第十一章：模块与包

Table of Contents

[1. 序言 2](#_Toc52863185)

[1.1 章节简介 2](#_Toc52863186)

[2. 使用标准库模块 2](#_Toc52863187)

[3. 自定义模块和使用 4](#_Toc52863188)

[4. 安装第三方模块 6](#_Toc52863189)

# 序言

## 章节简介

Python中的模块是什么？简而言之，在Python中，一个文件（以“.py”为后缀名的文件）就叫做一个模块，每一个模块在Python里都被看做是一个独立的文件。模块可以被项目中的其他模块、一些脚本甚至是交互式的解析器所使用，它可以被其他程序引用，从而使用该模块里的函数等功能，使用Python中的标准库也是采用这种方法。

在计算机程序的开发过程中，我们可能会写很多的代码，据统计一名google工程师每天提交的代码大概在100-150行左右。而一个大型项目有上百名工程师是很平常的规模。显然，我们不能将所有的函数和类的定义只写在一个文件里。为了编写可维护并容易被调用的代码，我们把很多函数和类分组，分别放到不同的文件里，这样，每个文件包含的代码就相对较少，很多编程语言都采用这种组织代码的方式。在Python中，一个.py文件就称之为一个模块（Module）。使用模块有什么好处？最大的好处是大大提高了代码的可维护性。其次，编写代码不必从零开始。当一个模块编写完毕，就可以被其他地方引用。我们在编写程序的时候，也经常引用其他模块，包括Python内置的模块和来自第三方的模块。本章节主要有以下一些内容：

