【白话python连载（7.2）】python的模块介绍—轻松实现复用

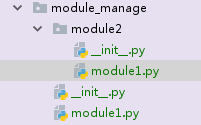
* python中的模块

模块的定义与生成

python模块的**是**一个包含定义的变量、函数和类的文件，其后缀名是.py。

特点：模块可以被其他程序导入，以使用该模块中定义的类，函数是模块中对应类的方法，而变量则是类的属性。

为了防止不同模块中变量名的重复和冲突，python中通常采用“包”即文件目录结构来管理不同的模块。只要顶层的目录文件名不同，就可以防止模块或模块中变量或函数名定义相同而冲突的问题。两个module1在使用中依次为：module\_manage,module2.module1；module\_manage.module1

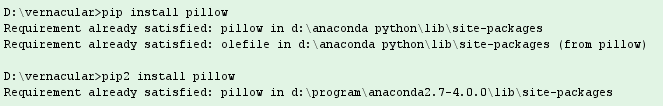


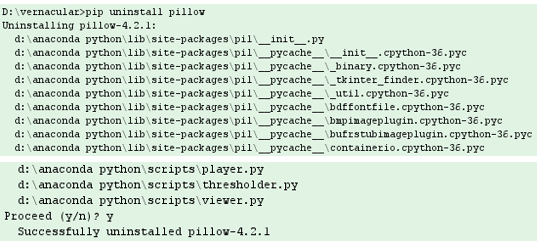
如module\_manage这个python包，与常规的文件夹所不同的是，需要额外添加一个\_\_init.py\_\_的模块（文件中可以不添加任何内容）来表明这个文件是python的包。如果使用pycharm编辑器，在新建包时将会自动完成\_\_init.py\_\_的模块的添加。而且在一个python包中，也可以嵌套其他的python包，方便模块功能的实现与复用。

*模块的安装和应用*

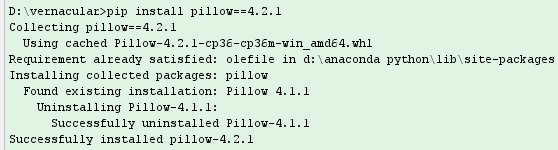
对于python自带的模块可以直接导入使用，而对于第三方模块的使用，需要在python环境中提前安装好要使用的模块。常规的安装方法直接使用python的pip模块（在初始化安装python编辑器的过程中，勾选后已经自动安装），安装方式：pip install module\_name。卸载方式：pip uninstall module\_name。

如pillow这个模块，是之前已经成功安装过的，所以再次安装时则会提示对应的模块已经存在。因此先将其卸载再尝试安装一下。因为电脑中有两个版本的python，因此为了对应管理不同版本中的模块可以重命名pip，进行不同版本中的模块安装。

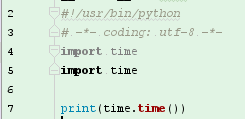




在安装过程中指定模块的版本号

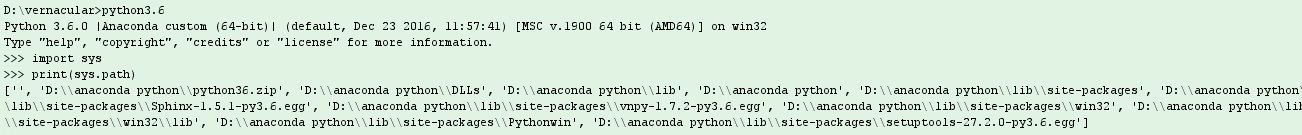


安装好的模块，使用 import module/from module import name1，name2 可以导入完整模块或者是模块的部分属性，而为了更方便的使用模块也可以再导入模块中另外给模块进行命名，如在python对数据的处理中经常使用的两个模块import numpy as np和import pandas as pd。在python中重复导入同一个模块，将只会以最后一次导入为主，这样可以减少查找重复模块的次数。

