**新建package**

**举例如下：**

* 一个结构良好的\_\_init\_\_.py文件，仅为一个非常重要的目的来服务：从子模块导入。你的\_\_init\_\_.py应该看起来像这个样子：

*# ORDER MATTERS HERE -- SOME MODULES ARE DEPENDANT ON OTHERS*

*# 导入顺序要考虑——一些模块会依赖另外的一些*

**from** exceptions **import** FSQError, FSQEnvError, FSQEncodeError,

FSQTimeFmtError, FSQMalformedEntryError,

FSQCoerceError, FSQEnqueueError, FSQConfigError,

FSQPathError, FSQInstallError, FSQCannotLockError,

FSQWorkItemError, FSQTTLExpiredError,

FSQMaxTriesError, FSQScanError, FSQDownError,

FSQDoneError, FSQFailError, FSQTriggerPullError,

FSQHostsError, FSQReenqueueError, FSQPushError

*# constants relies on: exceptions, internal*

**import** constants

*# const relies on: constants, exceptions, internal*

**from** **const** **import** **const**, set\_const *# has tests*

*# path relies on: exceptions, constants, internal*

**import** path *# has tests*

*# lists relies on: path*

**from** lists **import** hosts, queues

*#...*

**一、构建文件夹结构**

* 每个文件夹中都放入一个 **init**.py 文件

**二、\_*init*\_.py 文件**

* 使用\_\_init\_\_.py来限制导入顺序
* 文档字符串，以及在软件包层面对\_\_all\_\_属性的赋值应当是\_\_init\_\_.py中唯一与导入无关的代码：

**1 文档字符串**

''' **Analysis** **EEG** data **with** machine learning **method** through **Python**.

**This** package contains six sub-packages:

dom -- **The** **W3C** **Document** **Object** **Model**. **This** supports **DOM** **Level** 1 +

**Namespaces**.

'''

**2 \_\_all\_\_属性**

\_\_all\_\_ = ["eeg\_prepro", "eeg\_feature", "feature\_prepro", "ml", "chart", "table"]

**3 异常处理**

try:

from pandas import hashtable, tslib, lib

except ImportError as e: # **pragma**: **no** cover

**module** = **str**(**e**).lstrip('cannot import name ') # hack but overkill **to** **use** re

**raise** ImportError("C extension: {0} not built. If you want to import "

"pandas from the source directory, you may need to run "

"'python setup.py build\_ext --inplace' to build the C "

"extensions first.".**format**(**module**))

**4 导入模块**

**from** datetime **import** datetime

**import** numpy **as** np

**from** pandas.info **import** \_\_doc\_\_

**三、异常处理**

* 使用一个模块来定义所有的异常

1. 你应该为你的软件包的异常定义一个基类:

**class** **APackageException**(Exception):

'''root for APackage Exceptions, only used to except any APackage error, never raised'''

**pass**

1. 然后确保你的软件包在任何错误状态下，只会引发这个基类异常的子类异常，这样如果你需要的话，你就可以阻止全部的异常：

*# from fsq*

**class** **FSQEnvError**(FSQError):

'''An error if something cannot be loaded from env, or env has an invalid

value'''

**pass**

**class** **FSQEncodeError**(FSQError):

'''An error occured while encoding or decoding an argument'''

**pass**

*# ... and 20 or so more*

1. 在你的异常处理中保持更大的粒度，有利于让程序员们在一个try/except中包含越来越大的，互相不干涉的代码段。

*# this*

**try**:

item = fsq.senqueue('queue', 'str', 'arg', 'arg')

scanner = fsq.scan('queue')

**except** FSQScanError:

'''do something'''

**except** FSQEnqueueError:

'''do something else'''

*# not this*

**try**:

item = fsq.senqueue('queue', 'str', 'arg', 'arg')

**except** FSQEnqueueError:

'''do something else'''

**try**:

scanner = fsq.scan('queue')

**except** FSQScanError:

'''do something'''

*# and definitely not*

**try**:

item = fsq.senqueue('queue', 'str', 'arg', 'arg')

**try**:

scanner = fsq.scan('queue')

**except** FSQScanError:

'''do something'''

**except** FSQEnqueueError:

'''do something else'''