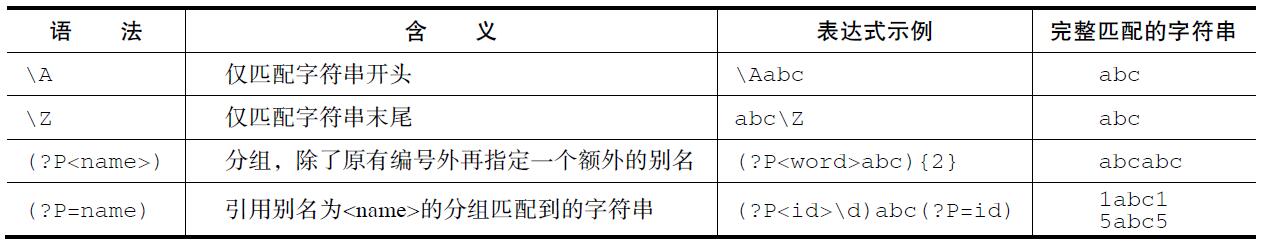
4.2.2　Python 与正则

上一节讲解了正则表达式的语法和应用，对于不同的编程语言来说，对正则表达式的语法绝大部分语言都是支持的，但是还是略有不同，每种编程语言都有一些独特的匹配规则，Python 也不例外。下面通过表4-7列出一些 Python 的匹配规则。

表4-7　Python 的匹配规则

[](http://popimage/?src=)

在讲 Python 对正则表达式的实现之前，首先让说一下反斜杠问题。正则表达式里使用“\”作为转义字符，这就可能造成反斜杠困扰。假如你需要匹配文本中的字符“\”，那么使用编程语言表示的正则表达式里将需要4个反斜杠“\\”：前两个和后两个分别用于在编程语言里转义成反斜杠，转换成两个反斜杠后再在正则表达式里转义成一个反斜杠。但是 Python 提供了对原生字符串的支持，从而解决了这个问题。匹配一个‘\’的正则表达式可以写为r‘\’，同样，匹配一个数字的‘\d’可以写成 r‘\d’，

Python 通过 re 模块提供对正则表达式的支持。使用 re 的一般步骤是先将正则表达式的字符串形式编译为 Pattern 实例，然后使用 Pattern 实例处理文本并获得匹配结果，最后使用 Match 实例获得信息，进行其他操作。主要用到的方法列举如下：

* re.compile（string[，flag]）
* re.match（pattern，string[，flags]）
* re.search（pattern，string[，flags]）
* re.split（pattern，string[，maxsplit]）
* re.findall（pattern，string[，flags]）
* re.finditer（pattern，string[，flags]）
* re.sub（pattern，repl，string[，count]）
* re.subn（pattern，repl，string[，count]）

首先说一下 re 中 compile 函数，它将一个正则表达式的字符串转化为 Pattern 匹配对象。示例如下：

pattern = re.compile(r'\d+')

这会生成一个匹配数字的 pattern 对象，用来给接下来的函数作为参数，进行进一步的搜索操作。

大家发现其他几个函数中，还有一个 flag 参数。参数 flag 是匹配模式，取值可以使用按位或运算符“|”表示同时生效，比如 re.I|re.M。flag 的可选值如下：

* re.I：忽略大小写。
* re.M：多行模式，改变“^”和“$”的行为。
* re.S：点任意匹配模式，改变“.”的行为。
* re.L：使预定字符类 \w\W\b\B\s\S 取决于当前区域设定。
* re.U：使预定字符类 \w\W\b\B\s\S\d\D 取决于 unicode 定义的字符属性。
* re.X：详细模式。这个模式下正则表达式可以是多行，忽略空白字符，并可以加入注释。

1.re.match（pattern，string[，flags]）

这个函数是从输入参数 string（匹配的字符串）的开头开始，尝试匹配 pattern，一直向后匹配，如果遇到无法匹配的字符或者已经到达 string 的末尾，立即返回 None，反之获取匹配的结果。示例如下：

*# coding:utf-8*

**import** re

*# 将正则表达式编译成 pattern 对象*

pattern = re.compile(r'\d+')

*# 使用 re.match 匹配文本，获得匹配结果，无法匹配时将返回 None*

result1 = re.match(pattern,'192abc')

**if** result1:

**print** result1.group()

**else**:

**print** '匹配失败1'

result2 = re.match(pattern,'abc192')

