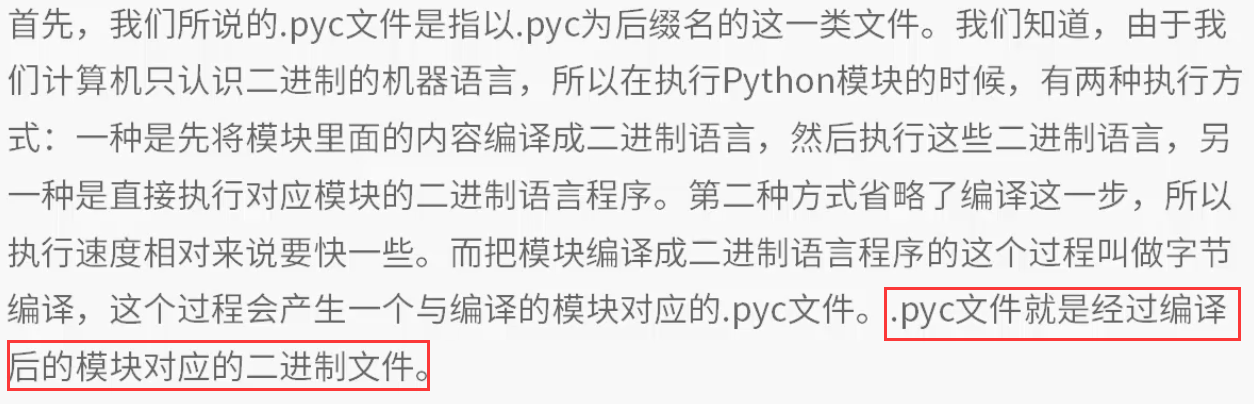
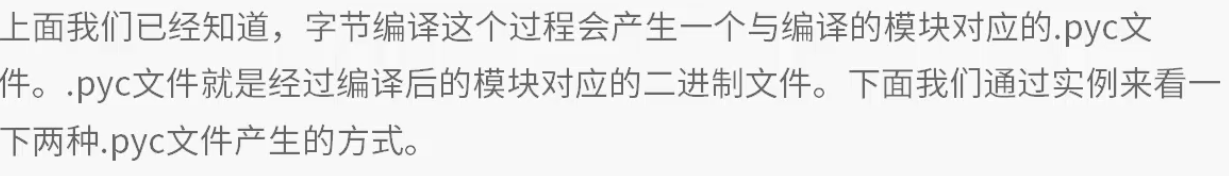
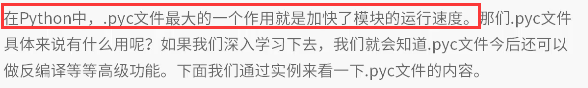
Python中的pyc文件

# pyc文件介绍



# Python是一种解释型语言





如果 **Python 进程**在机器上拥有**写入权限**，那么它将把**程序的字节码**保存为一个**以 .pyc 为扩展名的文件**（ ".pyc" 就是编译过的 ".py" 源代码）。当程序运行之后，你会在那些源代码的附近（也就是说同一个目录下）看到这些文件

Python这样保存**字节码**是作为一种**启动速度的优化**。下一次运行程序时，如果你在上次保存字节码之后没有修改过源代码的话，Python将会加载.pyc文件并跳过编译这个步骤。**当Python必须重编译时，它会自动检查源文件和字节码文件的时间戳：如果你又保存了源代码，下次程序运行时，字节码将自动重新创建**。

若想优化生成字节码，应注意这两点：

* **.pyc文件**是由.py文件经过编译后生成的字节码文件，其**加载速度相对于之前的.py文件有所提高**，而且还可以**实现源码隐藏**，以及**一定程度上的反编译**。比如，Python3.3编译生成的.pyc文件，Python3.4就别想着去运行啦！→\_→
* **.pyo文件**也是**优化**（注意这两个字，便于后续的理解）编译后的程序（**相比于.pyc文件更小**），也可以**提高加载速度**。但对于嵌入式系统，它可将所需模块编译成.pyo文件以**减少容量**。

# 手动编译生成pyc文件

## 利用py\_compile模块



import py\_compile  
py\_compile.compile('D:/Python/Python36/Lib/zipfile.py','D:/Python/Python36/Lib/zipfile.pyc')

## 在命令行下，执行命令

在所有的**Python选项**中：

* *-O*，表示**优化生成.pyo字节码**（这里又有“优化”两个字，得注意啦！）
* *-OO*，表示进一步**移除***-O*选项生成的字节码文件中的**文档字符串**（这是在作用效果上解释的，而不是说从*-O*选项得到的文件去除）
* *-m*，表示**导入并运行指定的模块**

对此，我们可以使用如下格式运行.py文件来生成.pyc文件。

**python -m py\_compile /path/to/需要生成.pyc的脚本.py** #若批量处理.py文件 #则替换为/path/to/{需要生成.pyc的脚本1,脚本2,...}.py #或者/path/to/

***其效果等效于如下代码*：**

**import py\_compile**

**py\_compile.compile(r'/path/to/需要生成.pyc的脚本.py')** #同样也可以是包含.py文件的目录路径

#此处尽可能使用raw字符串，从而避免转义的麻烦。比如，这里不加“r”的话，你就得对斜杠进行转义