爬虫09-闲谈正则表达式（四）

讨论re剩下的两个方法，分别是sub方法和search方法。

我们先来看看简单的search方法，可以将他和match做个比较，老规矩，直接上代码：

```
# 实现功能：比较search方法和match方法的区别
import re
content = '点赞数：12'
result_match = re.match('\d', content)
result_search = re.search('\d', content)
print(result_match)
# 得到的结果是None
print(result_search)
# 得到的结果是<_sre.SRE_Match object; span=(4, 5), match='1'>
 print(result_search.group())
 # 得到的结果是'1'

```

可以看到，使用match方法，会从content的开头去匹配\d，没有匹配到就直接返回None了。

而search方法也是从头开始匹配，只要匹配到有一个字符符合\d，就直接返回了，不会继续往下匹配。

search方法返回的也是一个SRE_Match对象，和match方法的取值是一样的，用group()。

接下来，我们来看下sub方法，这个方法能实现的功能是匹配出结果并替换掉内容，我们直接上代码：

```
# 实现功能：将content中的php全部替换成python
import re
content = 'python php java c javascript java php'
result = re.sub('php', 'python', content)
# sub方法的第一个参数是正则表达式，第二个参数是替换之后的字符串，第三个参数是目标字符串
print(result)
# 得到的结果是'python python java c javascript java python'
```

我们再来看下面这段代码：

```
# 实现功能：将content中的php全部替换成python,php不区分大小写
import re
content = 'python PHP java c javascript java php'
result = re.sub('php', 'python', content)
# 这一行没有加上flags=re.I，第一个参数php就只会匹配小写的php，大写的PHP匹配不到
result1 = re.sub('php', 'python', content, flags=re.I)
# 这一行的flags=re.I表示的是匹配模式，re.I表示第一个参数不区分大小写
# 还有一种常用的匹配模式re.S，在这种模式下.表示任意字符。普通模式下.表示除换行符外的任意字符
print(result)
 # 得到的结果是'python PHP java c javascript java python'
 print(result1)
 # 得到的结果是'python python java c javascript java python'

```

再来看下面这段代码：

```
# 实现功能：将第一个出现的php替换成python
import re
content = 'python PHP java c javascript java php'
result = re.sub('php', 'python', content, flags=re.I)
# re.sub的第四个参数是count，默认count=0，表示无论匹配到多少个php，都替换成python
result1 = re.sub('php', 'python', content, count=1, flags=re.I)
# 这里的count=1表示无论匹配到多少个php，最多只将第1个php替换成python，count=8的话就表示无论匹配到多少个php，最多只将前8个php替换成python
print(result)
# 得到的结果是'python python java c javascript java python'
 print(result1)
 # 得到的结果是'python python java c javascript java php'

```

接下来，我们来看下sub方法设计的精妙之处，就是sub的第二个参数可以是一个函数。

精妙之处在哪呢，就在于当你拿到匹配结果的时候，不一定要将它替换成固定的字符串，你可以传递一个函数，在函数中对匹配结果进行逻辑处理，这样主动权就交到了用户手上，用户可以随便处理。

我们来看下面的例子：

```
# 实现功能：将学生的分数替换成相应的等级，0-59分为不及格， 60-79分为中，80-89分为良，90-100分为优
# 这个功能用字符串的replace方法还是会比较复杂的。
import re

def judge(value):
 value = value.group()
 # 用group方法获取到匹配结果，以下逻辑是对value进行逻辑判断
 if int(value) < 60:
 return '不及格'
 elif int(value) < 80:
 return '中'
 elif int(value) < 90:
 return '良'
 else:
 return '优'

content = '小明：59 小红：66 小白：83 小绿：98 小王：100'
result = re.sub('\d+', judge, content)
# 这里sub的第二个参数就是一个函数judge，第一个参数匹配到的结果会作为value传递进judge函数中，从而在judge中可以对他进行判断，函数的返回值将会替换掉匹配结果。
print(result)
# 得到的结果是'小明：不及格  小红：中  小白：良  小绿：优  小王：优'
```

这个sub方法的使用并不难，可有一点值得我们去学习，那就是将函数作为一个参数传递到另一个函数中，这是一种非常经典的设计，值得大家去思考学习。