Python基础教程

Beginning Python From Novice to Professional, 3rd Edition

Stephen CUI[1]

October 30, 2022

[1]cuixuanStephen@gmail.com

Contents

Table 1: 复看重点

知识点　章节（位置）	是否复看
生成器	×

Chapter 1

文件

1.1 打开文件

1.1.1 文件模式

显式地指定读取模式的效果与根本不指定模式相同。写入模式让你能够写入文件，并在文件不存在时创建它。独占写入模式更进一步，在文件已存在时引发 FileExistsError 异常。在写入模式下打开文件时，既有内容将被删除（截断），并从文件开头处开始写入；如果要在既有文件末尾继续写入，可使用附加模式。

‘+’ 可与其他任何模式结合起来使用，表示既可读取也可写入。例如，要打开一个文本文件进行读写，可使用 ‘r+’。（你可能还想结合使用 seek，详情请参阅本章后面的旁注“随机存取”。）请注意，‘r+’ 和 ‘w+’ 之间有个重要差别：后者截断文件，而前者不会这样做。默认模式为 ‘rt’，这意味着将把文件视为经过编码的 Unicode 文本，因此将自动执行解码和编码，且默认使用 UTF-8 编码。要指定其他编码和 Unicode 错误处理策略，可使用关键字参数 encoding 和 errors。

Table 1.1: 函数 open 的参数 mode 的最常见取值

值	描述
‘r’	读取模式（默认值）
‘w’	写入模式
‘x’	独占写入模式
‘a’	附加模式
‘b’	二进制模式（与其他模式结合使用）
‘t’	文本模式（默认值，与其他模式结合使用）
‘+’	读写模式（与其他模式结合使用）

1.2 文件的基本方法

三个标准流

一个标准数据输入源是 sys.stdin。当程序从标准输入读取时，你可通过输入来提供文本，也可使用管道将标准输入关联到其他程序的标准输出。

你提供给 print 的文本出现在 sys.stdout中，向 input 提供的提示信息也出现在这里。写入到 sys.stdout 的数据通常出现在屏幕上，但可使用管道将其重定向到另一个程序的标准输入。

错误消息（如栈跟踪）被写入到 sys.stderr，但与写入到 sys.stdout 的内容一样，可对其进行重定向。

1.2.1 使用管道重定向输出

随机存取

随机存取在本章中，我将文件都视为流，只能按顺序从头到尾读取。实际上，可在文件中移动，只访问感兴趣的部分（称为随机存取）。为此，可使用文件对象的两个方法：seek 和 tell。方法 `seek(offset[, whence])` 将当前位置（执行读取或写入的位置）移到 offset 和whence 指定的地方。参数 offset 指定了字节（字符）数，而参数 whence 默认为 `io.SEEK_SET(0)`，这意味着偏移量是相对于文件开头的（偏移量不能为负数）。参数 whence 还可设置为

io.SEEK_CUR(1) 或 io.SEEK_END(2)，其中前者表示相对于当前位置进行移动（偏移量可以为负），而后者表示相对于文件末尾进行移动。示例
方法 tell() 返回当前位于文件的什么位置。

1.2.2 读取和写入行

要读取一行（从当前位置到下一个分行符的文本），可使用方法 readline。调用这个方法时，可不提供任何参数（在这种情况下，将读取一行并返回它）；也可提供一个非负整数，指定 readline 最多可读取多少个字符。要读取文件中的所有行，并以列表的方式返回它们，可使用方法 readlines。

方法 writelines 与 readlines 相反：接受一个字符串列表（实际上，可以是任何序列或可迭代对象），并将这些字符串都写入到文件（或流）中。请注意，写入时不会添加换行符，因此你必须自行添加。另外，没有方法 writeline，因为可以使用 write。

1.2.3 关闭文件

别忘了调用方法close将文件关闭。通常，程序退出时将自动关闭文件对象（也可能在退出程序前这样做），因此是否将读取的文件关闭并不那么重要。然而，关闭文件没有坏处，在有些操作系统和设置中，还可避免无意义地锁定文件以防修改。另外，这样做还可避免用完系统可能指定的文件打开配额。

对于写入过的文件，一定要将其关闭，因为 Python 可能**缓冲**你写入的数据（将数据暂时存储在某个地方，以提高效率）。因此如果程序因某种原因崩溃，数据可能根本不会写入到文件中。安全的做法是，使用完文件后就将其关闭。如果要重置缓冲，让所做的修改反映到磁盘文件中，但又不想关闭文件，可使用方法 flush。然而，需要注意的是，根据使用的操作系统和设置，flush 可能出于锁定考虑而禁止其他正在运行的程序访问这个文件。只要能够方便地关闭文件，就应将其关闭。

要确保文件得以关闭，可使用一条 try/finally 语句，并在 finally 子句中调用 close。实际上，有一条专门为此设计的语句，那就是 with 语句。with 语句让你能够打开文件并将其赋给一个变量（这里是 somefile）。在语句体

中，你将数据写入文件（还可能做其他事情）。到达该语句末尾时，将自动关闭文件，即便出现异常亦如此。

上下文管理器

上下文管理器 with 语句实际上是一个非常通用的结构，允许你使用所谓的上下文管理器。上下文管理器是支持两个方法的对象：`__enter__`和`__exit__`。
方法`__enter__`不接受任何参数，在进入with语句时被调用，其返回值被赋给关键字as后面的变量。
方法`__exit__`接受三个参数：异常类型、异常对象和异常跟踪。它在离开方法时被调用（通过前述参数将引发的异常提供给它）。如果`__exit__`返回 False，将抑制所有的异常。文件也可用作上下文管理器。
它们的方法`__enter__`返回文件对象本身，而方法`__exit__`关闭文件。