Contents

Chapter 1

培养 Pythonic 思维

1.1 查询自己使用的 Python 版本

在执行 Python 代码的过程中，也可以通过内置的 sys 模块查询相关的值确定当前使用的 Python 版本。

1.2 遵循 PEP 8 风格指南

与表达式和语句有关的建议

- 采用行内否定，即把否定词直接写在要否定的内容前面，而不要放在整个表达式的前面。
- 不要通过长度判断容器或序列是不是空的，因为 Python 会把空置自动评估为 False。
- 如果要判断容器或序列里面有没有内容，也不应该通过长度来判断，而是应该采用 `if somelist` 语句，因为 Python 会把非空的值自动判定为 True。
-

与引入有关的建议

- 引入模块时，总是应该使用绝对名称，而不应该根据当前模块路径来使用相对名称。
- 如果一定要用相对名称来编写 import 语句，那就应该明确地写成：`from . import foo`。s

1.3 了解 bytes 和 str 的区别

Python 有两种类型可以表示字符序列：一种是 bytes，另一种是 str。bytes 实例包含的是原始数据，即 8 位的无符号值（通常按照 ASCII 编码标准来显式）。

str 实例包含对而是 Unicode 码点（code point，也叫做代码点），这些码点与人类语言之中的文本字符相对应。

There are two big gotchas when dealing with raw 8-bit values and Unicode strings in Python.

The first issue is that bytes and str seem to work the same way, but their instances are not compatible with each other, so you must be deliberate about the types of character sequences that you’re passing around.

判断 bytes 与 str 实例是否相等，总是会评估为假（False），即便这两个实例表示的字符完全相同，它们也不想等。

两种类型的实例都可以出现在 % 操作符的右侧，用来替换左侧那个格式字符串里面的 %s。如果格式字符串是 bytes 类型，那么不能用 str 实例来替换其中的 %s，因为 Python不知道这个 str 应该按照什么方案来编码，但是反过来却可以，问题是，这可能跟你想要的结果不一样。

第二个问题发生在操作文件句柄的时候，这里的句柄指由内置的 open 函数返回的句柄。这样的句柄默认需要使用 Unicode 字符串操作，而不能采用原始的 bytes。

1.4 用支持插值的 f-string 取代 C 风格的格式字符串与 str.format 方法