Python专题教程：正则表达式re模块详解

版本：v1.0

Crifan Li

摘要

本文主要介绍了Python中的正则表达式re模块，详细解释其用法，包括其下各种函数，比如re.findall,re.search,re.match等等。

本文提供多种格式供：

在线阅读	HTML [1]	HTMLs [2]	PDF [3]	CHM [4]	TXT [5]	RTF [6]	WEBHELP [7]
下载（7zip压缩包）	HTML [8]	HTMLs [9]	PDF [10]	CHM [11]	TXT [12]	RTF [13]	WEBHELP [14]

HTML版本的在线地址为：

http://www.crifan.com/files/doc/docbook/python_topic_re/release/html/python_topic_re.html

有任何意见，建议，提交bug等，都欢迎去讨论组发帖讨论：

http://www.crifan.com/bbs/categories/python_topic_re/

修订历史		
修订 1.0	2013-09-05	crl
1. 将之前在正则表达式学习心得，Python语言总结中和Python相关的re模块的内容，都整理过来了。		

[1] http://www.crifan.com/files/doc/docbook/python_topic_re/release/html/python_topic_re.html
[2] http://www.crifan.com/files/doc/docbook/python_topic_re/release/htmls/index.html
[3] http://www.crifan.com/files/doc/docbook/python_topic_re/release/pdf/python_topic_re.pdf
[4] http://www.crifan.com/files/doc/docbook/python_topic_re/release/chm/python_topic_re.chm
[5] http://www.crifan.com/files/doc/docbook/python_topic_re/release/txt/python_topic_re.txt
[6] http://www.crifan.com/files/doc/docbook/python_topic_re/release/rtf/python_topic_re.rtf
[7] http://www.crifan.com/files/doc/docbook/python_topic_re/release/webhelp/index.html
[8] http://www.crifan.com/files/doc/docbook/python_topic_re/release/html/python_topic_re.html.7z
[9] http://www.crifan.com/files/doc/docbook/python_topic_re/release/htmls/index.html.7z
[10] http://www.crifan.com/files/doc/docbook/python_topic_re/release/pdf/python_topic_re.pdf.7z
[11] http://www.crifan.com/files/doc/docbook/python_topic_re/release/chm/python_topic_re.chm.7z
[12] http://www.crifan.com/files/doc/docbook/python_topic_re/release/txt/python_topic_re.txt.7z
[13] http://www.crifan.com/files/doc/docbook/python_topic_re/release/rtf/python_topic_re.rtf.7z
[14] http://www.crifan.com/files/doc/docbook/python_topic_re/release/webhelp/python_topic_re.webhelp.7z

Python专题教程：正则表达式re模块详解:

Crifan Li

版本：v1.0

出版日期 2013-09-05

[15] http://www.crifan.com/files/doc/docbook/soft_dev_basic/release/html/soft_dev_basic.html#cc_by_nc

目录

表格清单

前言

1. 本文目的

本文目的在于，介绍Python中的正则表达式re模块的详细用法

包括常见的re.search，re.findall等函数的详细用法举例和注意事项

2. 待完成

将下面帖子内容整理合并进来：

- 【总结】关于（C#和Python中的）正则表达式 [1]

之前抽空写了，Python中的正则表达式的系列教程。

暂时没写完，但是也算写了不少了。

现在整理出，已经写出的部分，供参考：

【教程】详解Python正则表达式 [2]

【教程】详解Python正则表达式之：'.' dot 点 匹配任意单个字符 [3]

【教程】详解Python正则表达式之：'^' Caret 脱字符/插入符 匹配字符串开始 [4]

【教程】详解Python正则表达式之：'$' dollar 美元符号 匹配字符串末尾 [5]

【教程】详解Python正则表达式之：'*' star 星号 匹配0或多个 [6]

【教程】详解Python正则表达式之：[] bracket 中括号 匹配某集合内的字符 [7]

【教程】详解Python正则表达式之：'|' vertical bar 竖杠 [8]

