2.7 最短匹配模式¶

问题¶

你正在试着用正则表达式匹配某个文本模式，但是它找到的是模式的最长可能匹配。 而你想修改它变成查找最短的可能匹配。

解决方案¶

这个问题一般出现在需要匹配一对分隔符之间的文本的时候(比如引号包含的字符串)。 为了说明清楚，考虑如下的例子：

```
>>> str_pat = re.compile(r'"(.*)"')
>>> text1 = 'Computer says "no."'
>>> str_pat.findall(text1)
['no.']
>>> text2 = 'Computer says "no." Phone says "yes."'
>>> str_pat.findall(text2)
['no." Phone says "yes.']
>>>
```

在这个例子中，模式 `r'\"(.*)\"'` 的意图是匹配被双引号包含的文本。 但是在正则表达式中*操作符是贪婪的，因此匹配操作会查找最长的可能匹配。 于是在第二个例子中搜索 `text2` 的时候返回结果并不是我们想要的。

为了修正这个问题，可以在模式中的*操作符后面加上?修饰符，就像这样：

```
>>> str_pat = re.compile(r'"(.*?)"')
>>> str_pat.findall(text2)
['no.', 'yes.']
>>>
```

这样就使得匹配变成非贪婪模式，从而得到最短的匹配，也就是我们想要的结果。

讨论¶

这一节展示了在写包含点(.)字符的正则表达式的时候遇到的一些常见问题。 在一个模式字符串中，点(.)匹配除了换行外的任何字符。 然而，如果你将点(.)号放在开始与结束符(比如引号)之间的时候，那么匹配操作会查找符合模式的最长可能匹配。 这样通常会导致很多中间的被开始与结束符包含的文本被忽略掉，并最终被包含在匹配结果字符串中返回。 通过在 `*` 或者 `+` 这样的操作符后面添加一个 `?` 可以强制匹配算法改成寻找最短的可能匹配。