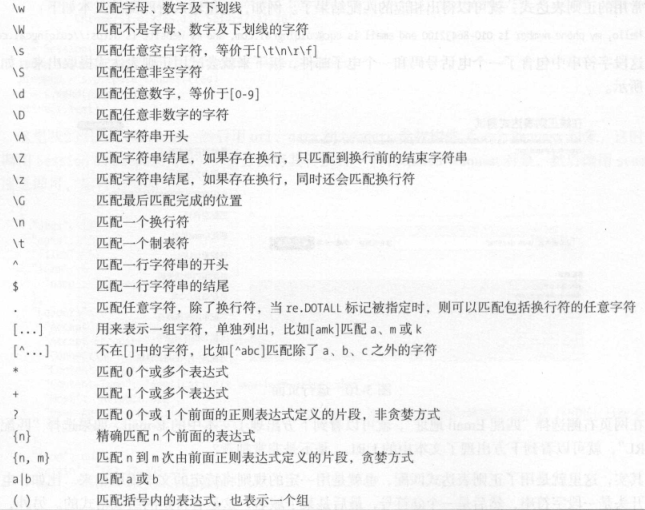
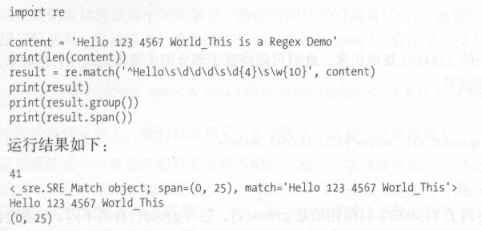
## 常用的匹配规则



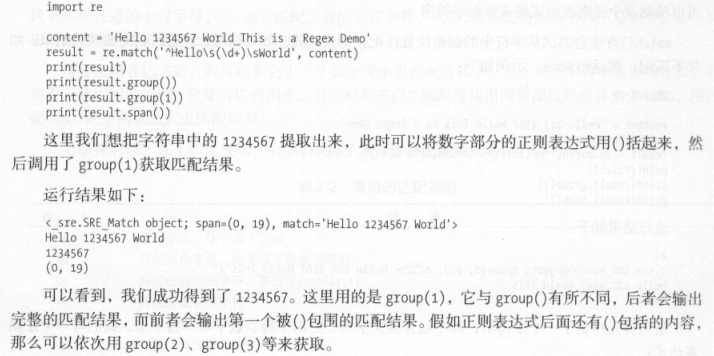
## match()方法：

在match()方法中，第一个参数传入正则表达式，第二个参数传入要匹配的字符，第三个参数是修饰符。此方法会尝试从字符串的起始位置匹配正则表达式，如果匹配，则返回匹配的字符串，没有则返回None。



## 分组--()

如果想要从一段文本中提取出对应的内容，就可以使用()将想提取的内容括起来，()实际上标记了一个子表达式的开始和结束的位置。被标记的每个子表达式会依次对应每一个分组，调用group()方法传入分组的索引即可获取提取的结果

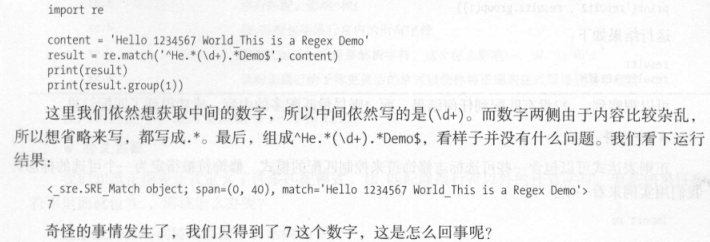


## 通用匹配---.\*

其中.可以匹配任意字符（除换行符），\*表示匹配前面的字符无限次，所以它们组合在一起就可以匹配任意字符了。

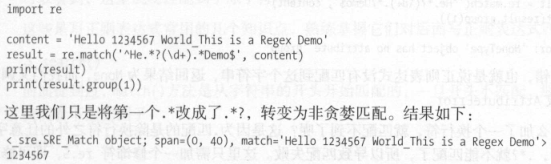
## 贪婪与非贪婪

看一个例子：



这里就涉及一个贪婪匹配与非贪婪匹配的问题了。在贪婪模式下，.\*会匹配尽可能多的字符。正则表达式中.\*后面是\d+，也就是至少一个数字，并没有指定具体多少个数字，因此，.\*尽可能匹配多的字符，这里就把123456匹配了（也就是说第一个.\*会一直匹配到123456），给\d+留下一个可满足条件的数字7，最后得到的结果是7了。

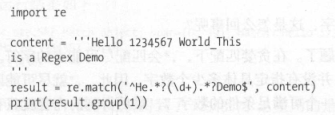
这明显会给我们带来很大的不便。有时候，匹配结果会莫名其妙少了一部分内容，其实这里只需要使用非贪婪匹配就好了。非贪婪匹配的写法就是.\*?，多了一个?，它达到的效果如下：

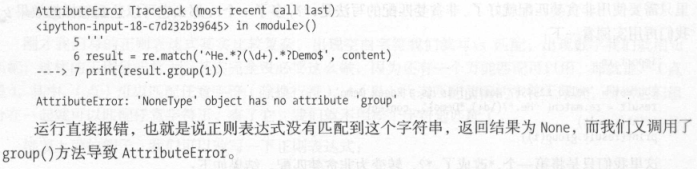


非贪婪匹配就是尽可能匹配少的字符。当.\*?匹配到Hello后面的空白字符时，再往后的字符就是数字了，而\d+恰好可以匹配，那么这里.\*?就不再进行匹配，交给\d+去匹配后面的数字，所以.\*?匹配了尽可能少的字符就一个都没有匹配，\d+的结果就是1234567了。

