test.py` 文件里引入 `newmsg` 的两个模块。

目录结构如下图所示：



sendmsg.py文件里的内容如下:

```
def send_msg():
  print('------sendmsg方法被调用了-------')
```

recvmsg.py文件里的内容如下:

```
def recv_msg():
  print('-----recvmsg方法被调用了--------')
```

可以使用以下几种方式来导入模块，使用模块里的方法。

1>. 直接使用包名.模块模块名导入指定的模块。

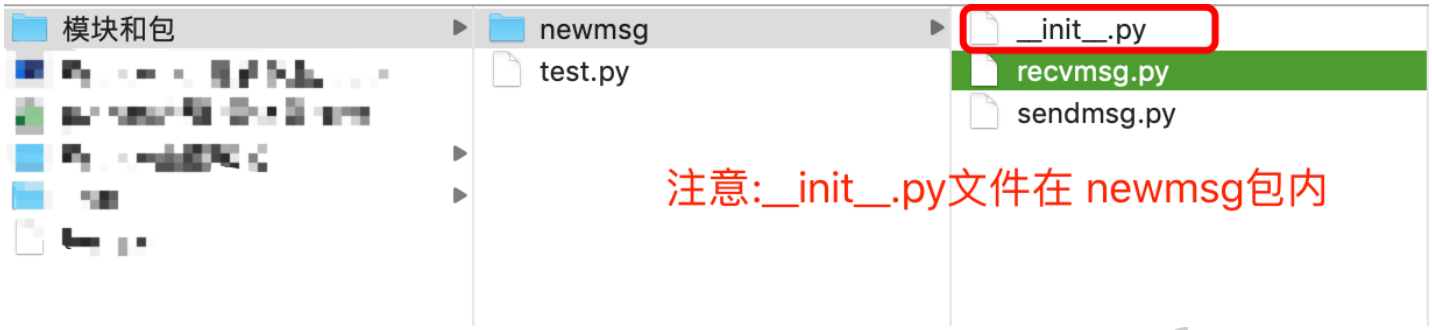
2>. 使用 `from xxx import xxx` 方式导入指定模块。

```python
from newmsg import sendmsg,recvmsg
sendmsg.send_msg()
recvmsg.recv_msg()
# newmsg.sendmsg.send_msg() 此时，需要直接写模块名，不能再写包名
```

使用 from import 语句，导入包里的指定模块

3>. 使用 `__init__.py` 文件，导入包里的指定模块。

可以在 `newmsg` 里创建 `__init__.py` 文件，在该文件里导入指定的内容。



在 `__init__.py` 文件里编写代码:

```python
from . import sendmsg  # 导入指定的模块     . 代表的是当前文件夹
```

test.py文件里的代码

```python
import newmsg # 导入时，只需要输入包名即可。在包名的__init__.py文件里，导入了指定模块
newmsg.sendmsg.sendm_msg()  # 可以直接调用对应的方法
# newmsg.recvmsg.recv_msg()   不可以使用 recvmsg 模块，因为 __init__.py文件里没有导入这个模块
```

4.> 使用 `__init__.py` 文件，结合 `__all__` 属性，导入包里的所有模块。

在 `newmsg` 包里的 `__init__.py` 文件里编写代码:

```python
__all__ = ["sendmsg","recvmsg"]  # 指定导入的内容
```

test.py文件代码:

```python
from newmsg import *  # 将newmsg里的__inint__.py文件里,__all__属性对应的所有模块都导入
sendmsg.sendmsg()
recvmsg.recvmsg()
```

总结

- 包将有联系的模块组织在一起，即放到同一个文件夹下，并且在这个文件夹创建一个名字为 `__init__.py` 文件，那么这个文件夹就称之为 `包`

- 有效避免模块名称冲突问题，让应用组织结构更加清晰

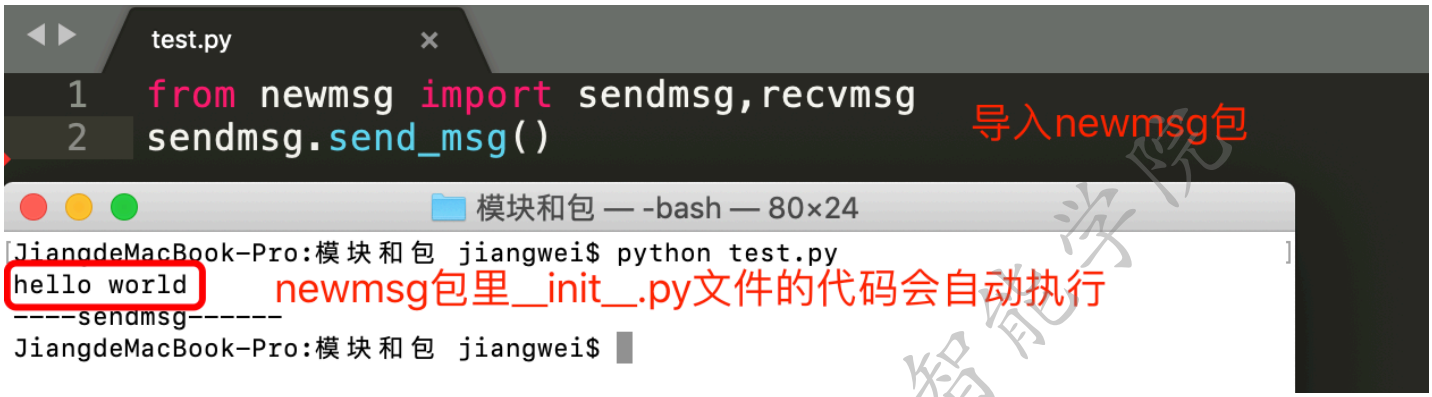
2. `__init__.py` 文件有什么用

`__init__.py` 控制着包的导入行为。`__init__.py` 为空仅仅是把这个包导入，不会导入包中的模块。可以在 `__init__.py` 文件中编写内容。

`newmsg/__init__.py` 文件：

```
print('hello world')
```

别的模块在引入这个包的时候，会自动调用这段代码。



3. `__all__`

在 `__init__.py` 文件中，定义一个 `__all__` 变量，它控制着 from 包名 import *时导入的模块。

`newmsg/__init__.py` 文件：

```
__all__ = ['sendmsg','recvmsg']
```