正则表达式

正则表达式是一些由字符和特殊字符组成的字符串，她们描述了这些字符和字符的某种重复模式，因此能按某种模式匹配一个有相似特征的字符串的集合。

查找与匹配的比较

搜索，在字符串的任意部分中查找匹配的模式，而匹配时指，判断一个字符串能否从起始处全部或者部分的匹配某个模式。搜索通过search()函数或方法来实现，而匹配则是调用match()来实现。

正则表达式的强大之处在于特殊符号的应用，特殊符号定义了字符集合，子组匹配，模式重复次数。

记号	说明	举例
iteral	匹配字符串的值	foo
re1/re2	匹配正则表达式re1或re2	
.	匹配任何字符（换行符除外）	b.b
^	匹配字符串的开始	Dear
$	匹配字符串的结尾	/bin/*sh$
*	匹配前面出现的正则表达式零次或多次	[A-Za-z0-9]*
+	匹配前面出现的正则表达式一次货多次	[a-z]+.com
?	匹配前面出现的正则表达式零次或一次	goo?
{N}	匹配前面出现的正则表达式	{0-9}{3}
{M，N}	匹配重复出现M次到N次的正则表达式	[0-9]{5,9}
[...]	匹配字符组里出现的任意一个字符	[aeiou]
[..x-y..]	匹配从字符x到y的任意一个字符	[0-9],[A-Za-z]
[^...]	不匹配此字符集中出现的任何一个字符，包括某一范围的字符（如果在此字符集中出现）	[^aeiou] [^A-Za-z0-9]
(*	+	?
{...}	匹配封闭括号中正则表达式（RE）,并保存为子组	{[0-9]{3}}?, (oo

\d	**匹配任何数字，和[0-9]一样（\D是\d的反义：任何非数字符）**	**data\d+.txt**
\w	匹配任何数字字母字符，和[A-Za-z0-9]相同（\W是\w的反义）	[A-Za-z_]\w+
\s	匹配任何空白符（\S是\s的反义）	of\sthe
\b	匹配单词边界（\B是\b的反义）	\bThe\b
\nn	匹配已保存的子组	price:\16
\c	逐一匹配特殊字符c(取消他的特殊含义，按字面匹配)	
\A(\Z)	匹配字符串的起始（结束）	

一对圆括号（()）和正则表达式一起使用时可以实现以下任意一个或两个功能：

- 对正则表达式进行分组
- 匹配子组

为什么需要使用子组匹配呢？

主要是有时除了进行匹配操作外，你还想提取匹配模式的内容。

re模块：核心函数和方法

函数/方法	描述
compile(pattern,flags=0)	对正则表达式pattern进行编译，flags是可选标识符，并返回一个regex对象。
match(pattern,string,flags=0)	尝试用正则表达式模式pattern匹配字符串string，flags是可选标志符，如果匹配成功，则返回一个匹配对象，否则返回none。
search(pattern,string,flags=0)	在字符串string中查找正则表达式模式pattern的第一次出现，flags是可选标志符，如果匹配成功，则返回一个匹配对象，否则返回none。
findall(pattern,string[,flags])	在字符串string中查找正则表达式模式pattern的所有（非重复）出现，返回一个匹配对象的列表。
finditer(pattern,stirng[,flags])	和findall()相同，但返回的不是列表而是迭代器，对于每个匹配，该迭代器返回一个匹配对象
split(pattern,string,max=0)	根据正则表达式pattern中的分隔符把字符string分割为一个列表，返回成功匹配的列表，最多分割max次，（默认是分割所有匹配的地方）
sub(pattern,repl,string,max=0)	把字符串string中所有正则表达式pattern的地方替换成字符串repl，如果max的值没有给出，则对所有匹配的地方进行替换

group(num=0)	返回全部匹配对象（或指定编号是num的子组）
groups()	返回一个包含全部匹配的子组的元组（如果没有成功匹配，就返回一个空元素）