Chapter 1

1.1 Using * to Grab Excess Items

Chapter 2

Unicode 文本和字节序列

2.1 字节概要

bytes 或 bytearray 对象的各个元素是介于 0~255（含）之间的整数。然而，二进制序列的切片始终是同一类型的二进制序列，包括长度为 1 的切片。

2.2 处理解码和编码问题

虽然有个一般性的 `UnicodeError` 异常，但是报告错误时几乎都会指明具体的异常：`UnicodeEncodeError`（把字符串转换成二进制序列时）或 `UnicodeDecodeError`（把二进制序列转换成字符串时）。如果源码的编码与预期不符，加载 Python 模块时还可能抛出 `SyntaxError`。

2.2.1 处理 UnicodeEncodeError

多数非 UTF 编解码器只能处理 Unicode 字符的一小部分子集。把文本转换成字节序列时，如果目标编码中没有定义某个字符，那就会抛出 UnicodeEncodeError 异常，除非把 errors 参数传给编码方法或函数，对错误进行特殊处理。

ASCII is a common subset to all the encodings that I know about, therefore encoding should always work if the text is made exclusively of ASCII characters. Python 3.7 added a new boolean method `str.isascii()` to check whether your Unicode text is 100% pure ASCII. If it is, you should be able to encode it to bytes in any encoding without raising UnicodeEncodeError.

2.2.2 处理UnicodeDecodeError

不是每一个字节都包含有效的 ASCII 字符，也不是每一个字符序列都是有效的 UTF-8 或 UTF-16。因此，把二进制序列转换成文本时，如果假设是这两个编码中的一个，遇到无法转换的字节序列时会抛出 UnicodeDecodeError。

2.3 Unicode 文本排序

Python 比较任何类型的序列时，会一一比较序列里的各个元素。对字符串来说，比较的是码位。可是在比较非 ASCII 字符时，得到的结果不尽如人意。

在 Python 中，非 ASCII 文本的标准排序方式是使用 locale.strxfrm 函数，根据 locale 模块的文档，这个函数会“把字符串转换成适合所在区域进行比较的形式”。使用 locale.strxfrm 函数之前，必须先为应用设定合适的区域设置，还要祈祷操作系统支持这项设置。

2.3.1

2.4 Unicode 数据库

2.4.1 字符的数值意义