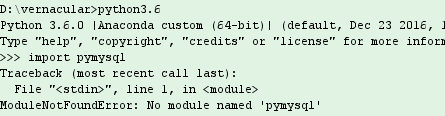


import的工作机制

使用import是在环境变量中依次查找是否有对应的python模块，使用自带的sys.path功能可以查看模块所存在的位置。



使用import将会在这些地方查找需要的模块，如果没有找到则会报错。找到的话则会将其编译成字节码也就是.pyc文件。然后会执行模块中的代码，并创建其中所需要的对象。



*模块的打包与发布*

如果要将完成的模块进行打包和发布的话，需要进行前期准备工作：以一个dist\_module.py模块为例。该模块实现的功能是输出当前时间和指定天数前后的时间。具体代码为：

from datetime import \*

def time\_manage(n):

today = datetime.now()

print("当前日期是：" ,today)

n = int(n)

memory = today + timedelta(days=n)

print(n,"天后的日期是：" , memory)

if \_\_name\_\_ == '\_\_main\_\_':

n = input("请输入你要查询的天数：")

time\_manage(n)

在完成模块下添加一个平行的setup.py的模块的，使用distutils来发布模块，在setup.py文件中，添加如下内容：

from distutils.core import setup

setup(

name='dist\_module',

version='1.0.0',

py\_modules = ['dist\_module'],

author='ziyi',

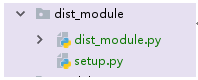
author\_email='635516607@qq.com',

url='1',

description='计算n天左右的时间'

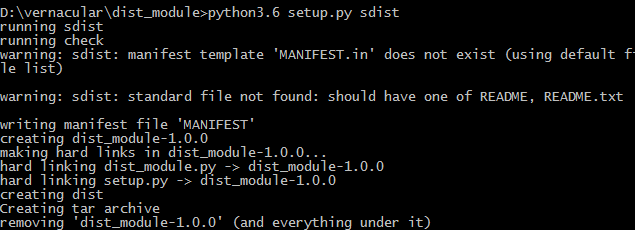
)

然后将对应完成的dist\_module.py和setup.py模块放入同一个文件中，该文件夹命名与dist\_module.py模块的名称一致，即dist\_module。目录结构如下：

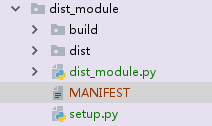


前期准备工作完成后，就可以进行模块的打包和发布了：

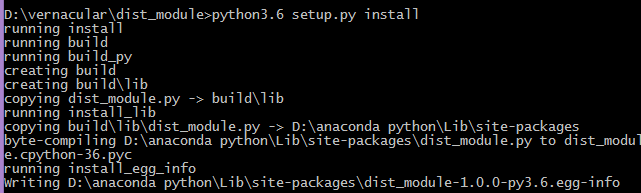
在dist\_module.py文件下打开命令窗口：否则的话给定setup.py文件的具体位置。然后执行python setup.py sdist。完成模块的打包工作，为对应的模块创建一个源码包。



模块打包结束后对应的文件目录变为如下：



然后再将该模块添加到本地的python包管理空间中，使用python setup.py install。

之后即可在其他文件中导入该模块进行复用。

**#!/usr/bin/python**

**# -\*- coding: utf-8 -\*-**

**import dist\_module**

**print(dist\_module.time\_manage(6))**

**输出结果**

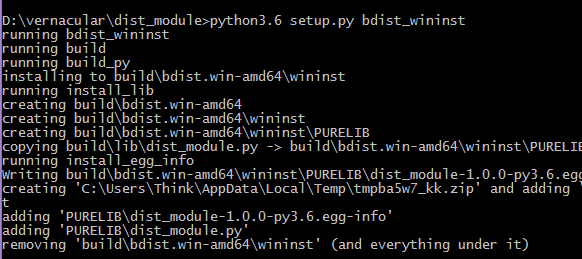
**D:/vernacular/module\_mana/module\_manage/module1.py**

**当前日期是： 2018-02-08 17:17:58.274428**

**6 天后的日期是： 2018-02-14 17:17:58.274428**

**Process finished with exit code 0**

**在windows系统下，生成可执行文件D:\anaconda python\** **python3.6.exe**



D:\JavaScript>python3.6