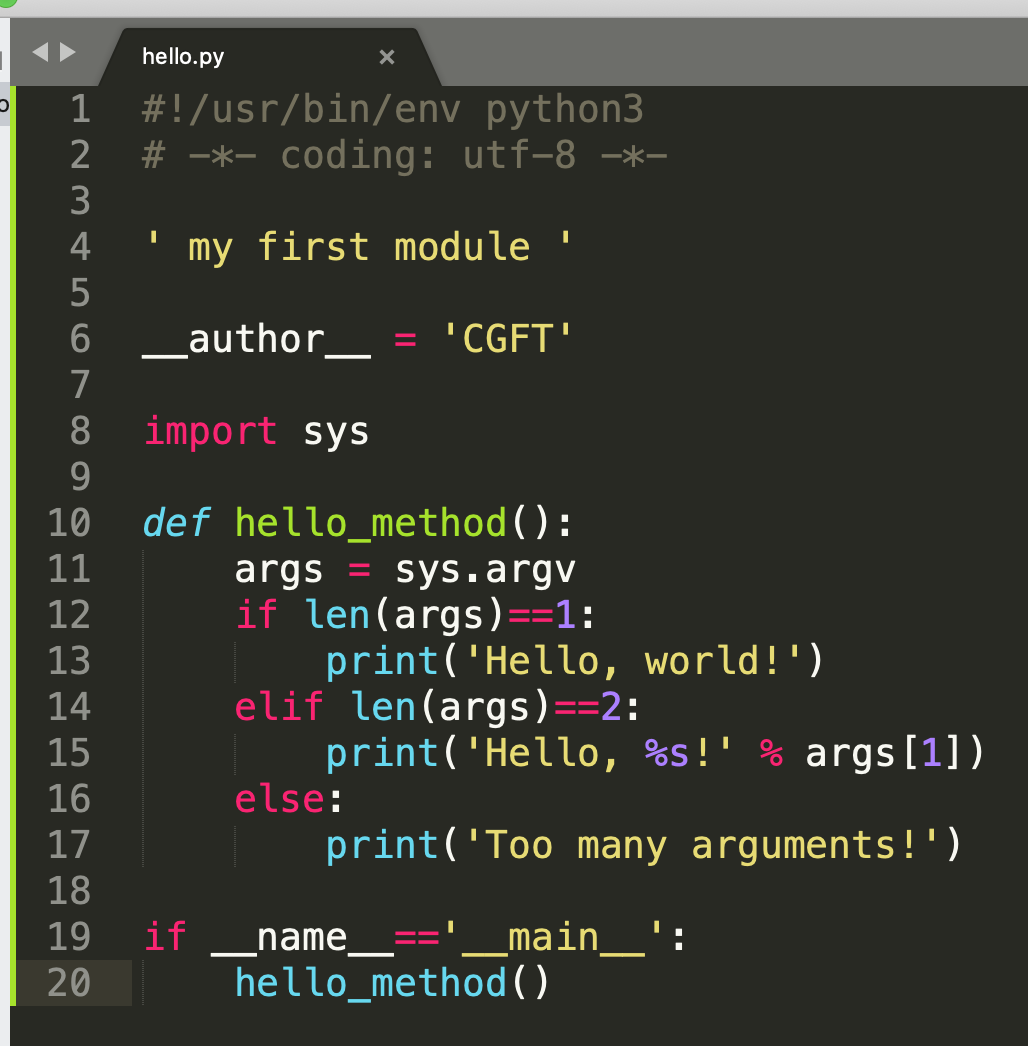
* 使用标准库模块
* 自定义模块和使用
* 安装第三方模块

# 使用标准库模块

我们先来看看标准库的使用的示例，本小节中我们首先新建一个文件夹叫做mymodule，再在该文件夹中新建一个叫做hello.py的文件。

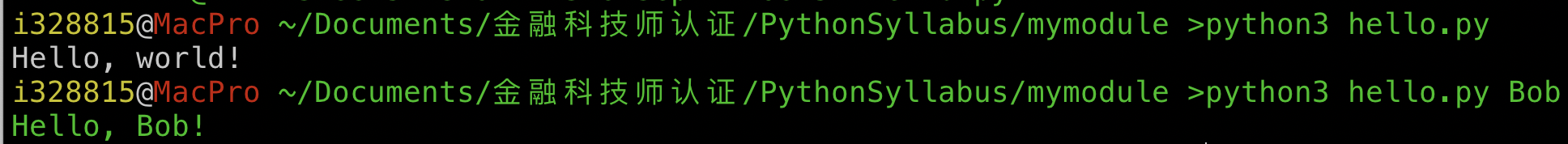


用编辑器打开hello.py文件，输入以下内容。



我们先忽略前面的部分，直接从代码的第8行看起，sys模块是Python内置的模块，当我们需要使用该模块的功能时，首先的第一步就是使用import导入模块。导入sys模块后，我们就有了变量sys指向该模块，利用sys这个变量，就可以访问sys模块的所有功能。接下来我们定义了一个方法hello\_method(),在第11行我们使用了sys.argv，argv是sys模块中的一个变量，它可以存储我们通过命令行调用python文件时的参数，所有的参数都是以list集合的方式保存在argv变量里。argv至少有一个元素，因为第一个参数永远是该.py文件的名称。如果我们在windows的CMD命令行或者Linux和Mac的终端里输入运行***python3 hello.py***， 那么argv保存就是['hello.py']，运行***python3 hello.py Anthony***获得的sys.argv就是['hello.py', 'Anthony']。在得到argv变量后我们赋值给变量args，并继续执行代码，如果args有一个参数我们就打印”Hello, world!”,如果有2个参数我们就执行*print('Hello, %s!' % args[1])*。继续看代码的19行，当我们在命令行运行hello模块文件时，Python解释器把一个特殊变量\_\_name\_\_置为\_\_main\_\_，并在第20行调用方法hello\_method()。而如果在其他地方导入该hello模块时，if判断将失败代码什么也不会做。

