在匹配字符串时，有时仅需要在开头或者结尾处匹配，这时可以使用脱字符^标记开始，使用美元符号$标记结尾。正则表达式的特殊字符以及特殊字符的组合使用不止上述这些，列表如下，在使用时可供查询：

字符	描述
text	匹配文字字符串text
.	匹配任何字符串，但换行符除外
^	匹配字符串的开始标志
$	匹配字符串的结束标志
*	匹配前面表达式的0个或多个副本，匹配尽可能多的副本
+	匹配前面表达式的1个或多个副本，匹配尽可能多的副本
?	匹配前面表达式的0个或多个副本
*?	匹配前面表达式的0个或多个副本，匹配尽可能少的副本
+?	匹配前面表达式的1个或多个副本，匹配尽可能少的副本
??	匹配前面表达式的0个或1个副本，匹配尽可能少的副本
{m}	准确匹配前面表达式的m个副本
{m,n}	匹配前面表达式的第m到n个副本 ，匹配尽可能多的副本。如果省略了m，它将默认设置为0。如果省略了n，它将默认设置为无穷大
{m,n}?	匹配前面表达式的第m到n个副本 ，匹配尽可能少的副本。
[…]	匹配一组字符，如r'[abcdef]'或r'[a-zA-Z]'。特殊字符（如*)在字符集中是无效的
[^…]	匹配集合中未包含的字符，如r'[^0-9]'
A|B	匹配A或B，其中A和B都是正则表达式
(…)	匹配圆括号中的正则表达式（圆括号中的内容为一个分组）并保存匹配的子字符串。 在匹配时，分组中的内容可以使用所获得的MatchObject对象的group（）方法获取
(?aiLmsux)	将字符"a","I","L","m","s","u"和"x"解释为与提供给re.compile()的re.A,re.i,re.L,re.M,re.s,re.u.re.x相对应的标志设置。"a"仅在Python3中可用
(?:…)	匹配圆括号中的正则表达式，但丢弃匹配的子字符串
(?P<name>…)	匹配圆括号中的正则表达式并创建一个指定分组。分组名称必须是有效的Python标识符
(?P=name)	匹配一个早期指定的分组所匹配的文本
(?#...)	一个注释。圆括号中的内容将被忽略
(?=…)	只有在括号中的模式匹配时，才匹配前面的表达式。例如，'hello(?!=world)'只有'world"匹配时才匹配'hello'
(?!...)	只有在括号中的模式不匹配时，才匹配前面的表达式。例如，'hello(?!=world)'只有'world"不匹配时才匹配'hello'
(?<=…)	如果括号后面的表达式前面的值与括号中的模式匹配，则匹配该表达式，例如，只有当'def'前面是'abc'时，r'(?<abc)def'才会与之匹配
(?<!...)	如果括号后面的表达式前面的值与括号中的模式不匹配，则匹配该表达式，例如，只有当'def'前面是'abc'时，r'(?<abc)def'才会与之匹配
(?(id|name)ypat|npat	检查id或name标识的正则表达式组是否存在。如果存在，则匹配正则表达式ypat。否则，匹配可选的表达式npat.例如r'(Hello)?(/(1)World|Howday)'匹配字符串'Hello World'或'Howdy'

一些用 \ 开始的特殊字符所表示的预定义字符集通常是很有用的，像数字集、字母集或其他非空字符集。

字符	描述
\number	匹配与前面的组编号匹配的文本。组编号范围为1到99从左侧开始
\A	仅匹配字符串的开始标志
\b	匹配单词开始或结尾处的空字符串。单词(word)是一个字母数字混合的字符序列，以空格或任何其他非字母数字字符结束
\B	匹配不在单词开始或结尾处的空字符串
\d	匹配任何十进制数。等同于r'[0-9]'
\D	匹配任何非数字字符，等同于r'[^0-9]'
\s	匹配任何空格字符。等同于r'[\t\n\r\f\v]'
\S	匹配任何非空格字符。等同于r'[^\t\n\r\f\v]'
\w	匹配任何字母数字字符
\W	匹配\w定义的集合中不包含的字符
\z	仅匹配字符串的结束标志
\\	匹配反斜杠本身

- re 模块的方法

正则表达式的模式需要配合正则表达式的方法使用，前文在阐述特殊字符串时已经使用了方法 re.findall(pattern,string)，以列表形式返回给定模式的所有匹配项；re.search(pattern,string) 会在给定字符串中寻找第一个匹配给定正则表达式的子字符串，并返回 MatchObject 布尔值，存在为 True，否则为 False；re.match(pattern,string) 会在给定字符串的开头匹配正则表达式，返回 MatchObject 布尔值；re.split(pattern,string[,maxsplit=0]) 会根据模式的匹配项来分隔字符串，这样可以使用任意长度的分隔符分隔字符串，其中 maxsplit 参数为字符串最多可以分隔成的部分数；re.sub(pattern,repl,string) 使用给定的替换内容将匹配模式的子字符串替换掉；re.escape(string) 可以对字符串中所有可能被解释为正则运算符的字符进行转义，避免输入较多的反斜杠；re.compile(pattern) 可以将以字符串书写的正则表达式转换为模式对象，例如转换为模式对象后可以直接使用 pattern.search(string) 的方法，这与 re.search(pattern,string) 方式一样。因为使用 re 模块的方法时，不管是 re.search() 还是 re.match() 都会在内部将字符串表示的正则表达式转换为正则表达式模式对象，因此 re.compile() 的方法可以避免每次使用模式时都得重新转化的过程。

```
>>> import re # 调入正则表达式模块
>>> pat='[a-z]' # 建立模式，匹配 a-z 的所有小写字母
>>> text='python PYTHON' # 建立字符串
>>> re.findall(pat,text) # 以列表形式返回给定模式的所有匹配项
['p', 'y', 't', 'h', 'o', 'n']
>>> pat='[a-z]+' # 建立模式，匹配 a-z 的所有小写字母，同时增加了 + 加号特殊字符，尽可能多地匹配项
>>> re.findall(pat,text) # 增加 + 加号特殊字符后，小写字母不再单独返回列表，而是作为紧凑的整体
```