**if** result2:

**print** result2.group()

**else**:

**print** '匹配失败2'

运行结果如下：

192

匹配失败2

匹配 192abc 字符串时，match 函数是从字符串开头进行匹配，匹配到192立即返回值，通过 group（） 可以获取捕获的值。同样，匹配 abc192 字符串时，字符串开头不符合正则表达式，立即返回 None。

2.re.search（pattern，string[，flags]）

search 方法与 match 方法极其类似，区别在于 match（） 函数只从 string 的开始位置匹配，search（） 会扫描整个 string 查找匹配，match（） 只有在 string 起始位置匹配成功的时候才有返回，如果不是开始位置匹配成功的话，match（） 就返回 None。search 方法的返回对象和 match（） 返回对象在方法和属性上是一致的。示例如下：

**import** re

*# 将正则表达式编译成 pattern 对象*

pattern = re.compile(r'\d+')

*# 使用 re.match 匹配文本获得匹配结果；无法匹配时将返回 None*

result1 = re.search(pattern,'abc192edf')

**if** result1:

**print** result1.group()

**else**:

**print** '匹配失败1'

输出结果为：

192

3. re.split(pattern, string[, maxsplit])

按照能够匹配的子串将 string 分割后返回列表。maxsplit 用于指定最大分割次数，不指定，则将全部分割。示例如下：

**import** re

pattern = re.compile(r'\d+')

**print** re.split(pattern,'A1B2C3D4')

输出结果为：

\['A', 'B', 'C', 'D', ''\]

4.re.findall（pattern，string[，flags]）

搜索整个 string，以列表形式返回能匹配的全部子串。示例如下：

**import** re

pattern = re.compile(r'\d+')

**print** re.findall(pattern,'A1B2C3D4')

输出结果为：

['1', '2', '3', '4']

5.re.finditer（pattern，string[，flags]）

搜索整个 string，以迭代器形式返回能匹配的全部 Match 对象。示例如下：

**import** re

pattern = re.compile(r'\d+')

matchiter = re.finditer(pattern,'A1B2C3D4')

**for** match **in** matchiter:

**print** match.group()

输出结果为：

1

2

3

4

6.re.sub（pattern，repl，string[，count]）

使用 repl 替换 string 中每一个匹配的子串后返回替换后的字符串。当 repl 是一个字符串时，可以使用 \id 或\g<id>、\g<name>引用分组，但不能使用编号0。当 repl 是一个方法时，这个方法应当只接受一个参数（Match 对象），并返回一个字符串用于替换（返回的字符串中不能再引用分组）。count 用于指定最多替换次数，不指定时全部替换。示例如下：

**import** re

p = re.compile(r'(P<word1>\w+) (P<word2>\w+)')*# 使用名称引用*

s = 'i say, hello world!'

**print** p.sub(r'\g<word2> \g<word1>', s)

p = re.compile(r'(\w+) (\w+)')*# 使用编号*

**print** p.sub(r'\2 \1', s)

**def** **func**(m):

**return** m.group(1).title() + ' ' + m.group(2).title()

**print** p.sub(func, s)

输出结果为：

**say** i, world hello!

**say** i, world hello!

I Say, Hello World!

7.re.subn（pattern，repl，string[，count]）

返回（sub（repl，string[，count]），替换次数）。示例如下：

**import** re

s = 'i say, hello world!'

p = re.compile(r'(\w+) (\w+)')

**print** p.subn(r'\2 \1', s)

**def** **func**(m):

**return** m.group(1).title() + ' ' + m.group(2).title()

**print** p.subn(func, s)

输出结果为：

('say i, world hello!', 2)

('I Say, Hello World!', 2)

以上7个函数在 re 模块中进行搜索匹配，如何将捕获到的值提取出来呢？这就需要用到 Match 对象，之前已经使用了 Match 中的 groups 方法，现在介绍一下 Match 对象的属性和方法。

