Chapter 1

1.1 用正则表达式查找文本模式

向 re.compile()传入一个字符串值，表示正则表达式，它将返回一个 Regex 模式对象（或者就简称为 Regex 对象）。

Regex 对象的 search()方法查找传入的字符串，寻找该正则表达式的所有匹配。如果字符串中没有找到该正则表达式模式，search()方法将返回 None。如果找到了该模式，search()方法将返回一个 Match 对象。Match 对象有一个 group()方法，它返回被查找字符串中实际匹配的文本。

虽然在 Python 中使用正则表达式有几个步骤，但每一步都相当简单。

1. 用 import re 导入正则表达式模块。
2. 用 re.compile()函数创建一个 Regex 对象（记得使用原始字符串）。
3. 向 Regex 对象的 search()方法传入想查找的字符串。它返回一个 Match 对象。
4. 调用 Match 对象的 group()方法，返回实际匹配文本的字符串。

1.2 用正则表达式匹配更多模式

1.2.1 利用括号分组

> 我觉得可以不翻译为分组，而是理解为组件。

添加括号将在正则表达式中创建“分组”：然后可以使用 group()匹配对象方法，从一个分组中获取匹配的文本。正则表达式字符串中的第一对括号是第 1 组。第二对括号是第 2 组。向 group()匹配对象方法传入整数 1 或 2，就可以取得匹配文本的不同部分。向 group()方法传入 0 或不传入参数，将返回整个匹配的文本。**只找第一个不会多找**

如果想要一次就获取所有的分组，请使用 groups()方法。

括号在正则表达式中有特殊的含义，但是如果你需要在文本中匹配括号，怎么办？在这种情况下，就需要用斜杠对 (与) 进行字符转义。

在正则表达式中，以下字符具有特殊含义：

```
. ^ $ * + ? { } ( ) [ ] \ |
```

如果要检测包含这些字符的文本模式，那么就需要用斜杠对它们进行转义。

要确保正则表达式中，没有将转义的括号 (与) 误作为 (与)。如果你收到类似 missing 或 unbalanced parenthesis 的错误信息，则可能是忘记了为分组添加转义的右括号。

1.2.2 用管道匹配多个分组

字符 | 称为“管道”。希望匹配许多表达式中的一个时，就可以使用它。例如，正则表达式 r‘Batman|Tina Fey’ 将匹配 ‘Batman’ 或 ‘Tina Fey’。

如果 Batman 和 Tina Fey 都出现在被查找的字符串中，第一次出现的匹配文本，将作为 Match 对象返回。

> 利用 findall()方法，可以找到“所有”匹配的地方。

1.2.3 用问号实现可选匹配

有时候，想匹配的模式是可选的。就是说，不论这段文本在不在，正则表达式都会认为匹配。字符 ? 表明它前面的分组在这个模式中是可选的。你可以认为 ? 是在说，“匹配这个问号之前的分组零次或一次”。

1.2.4 用星号匹配零次或多次

*（称为星号）意味着“匹配零次或多次”，即星号之前的分组，可以在文本中出现任意次。它可以完全不存在，或一次又一次地重复。

1.2.5 用加号匹配一次或多次

* 意味着“匹配零次或多次”，+（加号）则意味着“匹配一次或多次”。星号不要求分组出现在匹配的字符串中，但加号不同，加号前面的分组必须“至少出现一次”。这不是可选的。

1.2.6 用花括号匹配特定次数

如果想要一个分组重复特定次数，就在正则表达式中该分组的后面，跟上花括号包围的数字。除了一个数字，还可以指定一个范围，即在花括号中写下一个最小值、一个逗号和一个最大值。也可以不写花括号中的第一个或第二个数字，不限定最小值或最大值。

1.3 贪心和非贪心匹配

Python 的正则表达式默认是“贪心”的，这表示在有二义的情况下，它们会尽可能匹配最长的字符串。花括号的“非贪心”版本匹配尽可能最短的字符串，即在结束的花括号后跟着一个问号。

> 请注意，问号在正则表达式中可能有两种含义：声明非贪心匹配或表示可选的分组。这两种含义是完全无关的。