一篇搞定Python正则表达式

GreatAnt

1. 正则表达式语法

1.1 字符与字符类

1 特殊字符：. ^$?+*{} [] () |

以上特殊字符要想使用字面值，必须使用进行转义

2 字符类

1. 包含在[]中的一个或者多个字符被称为字符类，字符类在匹配时如果没有指定量词则只会匹配其中的一个。

2. 字符类内可以指定范围，比如[a-zA-Z0-9]表示a到z，A到Z，0到9之间的任何一个字符

3. 左方括号后跟随一个^，表示否定一个字符类，比如[^0-9]表示可以匹配一个任意非数字的字符。

4. 字符类内部，除了之外，其他特殊字符不再具备特殊意义，都表示字面值。^放在第一个位置表示否定，放在其他位置表示^本身，-放在中间表示范围，放在字符类中的第一个字符，则表示-本身。

5. 字符类内部可以使用速记法，比如d s w

3 速记法

. 可以匹配除换行符之外的任何字符，如果有re.DOTALL标志，则匹配任意字符包括换行

d匹配一个Unicode数字，如果带re.ASCII，则匹配0-9

D 匹配Unicode非数字

s匹配Unicode空白，如果带有re.ASCII，则匹配 中的一个

S 匹配Unicode非空白

w匹配Unicode单词字符，如果带有re.ascii,则匹配[a-zA-Z0-9_]中的一个

W 匹配Unicode非单子字符

1.2 量词

1. ?匹配前面的字符0次或1次
2. *匹配前面的字符0次或多次
3. +匹配前面的字符1次或者多次
4. {m}匹配前面表达式m次
5. {m,}匹配前面表达式至少m次
6. {,n}匹配前面的正则表达式最多n次
7. {m,n}匹配前面的正则表达式至少m次，最多n次

注意点：

以上量词都是贪婪模式，会尽可能多的匹配，如果要改为非贪婪模式，通过在量词后面跟随一个?来实现

1.3 组与捕获

1 ()的作用：

1. 捕获()中正则表达式的内容以备进一步利用处理，可以通过在左括号后面跟随?:来关闭这个括号的捕获功能

2. 将正则表达式的一部分内容进行组合，以便使用量词或者|

2 反响引用前面()内捕获的内容：

1. 通过组号反向引用

每一个没有使用?:的小括号都会分配一个组好，从1开始，从左到右递增，可以通过i引用前面()内表达式捕获的内容

2. 通过组名反向引用前面小括号内捕获的内容

可以通过在左括号后面跟随?P<name>,尖括号中放入组名来为一个组起一个别名，后面通过(?P=name)来引用 前面捕获的内容。如(?P<word>w+)s+(?P=word)来匹配重复的单词。

3 注意点：

反向引用不能放在字符类[]中使用。

1.4 断言与标记

断言不会匹配任何文本，只是对断言所在的文本施加某些约束

1 常用断言：

1. 匹配单词的边界，放在字符类[]中则表示backspace
2. B 匹配非单词边界，受ASCII标记影响
3. A 在起始处匹配
4. ^在起始处匹配，如果有MULTILINE标志，则在每个换行符后匹配
5. Z 在结尾处匹配
6. $在结尾处匹配，如果有MULTILINE标志，则在每个换行符前匹配
7. (?=e)正前瞻
8. (?!e)负前瞻
9. (?<=e)正回顾
10. (?<!e)负回顾

2 前瞻回顾的解释

前瞻： exp1(?=exp2) exp1后面的内容要匹配exp2

负前瞻: exp1(?!exp2) exp1后面的内容不能匹配exp2

后顾: (?<=exp2)exp1 exp1前面的内容要匹配exp2

负后顾: (?<!exp2)exp1 exp1前面的内容不能匹配exp2

例如：我们要查找hello，但是hello后面必须是world，正则表达式可以这样写："(hello)s+(?=world)",用来匹配"hello wangxing"和"hello world"只能匹配到后者的hello

1.5 条件匹配

(?(id)yes_exp|no_exp):对应id的子表达式如果匹配到内容，则这里匹配yes_exp，否则匹配no_exp

1.6 正则表达式的标志

1. 正则表达式的标志有两种使用方法

1. 通过给compile方法传入标志参数，多个标志使用|分割的方法,如re.compile(r"#[da-f]{6}", re.IGNORECASE|re.MULTILINE)

2. 通过在正则表达式前面添加(?标志)的方法给正则表达式添加标志，如(?ms)#[da-z]{6}

2. 常用的标志

re.A或者re.ASCII， 使 B s S w W d D都假定字符串为假定字符串为ASCII

re.I或者re.IGNORECASE使正则表达式忽略大小写

re.M或者re.MULTILINE 多行匹配，使每个^在每个回车后，每个$在每个回车前匹配

re.S或者re.DOTALL 使.能匹配任意字符，包括回车

re.X或者re.VERBOSE 这样可以在正则表达式跨越多行，也可以添加注释，但是空白需要使用s或者[ ]来表示，因为默认的空白不再解释。如：

```
re.compile(r"""
    <imgs +)#标签的开始
    [^>]*?#不是src的属性
    src=#src属性的开始
    (?:
    (?P<quote>["'])#左引号
    (?P<image_name>[^>]+?)#图片名字
    (?P=quote)#右括号
    """,re.VERBOSE|re.IGNORECASE)
```