Match 对象的属性：

* string：匹配时使用的文本。
* re：匹配时使用的 Pattern 对象。
* pos：文本中正则表达式开始搜索的索引。值与 Pattern.match（） 和 Pattern.search（） 方法的同名参数相同。
* endpos：文本中正则表达式结束搜索的索引。值与 Pattern.match（） 和 Pattern.search（） 方法的同名参数相同。
* lastindex：最后一个被捕获的分组在文本中的索引。如果没有被捕获的分组，将为 None。
* lastgroup：最后一个被捕获的分组的别名。如果这个分组没有别名或者没有被捕获的分组，将为 None。
* Match 对象的方法：
* group（[group1，…]）：获得一个或多个分组截获的字符串，指定多个参数时将以元组形式返回。group1 可以使用编号也可以使用别名，编号0代表整个匹配的子串，不填写参数时，返回 group（0）。没有截获字符串的组返回 None，截获了多次的组返回最后一次截获的子串。
* groups（[default]）：以元组形式返回全部分组截获的字符串。相当于调用 group（1，2，…last）。default 表示没有截获字符串的组以这个值替代，默认为 None。
* groupdict（[default]）：返回以有别名的组的别名为键、以该组截获的子串为值的字典，没有别名的组不包含在内。default 含义同上。
* start（[group]）：返回指定的组截获的子串在 string 中的起始索引（子串第一个字符的索引）。group 默认值为0。
* end（[group]）：返回指定的组截获的子串在 string 中的结束索引（子串最后一个字符的索引+1）。group 默认值为0。
* span（[group]）：返回（start（group），end（group））。
* expand（template）：将匹配到的分组代入 template 中然后返回。template 中可以使用 \id 或 \g<id>、\g<name>引用分组，但不能使用编号0。\id 与\g<id>是等价的，但\10将被认为是第10个分组，如果你想表达\1之后是字符‘0’，只能使用\g<1>0。

示例如下：

**import** re

pattern = re.compile(r'(\w+) (\w+) (P<word>.\*)')

match = pattern.match( 'I love you!')

**print** "match.string:", match.string

**print** "match.re:", match.re

**print** "match.pos:", match.pos

**print** "match.endpos:", match.endpos

**print** "match.lastindex:", match.lastindex

**print** "match.lastgroup:", match.lastgroup

**print** "match.group(1,2):", match.group(1, 2)

**print** "match.groups():", match.groups()

**print** "match.groupdict():", match.groupdict()

**print** "match.start(2):", match.start(2)

**print** "match.end(2):", match.end(2)

**print** "match.span(2):", match.span(2)

**print** r"match.expand(r'\2 \1 \3'):", match.expand(r'\2 \1 \3')

输出结果：

match.string: I love you!

match.re: <\_sre.SRE\_Pattern object at 0x003F47A0>

match.pos: 0

match.endpos: 11

match.lastindex: 3

match.lastgroup: word

match.group(1,2): ('I', 'love')

match.groups(): ('I', 'love', 'you!')

match.groupdict(): {'word': 'you!'}

match.start(2): 2

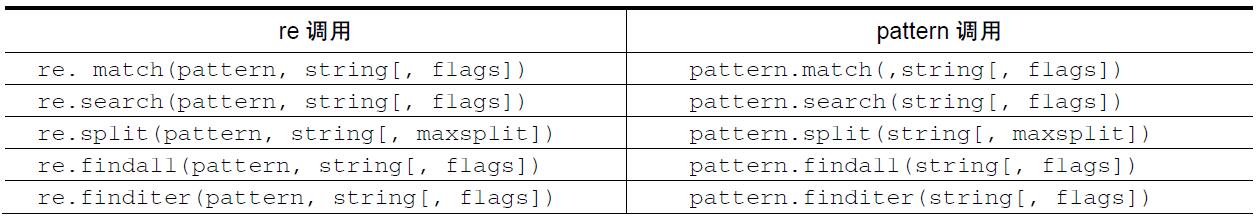
match.end(2): 6

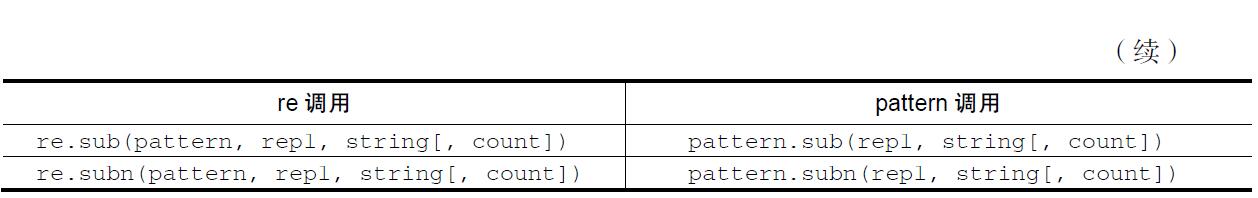
match.span(2): (2, 6)

match.expand(r'\2 \1 \3'): love I you!

前文介绍的7种方法的调用方式大都是 re.match、re.search 之类，其实还可以使用由 re.compile 方法产生的 Pattern 对象直接调用这些函数，类似 pattern.match，pattern.search，只不过不用将 Pattern 作为第一个参数传入。函数对比如表4-8所示。

表4-8　函数调用方式

[](http://popimage/?src=)

[](http://popimage/?src=)