【教程】详解Python正则表达式之：(...) group 分组 [9]

【教程】详解Python正则表达式之：(?...) extension notation 扩展助记符 [10]

【教程】详解Python正则表达式之：(?:...) non-capturing group 非捕获组 [11]

【教程】详解Python正则表达式之：(?P<name>...) named group 带命名的组 [12]

【教程】详解Python正则表达式之：(?P=name) match earlier named group 匹配前面已命名的组 [13]

[1] http://www.crifan.com/summary_regular_expression_csharp_python/
[2] http://www.crifan.com/detailed_explanation_about_python_regular_express
[3] http://www.crifan.com/detailed_explanation_about_python_regular_express_dot_match_any_single_char
[4] http://www.crifan.com/detailed_explanation_about_python_regular_express_caret_match_string_start
[5] http://www.crifan.com/detailed_explanation_about_python_regular_express_dollar_match_string_end
[6] http://www.crifan.com/detailed_explanation_about_python_regular_express_match_zero_or_more
[7] http://www.crifan.com/detailed_explanation_about_python_regular_express_match_a_set_of_chars
[8] http://www.crifan.com/detailed_explanation_about_python_regular_express_about_vertical_bar/
[9] http://www.crifan.com/detailed_explanation_about_python_regular_express_about_group/
[10] http://www.crifan.com/detailed_explanation_about_python_regular_express_extension_notation
[11] http://www.crifan.com/detailed_explanation_about_python_regular_express_non_capturing_group
[12] http://www.crifan.com/detailed_explanation_about_python_regular_express_named_group
[13] http://www.crifan.com/detailed_explanation_about_python_regular_express_match_named_group

【教程】详解Python正则表达式之：(?(id/name)yes-pattern|no-pattern) 条件性匹配 [14]

【教程】详解Python正则表达式之：(?=...) lookahead assertion 前向匹配 /前向断言 [15]

【教程】详解Python正则表达式之：(?!...) negative lookahead assertion 前向否定匹配 /前向否定断言 [16]

【教程】详解Python正则表达式之：(?<=...) positive lookbehind assertion 后向匹配 /后向断言 [17]

【教程】详解Python正则表达式之：\s 匹配任一空白字符 [18]

【教程】详解Python正则表达式之：re.LOCALE re.L 本地化标志 [19]

【教程】详解Python正则表达式之：re.UNICODE re.U 统一码标志 [20]

另外，也针对re模块中的一些功能，比如findall，进行了整理：

【整理】Python中的re.search和re.findall之间的区别和联系 + re.finall中带命名的组，不带命名的组，非捕获的组，没有分组四种类型之间的区别 [21]

还有些和group相关的内容：

【已解决】Python中的正则re查找中，从多个匹配的组中获得所有的匹配的值 [22]

[14] http://www.crifan.com/detailed_explanation_about_python_regular_express_yes_or_no_conditional_match
[15] http://www.crifan.com/detailed_explanation_about_python_regular_express_lookahead_assertion
[16] http://www.crifan.com/detailed_explanation_about_python_regular_express_negative_lookahead_assertion
[17] http://www.crifan.com/detailed_explanation_about_python_regular_express_positive_lookbehind_assertion
[18] http://www.crifan.com/detailed_explanation_about_python_regular_express_match_any_whitespace_char
[19] http://www.crifan.com/detailed_explanation_about_python_regular_express_flag_re_locale_re_l
[20] http://www.crifan.com/detailed_explanation_about_python_regular_express_flag_re_unicode_re_u
[21] http://www.crifan.com/python_re_search_vs_re_findall/
[22] http://www.crifan.com/python_re_find_get_multiple_match_object_value/

第 1 章 Python正则表达式re模块简介

1.1. 什么是Python的re

正则表达式，是一门相对通用的语言。

Python中也有对正则表达式的支持，

对应的就是Python内置的re模块。

关于正则表达式

简单说就是：

用一系列的规则语法，去匹配，查找，替换等操作字符串，

以达到对应的目的

此套规则，就是所谓的正则表达式

更详细的解释参见详细的教程：

正则表达式学习心得 [1]

