爬虫08-闲谈正则表达式（三）

组的概念，还有正则中相对复杂的贪婪与非贪婪。

我们为什么要有组这个概念呢，因为一个字符串中，我们有时候只想要其中某一**连续的**满足某个条件的字符串。

例如content = '发布于2018/12/23'，我们需要提取出其中的发布时间，用之间的\d是提取不出来的，因为\d提取不出/，这个时候就可以用到组的概念了。

现在，我们来看看什么是组：

匹配规则（举例说明）	释义
（\d+）	（）内的内容构成一个组，只要当前位置满足\d+就匹配成功，返回（）内匹配成功的内容

老规矩，还是用代码来解释地好：

```
# 实现功能：提取出文章发布日期
import re
content = '发布于2018/12/23'
result = re.findall('.*?(\d.*\d)', content)
# 这一行的.*表示匹配除换行符外的任意字符，？表示非贪婪匹配，这个放在后面讲
# (\d.*\d)表示一个组，以数字开头，以数字结尾，.*表示中间可以是除换行以外的任意字符
# 最终返回的结果就是括号内匹配到的结果。
print(result)
# 得到的结果是['2018/12/23']

```

俗话说，没有对比就没有伤害，上述代码，如果不加括号呢，会是什么结果呢？

```
# 实现功能：提取发布时间
import re
content = '发布于2018/12/23'
result = re.findall('.*?\d.*\d', content)
print(result)
# 得到的结果是['发布于2018/12/23']
```

```
# 因为python默认会在正则表达式首尾各添加一个括号，第三行代码其实等价于
# result = re.findall(('.*?\d.*\d'), content)
```

当然了，正则表达式中是可以存在多个组的，例如下面这种情况：

```
# 实现功能：提取发布时间和发布人
import re
content = '发布于2018/12/23,发布人：十月'
result = re.findall('.*?(\d.*\d).*：(.*)', content)
```

# 这里的前一部分和上面的代码是一样的意思，两个括号之间的内容.*：表示中间是除换行符以外的任意字符，直到遇见：才终止，进入第二个组。
# 所以上述正则表达式的意思是：以除换行符以外的任意字符开头，直到遇见第一个组，以数字开头，以数字结尾，这样就能匹配到发布时间2018/12/23，然后又是除换行符外的任意字符，直到遇见：进入第二个组，分号后面所有的内容构成第二个组，匹配到发布人十月
print(result)
# 得到的结果是[('2018/12/23', '十月')]

会发现得到的结果好像和我们想的不一样呀，我们希望得到 ('2018/12/23', '十月') ，虽然上述结果我们也方便获取我们想要的内容，但是能不能直接获取到类似 ('2018/12/23', '十月') 呢？

这就需要用到re的另一个方法了，match方法。

实现功能：提取发布时间和发布人

```
import re
content = '发布于2018/12/23,发布人：十月。'
result = re.match('.*?(\d.*\d).*：(.*)', content)
# match方法的参数和findall是一样的，返回的结果是SRE_Match对象，对象有几个方法需要了解一下
print(result.group())
# 得到的结果是'发布于2018/12/23,发布人：十月'，该方法默认是result.group(0)
# 前文说过re.match('.*?(\d.*\d).*：(.*)', content)等价于
# re.match('(.*?(\d.*\d).*：(.*))', content)
# result.group(0)获取的内容就是最外层的括号匹配的内容。
print(result.group(1))
# 得到的结果是'2018/12/23'，获取的内容是(\d.*\d)匹配到的内容
print(result.group(2))
# 得到的结果是'十月'，获取的内容是(.*)匹配到的内容
```

```
print(result.groups())
# 得到的结果是('2018/12/23', '十月')
```

在用match方法的时候有一个需要注意的地方，很重要，非常容易导致出错，老规矩，用代码解释：

```
import re
content = '评论数：12'
result = re.match('\d', content)
print(result)
# 得到的结果是None，如果直接print(result.group())是会报错的。
# 原因在于match方法是从content第一个字符开始去匹配\d，如果未匹配到，直接就返回None。这里因为content第一个字符不是数字，所以直接返回None
```

到这里，我们的组就讨论完了，接下来我们进入正则的难点，也就是贪婪与非贪婪，老规矩，还是通过代码来解释，因为能更好地看到效果：

先来看下非贪婪模式：

```
# 实现功能：提取发布时间，比较贪婪与非贪婪
import re
content = '发布于2018/12/23'
result = re.findall('.*?(\d.*\d)', content)
# 这里的?表示的就是非贪婪模式，第一个.*会尽可能少地去匹配内容，因为后面跟的是\d，所以碰见第一个数字就终止了。
print(result)
# 得到的结果是['2018/12/23']
```

再来看下贪婪模式：

```
import re
content = '发布于2018/12/23'
result = re.findall('.*(\d.*\d)', content)
# 这里的第一个.*后面没有添加问号，表示的就是贪婪模式，第一个.*会尽可能多地去匹配内容，后面跟的是\d，碰见第一个数字并不一定会终止，当它匹配到2018的2的时候，发现剩下的内容依然满足(\d.*\d)，所以会一直匹配下去，直到匹配到12后面的/的时候，发现剩下的23依然满足(\d.*\d)，但是如果再匹配下去，匹配到23的2的话，剩下的3就不满足(\d.*\d)了，所以第一个.*就会停止匹配，(\d.*\d)最终匹配到的结果就只剩下23了。
print(result)
# 得到的结果是['23']
```

```
7
8  我们再来看下这段代码：
9
10 import re
11 content = '发布于2018/12/23'
12 result = re.findall('.*(\d.*?\d)', content)
13 print(result)
14 # 得到的结果是['23']，原因在于第一个.*是贪婪模式，会一直匹配到12后面的/，这样结果就是['23']
15
```

再来看下这段代码：

```
1 import re
2 content = '发布于2018/12/23'
3 result = re.findall('.*?(\d.*?\d)', content)
4 # 这里的第一个.*?表示非贪婪模式，匹配到2018前面的'于'之后就停止了
5 # 括号里的.*?也是表示非贪婪模式，括号里的内容从2018的2开始匹配，因为后面一个数字是0，那么也就满足了(\d.*?\d)，所以就直接返回结果了，同样的，接下来的18也是这样，一直匹配到23才结束。
6 print(result)
7 # 得到的结果是['20', '18', '12', '23']
```

简单来说，贪婪模式就是尽可能多地去匹配字符，非贪婪模式就是尽可能少地去匹配字符，python默认采取的是贪婪模式。

贪婪与非贪婪在使用的要慎重，因为一不小心就容易出错，匹配到的结果并不是我们想要的。