说了这么多，让我们运行一下该文件看看效果。下面是在MAC的终端中执行代码开到的信息，在windows的CMD终端里输入相同命令可以看到相同的内容：

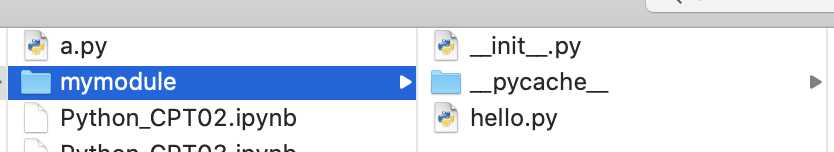


# 自定义模块和使用

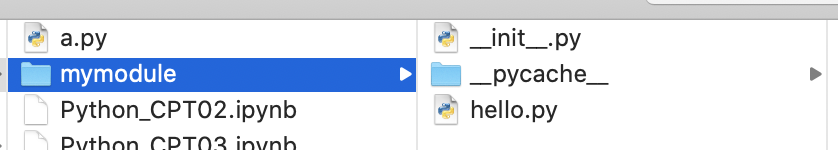
在了解如何在自己的代码里调用Python内置模块后，我们再看看如何自定义自己的模块。我们现在来关注下上一小节实现的.py文件里的前几行代码。第1行和第2行是标准注释，第1行注释可以让这个hello.py文件直接在Unix/Linux/Mac上运行，第2行注释表示.py文件本身使用标准UTF-8编码；第4行是一个字符串，表示模块的文档注释，任何模块代码的第一个字符串都被视为模块的文档注释；第6行使用\_\_author\_\_变量把作者写进去，这样当你公开源代码后别人就可以瞻仰你的大名。



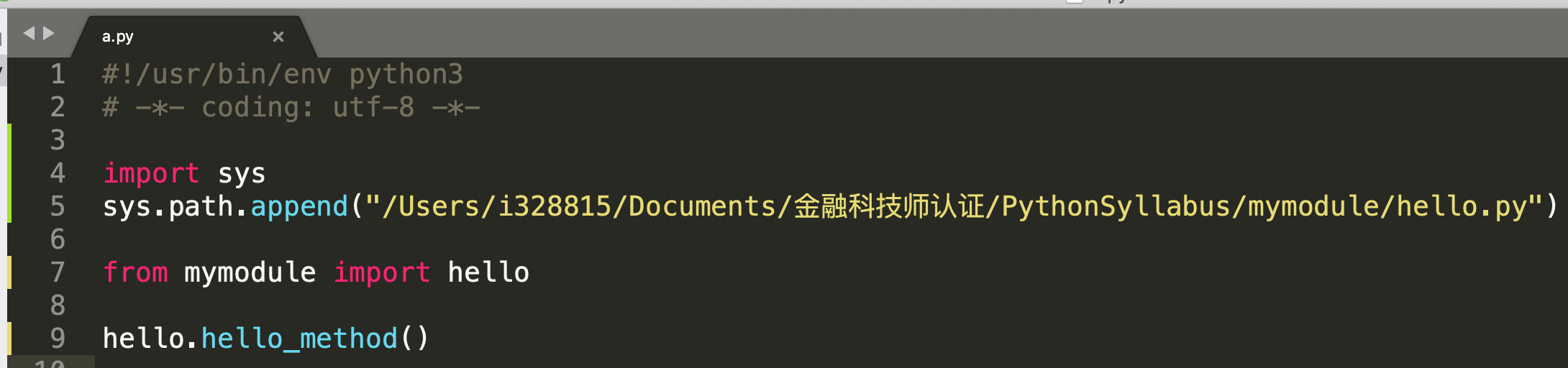
同时，我们在hello.py所在的文件夹里新建一个名字\_\_init\_\_.py的文件，文件内容可以为空。创建\_\_init\_\_.py文件以后，mymodule文件夹就不仅仅是一个文件夹，还可以认识是一个python的包，于是在这个包下面的所有模块我们都可以在程序中调用。里面的\_\_pycache\_\_文件夹可以先忽略，这个是在python程序运行时生成的临时文件。



