字符串和正则表达式

在编写处理字符串的程序或网页时，经常会有查找符合某些复杂规则的字符串的需要，正则表达式就是用于描述这些规则的工具，换句话说正则表达式是一种工具，它定义了字符串的匹配模式（如何检查一个字符串是否有跟某种模式匹配的部分或者从一个字符串中将与模式匹配的部分提取出来或者替换掉）。如果你在Windows操作系统中使用过文件查找并且在指定文件名时使用过通配符（*和?），那么正则表达式也是与之类似的用来进行文本匹配的工具，只不过比起通配符正则表达式更强大，它能更精确地描述你的需求（当然你付出的代价是书写一个正则表达式比打出一个通配符要复杂得多，要知道任何给你带来好处的东西都是有代价的，就如同学习一门编程语言一样），比如你可以编写一个正则表达式，用来查找所有以0开头，后面跟着2-3个数字，然后是一个连字号“-”，最后是7或8位数字的字符串（像028-12345678或0813-7654321），这不就是国内的座机号码吗。最初计算机是为了做数学运算而诞生的，处理的信息基本上都是数值，而今天我们在日常工作中处理的信息基本上都是文本数据，我们希望计算机能够识别和处理符合某些模式的文本，正则表达式就显得非常重要了。今天几乎所有的编程语言都提供了对正则表达式操作的支持，Python通过标准库中的re模块来支持正则表达式操作。

我们可以考虑下面一个问题：我们从某个地方（可能是一个文本文件，也可能是网络上的一则新闻）获得了一个字符串，希望在字符串中找出手机号和座机号。当然我们可以设定手机号是11位的数字（注意并不是随机的11位数字，因为你没有见过“25012345678”这样的手机号吧）而座机号跟上一段中描述的模式相同，如果不使用正则表达式要完成这个任务就会很麻烦。

下面我们先对正则表达式中的一些基本符号进行扼要的介绍。

符号	解释	示例	说明
.	匹配任意字符	b.t	可以匹配bat / but / b#t / b1t等
\w	匹配字母/数字/下划线	b\wt	可以匹配bat / b1t / b_t等 但不能匹配b#t
\s	匹配空白字符（包括\r、\n、\t等）	love\syou	可以匹配love you
\d	匹配数字	\d\d	可以匹配01 / 23 / 99等
\b	匹配单词的	\bThe\b	

	边界		
^	匹配字符串的开始	^The	可以匹配The开头的字符串
$	匹配字符串的结束	.exe$	可以匹配.exe结尾的字符串
\W	匹配非字母/数字/下划线	b\Wt	可以匹配b#t / b@t等但不能匹配but / b1t / b_t等
\S	匹配非空白字符	love\Syou	可以匹配love#you等但不能匹配love you
\D	匹配非数字	\d\D	可以匹配9a / 3# / 0F等
\B	匹配非单词边界	\Bio\B	
[]	匹配来自字符集的任意单一字符	[aeiou]	可以匹配任一元音字母字符
[^]	匹配不在字符集中的任意单一字符	[^aeiou]	可以匹配任一非元音字母字符
*	匹配0次或多次	\w*	
+	匹配1次或多次	\w+	
?	匹配0次或1次	\w?	
{N}	匹配N次	\w{3}	
{M,}	匹配至少M次	\w{3,}	
{M,N}	匹配至少M次至多N次	\w{3,6}	
|	分支	foo|bar	可以匹配foo或者bar

(?#)	注释		
(exp)	匹配exp并捕获到自动命名的组中		
(?<name>exp)	匹配exp并捕获到名为name的组中		
(?:exp)	匹配exp但是不捕获匹配的文本		
(?=exp)	匹配exp前面的位置	\b\w+(?=ing)	可以匹配I'm dancing中的danc
(?<=exp)	匹配exp后面的位置	(?<=\bdanc)\w+\b	可以匹配I love dancing and reading中的第一个ing
(?!exp)	匹配后面不是exp的位置		
(?<!exp)	匹配前面不是exp的位置		
*?	重复任意次，但尽可能少重复	a.*b a.*?b	将正则表达式应用于aabab，前者会匹配整个字符串aabab，后者会匹配aab和ab两个字符串
+?	重复1次或多次，但尽可能少重复		
??	重复0次或1次，但尽可能少重复		
{M,N}?	重复M到N次，但尽可能少重复		
{M,}?	重复M次以上，但尽可能少重复		

> ***说明：*** *如果需要匹配的字符是正则表达式中的特殊字符，那么可以使用\进行转义处理，例如想匹配小数点可以写成\.就可以了，因为直接写.会匹配任意字符。其他关于正则表达式的东西我们推荐阅读《正则表达式30分钟入门教程》以及该文章提供的相关链接。*

下面是Python中re模块核心函数和方法。

函数	说明
compile(pattern, flags=0)	编译正则表达式返回正则表达式对象
match(pattern, string, flags=0)	用正则表达式匹配字符串 成功返回匹配对象 否则返回None
search(pattern, string, flags=0)	搜索字符串中第一次出现正则表达式的模式 成功返回匹配对象 否则返回None
split(pattern, string, maxsplit=0, flags=0)	用正则表达式指定的模式分隔符拆分字符串 返回列表
sub(pattern, repl, string, count=0, flags=0)	用指定的字符串替换原字符串中与正则表达式匹配的模式 可以用count指定替换的次数
fullmatch(pattern, string, flags=0)	match函数的完全匹配（从字符串开头到结尾）版本
findall(pattern, string, flags=0)	查找字符串所有与正则表达式匹配的模式 返回字符串的列表
finditer(pattern, string, flags=0)	查找字符串所有与正则表达式匹配的模式 返回一个迭代器
purge()	清除隐式编译的正则表达式的缓存
re.I / re.IGNORECASE	忽略大小写匹配标记
re.M / re.MULTILINE	多行匹配标记

> ***说明：****上面的函数re模块提供的函数，实际开发中也可以用正则表达式对象的方法替代对这些函数的使用，如果一个正则表达式需要重复的使用，那么先通过compile函数编译正则表达式并创建出正则表达式对象无疑是更为明智的选择。*