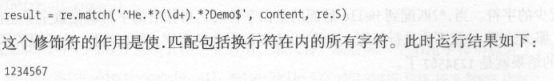
## 修饰符

假如有一个例子，如下：

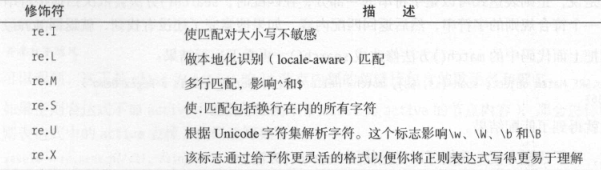
结果：



为什么上面得例子加了一个换行符就匹配不到了？是因为.匹配的是除换行符之外的任意字符，当遇到换行符是，.\*?就不能匹配了，所以导致匹配失败。这里需要加一个修饰符re.S，即可修正找个错误：



修饰符表格：



对修饰符的具体介绍，如什么是本地化识别匹配：

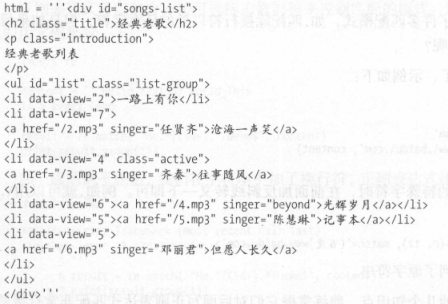
<https://blog.csdn.net/zouyee/article/details/17036135>

## search()方法

match()方法是从字符串的开头开始匹配的，一旦开头不匹配，那么整个匹配就失败了。match()方法更适合用来检测某个字符串是否符合某个正则表达式的规则。

search()方法在匹配时会扫描整个字符串，然后返回第一个成功匹配的结果。也就是说，正则表达式可以是字符串的一部分，在匹配时，search()方法会依次扫描字符串，直到找到第一个符合规则的字符串，然后返回匹配内容，如果搜索完了还没有找到，就返回None。

实现信息提取的例子---提取下面第三个li节点下a标签中的歌手名和歌曲名：

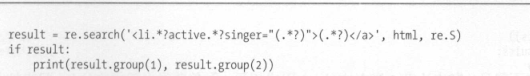


**分析：**

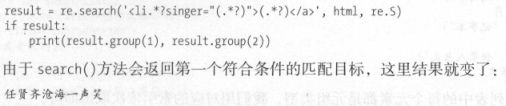
首先我们尝试提取class为active的li节点内部的超链接包含的歌手名和歌曲名，此时需要提取第三个li节点下a节点的singer属性和文本。此时正则表达式可以以li开头，然后寻找一个标志符active，中间的部分可以用.\*?来匹配。接下来，要提取singer找个属性值，所以还需要写入singer=”(.\*?)”，这里需要提取的部分用小括号括起来，以便使用group()方法提取出来，它的两侧边界是双引号。然后还学要匹配a节点的文本，其中它的做边界是啊>，右边界是</a>。然后目标内容依然用(.\*?)来匹配，所以最后的正则表达式就成了：



最后的代码：

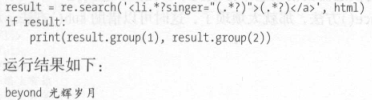


将正则表达式的active去掉：



把active标签去掉后，从字符串开头开始搜索，此时符合条件的节点就变成了第二个li节点，后面的就不再匹配，所以运行结果就变成了第二个li节点中的内容。

将search()方法中的第三个参数re.S去掉：



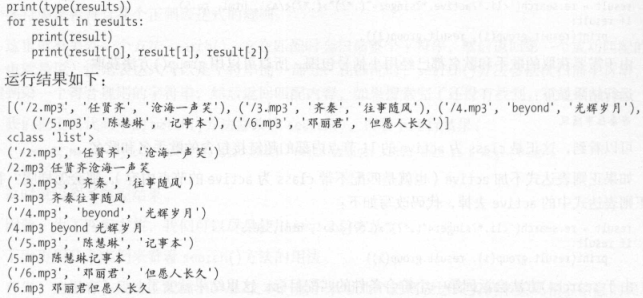
去掉re.S后，.\*?不再匹配换行符，所以正则表达式就不会匹配到第二个和第三个li节点了，而第四个li节点中，不包含换行符，所以成功匹配。

## findAll()方法

如果想要获取匹配正则表达式的所有内容，就可以使用fiandAll()方法。此方法会搜索整个字符串，然后返回匹配正则表达式的所有内容。

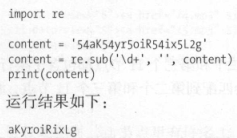
还是search()方法这一章节中的html文本，如果想要获取所有a节点的超链接、歌手和歌曲名，就可以使用findAll()。如果有返回内容，就是列表类型，所以需要遍历以下来依次获每组内容，代码如下：



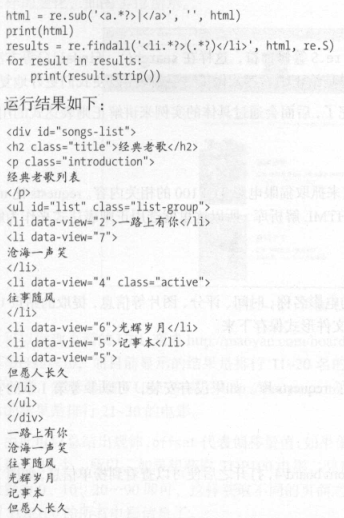


## sub()方法

此方法可用于使用正则表达式来修改文本。如下，去掉字符串中的所有数字：



比如finadAll()方法章节中的提取歌曲名字，可以改写为如下方式：



## compile()方法

此方法可以将正则字符串编译成正则对象然后反复使用。