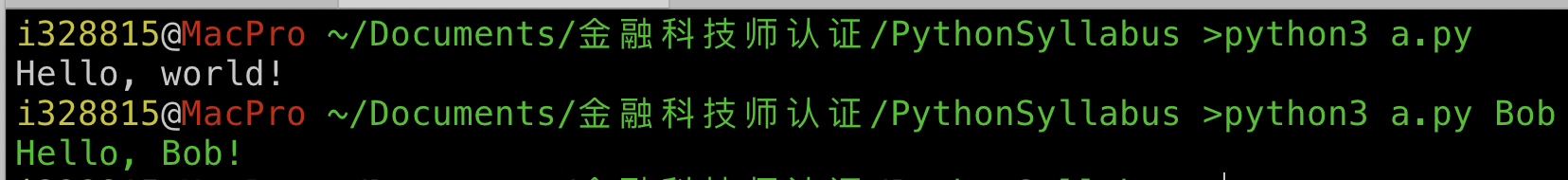
做好这些准备以后，我们来看示例如何在代码中调用mymodule包和里面的模块。我们新建一个文件a.py,该文件可以在任意文件夹下。



然后编辑a.py文件如下所示，代码的前2行仍然是默认的，指明这是一个python文件。代码的第4-5行，因为我们的包mymodule不在系统的path路径里，所以我们把之前定义的hello模块的路径添加到sys.path变量中，这样在a.py中想使用包mymodule及其包含的模块时，可以找到对应的python代码，在代码的第7行，我们通过from…import导入mymodule包中的hello模块。最后在代码的第9行，直接调用模块中的hello\_method()方法：



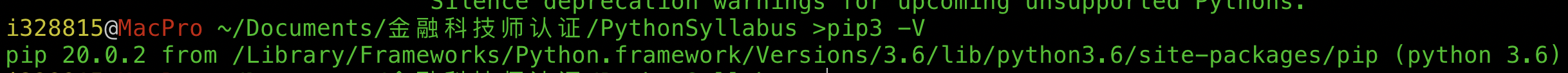
万事俱备，让我们在命令行调用a.py文件看看结果：



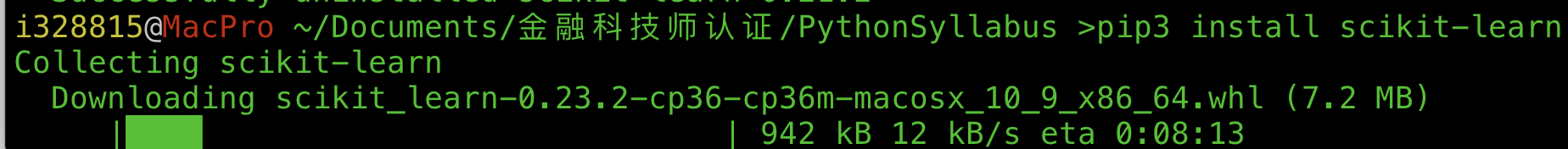
# 安装第三方模块

所谓的第三方模块，是指完成特定功能的模块，我们安装python时，只是安装了python解释器以及标准内置库的代码，对于解决特定任务的包和模块，比如MySQL驱动程序，Web框架Flask，科学计算Numpy，机器学习库sk-learn等,需要我们在安装好Python后单独安装必要的模块。

如果你正在使用Mac或Linux，安装pip本身这个步骤就可以跳过了。如果你正在使用Windows，请在安装Python时，确保安装时勾选了pip和Add python.exe to Path。在命令提示符窗口下尝试运行pip，如果Windows提示未找到命令，可以重新运行安装程序添加pip。注意：Mac或Linux上有可能并存Python 3.x和Python 2.x，因此对应的pip命令是pip3。安装好以后，在终端输入*pip3 -V*指令，可以查看pip的版本：



比如我们现在要进行机器学习的学习与应用，需要下载和使用sk-learn的包，可以使用pip3 install scikit-learn命令来安装对应的包，执行代码后首先下载对应的包文件并自动安装，让我们等待：



安装成功后,我们可以使用pip3 list命令查看已经安装的第三方包有哪些，如果看到scikit-learn，说明安装成功！