2. Python正则表达式模块

2.1 正则表达式处理字符串主要有四大功能

1. 匹配 查看一个字符串是否符合正则表达式的语法，一般返回true或者false

2. 获取正则表达式来提取字符串中符合要求的文本

3. 替换查找字符串中符合正则表达式的文本，并用相应的字符串替换

4. 分割使用正则表达式对字符串进行分割。

2.2 Python中re模块使用正则表达式的两种方法

1. 使用re.compile(r, f)方法生成正则表达式对象，然后调用正则表达式对象的相应方法。这种做法的好处是生成正则对象之后可以多次使用。

2. re模块中对正则表达式对象的每个对象方法都有一个对应的模块方法，唯一不同的是传入的第一个参数是正则表达式字符串。此种方法适合于只使用一次的正则表达式。

2.3 正则表达式对象的常用方法

1. rx.findall(s,start, end):

返回一个列表，如果正则表达式中没有分组，则列表中包含的是所有匹配的内容，

如果正则表达式中有分组，则列表中的每个元素是一个元组，元组中包含子分组中匹配到的内容，但是没有返回整个正则表达式匹配的内容

2. rx.finditer(s, start, end):

返回一个可迭代对象

对可迭代对象进行迭代，每一次返回一个匹配对象，可以调用匹配对象的group()方法查看指定组匹配到的内容，0表示整个正则表达式匹配到的内

容

3. rx.search(s, start, end):

返回一个匹配对象,倘若没匹配到，就返回None

search方法只匹配一次就停止，不会继续往后匹配

4. rx.match(s, start, end):

如果正则表达式在字符串的起始处匹配，就返回一个匹配对象，否则返回None

5. rx.sub(x, s, m):

返回一个字符串。每一个匹配的地方用x进行替换，返回替换后的字符串，如果指定m，则最多替换m次。对于x可以使用/i或者/g<id>id可以是组名或者编号来引用捕获到的内容。

模块方法re.sub(r, x, s, m)中的x可以使用一个函数。此时我们就可以对捕获到的内容推过这个函数进行处理后再替换匹配到的文本。

6. rx.subn(x, s, m):

与re.sub()方法相同，区别在于返回的是二元组，其中一项是结果字符串，一项是做替换的个数。

7. rx.split(s, m):分割字符串

返回一个列表

用正则表达式匹配到的内容对字符串进行分割

如果正则表达式中存在分组，则把分组匹配到的内容放在列表中每两个分割的中间作为列表的一部分，如：

```
rx = re.compile(r"(d)[a-z]+(d)")

s = "ab12dk3klj8jk9jks5"

result = rx.split(s)
```

返回['ab1', '2', '3', 'klj', '8', '9', 'jks5']

8. rx.flags():正则表达式编译时设置的标志

9. rx.pattern():正则表达式编译时使用的字符串

2.4 匹配对象的属性与方法

01. m.group(g, ...)

返回编号或者组名匹配到的内容，默认或者0表示整个表达式匹配到的内容，如果指定多个，就返回一个元组

02. m.groupdict(default)

返回一个字典。字典的键是所有命名的组的组名，值为命名组捕获到的内容

如果有default参数，则将其作为那些没有参与匹配的组的默认值。

03. m.groups(default)

返回一个元组。包含所有捕获到内容的子分组，从1开始，如果指定了default值，则这个值作为那些没有捕获到内容的组的值

04. m.lastgroup()

匹配到内容的编号最高的捕获组的名称，如果没有或者没有使用名称则返回None(不常用)

05. m.lastindex()

匹配到内容的编号最高的捕获组的编号，如果没有就返回None。

06. m.start(g):

当前匹配对象的子分组是从字符串的那个位置开始匹配的,如果当前组没有参与匹配就返回-1

07. m.end(g)

当前匹配对象的子分组是从字符串的那个位置匹配结束的，如果当前组没有参与匹配就返回-1

08. m.span()

返回一个二元组，内容分别是m.start(g)和m.end(g)的返回值

09. m.re()

产生这一匹配对象的正则表达式

10. m.string()

传递给match或者search用于匹配的字符串

11. m.pos()

搜索的起始位置。即字符串的开头，或者start指定的位置(不常用)

12. m.endpos()

搜索的结束位置。即字符串的末尾位置，或者end指定的位置(不常用)

2.5 总结

1. 对于正则表达式的匹配功能，Python没有返回true和false的方法，但可以通过对match或者search方法的返回值是否是None来判断

2. 对于正则表达式的搜索功能，如果只搜索一次可以使用search或者match方法返回的匹配对象得到，对于搜索多次可以使用finditer方法返回的可迭代对象来迭代访问

3. 对于正则表达式的替换功能，可以使用正则表达式对象的sub或者subn方法来实现，也可以通过re模块方法sub或者subn来实现，区别在于模块的sub方法的替换文本可以使用一个函数来生成

4. 对于正则表达式的分割功能，可以使用正则表达式对象的split方法，需要注意如果正则表达式对象有分组的话，分组捕获的内容也会放到返回的列表中

作者：GreatAnt

来源:http://www.cnblogs.com/greatfish/p/7572131.html

- END -

[阅读原文]