3.3 正则表达式

本节主要介绍正则表达式的相关用法，它可以实现字符串的检索、替换和匹配验证，从HTML中提取想要的信息。

一、什么是正则表达式

1. 正则表达式：正则表达式是对字符串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个"规则字符串"，这个规则字符串用来表达对字符串的一种过滤逻辑。

2. 正则表达式的匹配规则：

a-z 表示匹配任意的小写字母，\s 表示匹配任意的空白字符，* 表示匹配前面的字符任意多个。

※其余常见的匹配规则详见 P140

3. 正则表达式并非Python独有，在Python中主要是通过re库来实现的。

二、match( )

1. match( )是re库中的一个匹配方法，它可以检测这个正则表达式是否匹配字符串。

2. 基础实例：

```
import re
content = 'Hello 123 4567 World_This is a Regex Demo'
print (len(content)) ⇒ 打印字符串长度
result = re.match('^Hello\s\d\d\d\s\d{4}\s\w{10}', content)
print (result) ⇒ 返回结果是SRE_Match对象，说明匹配成功
print (result.group( ))
print (result. span( ))
```

match( )方法需要两个参数，一个是正则表达式，一个是要匹配的字符串

SRE_Match对象的group( )方法可以输出匹配到的内容。

SRE_Match对象的span( )方法可以输出匹配的范围。

3. 匹配目标。

(1) 如果想从字符串中得到某一部分内容，我们可以将想提取的子字符串用括号( )括起来。

(2) ( )实际上标记了一个子表达式的开始和结束位置，被标记的每个子表达式会依次对应每一个分组group，调用group( )方法传入分组的索引即可获取相应分组的结果

eg: group(1) 即表示第一个括号内的匹配结果。

4. 通用匹配

(1) 通用匹配(万能匹配): .* (点星)

其中.(点)可以匹配除换行符外的任意字符,而*(星)代表匹配前面的字符任意次

⇒ .* 可以匹配任意字符.

(2) 实例:

```
import re
content = 'Hello 123 4567 World_This is a Regex Demo'
result = re.match('^Hello.*Demo$', content)
print(result)
print(result.group())
print(result.span())
```

5. 贪婪与非贪婪

(1) 实例: 对于字符串 Hello 123 4567 World_This is a Regex Demos

① 如果正则表达式为 ^He.*(\d+).*Demos$,此时是贪婪匹配,.*会匹配尽可能多的字符,最终得到 7

② 如果正则表达式为 ^He.*?(\d+).*Demos$,此时是非贪婪匹配,.*?会匹配尽可能少的字符,最终得到 1234567

(2) 应用: 在做匹配时,字符串中间尽量使用非贪婪匹配,以免出现匹配结果缺失的情况;相反,当要匹配的结果出现在字符串尾部时,.*?有可能匹配不到任何内容,这时需要改用.*

6. 修饰符

(1) 实例: 当字符串中含有换行符时,之前的正则表达式就不起作用了,因为.匹配的是除换行符以外的任意字符,要解决该问题就需要在match()方法内加入修饰符 re.S

(2) 常用修饰符:

re.I	使匹配对大小写不敏感
re.L	做本地化识别(locale-aware)匹配
re.M	多行匹配,影响^和$
re.S	使.匹配包括换行在内的所有字符
re.U	根据Unicode字符集解析字符。它影响\w、\W、\b和\B
re.X	该标志通过给予你更灵活的格式以便你将正则表达式写的更易于理解

7. 转义匹配

当遇到用于正则匹配模式的特殊字符时，在前面加反斜线转义一下就可以。

比如：.匹配除换行符以外的任意字符，但如果目标字符中就包含.，这时就不好匹配了，此此时用\.就可以匹配.这个字符。

三、search()

1. match()的局限性：match()方法只能从字符串的开头进行匹配，一旦开头对不上，就会匹配失败，因此，match()方法更适合用来检验字符串与正则表达式是否匹配。

2. search()：search()方法会在匹配时扫描整个字符串，然后返回第一个成功匹配的结果。如果搜索完了还没有找到匹配的结果，就会返回None.

实例：见书P146.

具体用法与match()类似

注意：在匹配过程中，由于绝大多数的HTML文本都带有换行符，所以使用match()或search()时，我们尽量加上re.S修饰符.

四、findall()

1. 与search()方法不同，findall()方法会搜索整个字符串，然后返回匹配正则表达式的所有内容.

2. findall()方法的返回结果是列表类型.

五、sub()

1. sub()的作用是用来修改文本

2. 实例：删去字符串中的所有数字.

```
import re
content = '54ak54yr50iR54ix5L2g'
content = re.sub('\d+', '', content)
print(content)
```

⇒ akyroiRixLg

3. sub()的参数设置

sub('正则表达式', '想要替换成的东西', 字符串)

六. complile()

1. complile()方法可以将正则字符串编译成正则表达式对象. 它的作用相当于给正则表达式做了一层封装,以便于在后面的匹配中复用.

2. 实例:

```
import re
content1 = '2016-12-15 12:00'
content2 = '2016-12-17 12:55'
content3 = '2016-12-22 13:21'
pattern = re.compile('/d{2}:/d{2}')
result1 = re.sub(pattern,'',content1)
result2 = re.sub(pattern,'',content2)
result3 = re.sub(pattern,'',content3)
print(content1, content2,
print(result1, result2, result3)
```

3. compile()方法中也可以传入修饰符.