Python中的正则表达式模块，即re模块，功能还是很强大的。

其支持常见的查找替换等功能，对应的是re.search，re.findall等函数。

详见后续的解释。

[1] http://www.crifan.com/files/doc/docbook/regular_expression/release/html/regular_expression.html

第 2 章 Python中正则表达式的语法

其实，Python中的正则表达式的语法，

和通用的正则表达式的语法，

正则表达式的通用语法 [1]

基本没太大区别。

下面，再详细的解释一下，Python中的正则表达式的语法：

2.1. Python中的正则表达式的特点

下面总结一些Python中的正则表达式相对于其他语言中的正则表达式的一些特点，包括优点和缺点：

1. python中字符串的表示，单引号和双引号，都是支持的。
   所以对于字符串中，有双引号的，可以在写字符串最外层用单引号括起来，而不需要用反斜杠了。

   反之，如果需要表示的其中包括单引号，那么最外层用双引号，所以，还是很方便的。

2. 对于匹配多个字符串的时候，好像不能加括号分组的，如果加括号分组了，那么只能匹配单个一个group就结束了。对应的要匹配多个字符串，好像只能使用findall。

2.2. Python正则表达式的语法

其实，其详细语法，可以参考Python自带的帮助（help）文件

可以通过在帮助文件的搜索框中输入re，然后就可以找到“(re.MatchObject　attribute)”，双击，即调转到对应的re模块的内容的详细解释部分了。

2.2.1. re模块中的语法总结

关于re模块的基本语法，简单总结如下：

表 2.1. Python中re模块中的特殊字符

.	匹配任意字符
[]	用来匹配一个指定的字符类别，所谓的字符类别就是你想匹配的一个字符集，对于字符集中的字符可以理解成或的关系
^	对于字符串，表示字符串的开头 对于^加上一个其他数字或字符，表示取反。比如，[^5]表示除了5之外的任意字符。 [^^]表示除了^字符之外任意字符。
$	匹配字符串的末尾，或者匹配换行之前的字符串末尾
*	对于前一个字符重复0到无穷次
+	对于前一个字符重复1到无穷次
?	对于前一个字符重复0到1次
{m,n}	对于前一个字符重复次数在为m到n次。 {0,} == * {1,} == {0,1} == ?

[1] http://www.crifan.com/files/doc/docbook/regular_expression/release/htmls/regular_expression_grammar.html

	{m} 对于前一个字符重复m次

表 2.2. Python中re模块中特殊转义序列（字符）

\A	匹配字符串的开头
\b	匹配一个空字符（仅对一个单词word的开始或结束有效）
\B	与\b含义相反
\d	匹配任何十进制数；它相当于类 [0-9]
\D	匹配任何非数字字符；它相当于类 [^0-9]
\s	匹配任何空白字符；它相当于类 [\t\n\r\f\v]
\S	匹配任何非空白字符；它相当于类 [^ \t\n\r\f\v]
\w	匹配任何字母数字字符；它相当于类 [a-zA-Z0-9_]
\W	匹配任何非字母数字字符；它相当于类 [^a-zA-Z0-9_]
\Z	匹配字符串的结尾

第 3 章 Python中的re.search

此处介绍，Python中的正则表达式模块re中search函数的详细使用方法。

即对应的re.search的功能和用法

第 4 章 Python中的re.findall

此处介绍，Python中的正则表达式模块re中findall函数的详细使用方法。

即对应的re.findall的功能和用法

第 5 章 Python中的re.match

此处介绍，Python中的正则表达式模块re中match函数的详细使用方法。

即对应的re.match的功能和用法

第 6 章 Python中正则表达式的使用心得

此处整理一下，Python中使用正则表达式的心得：

6.1. re模块搜索时要注意竖线"|"的使用

某次，对于字符串

```
footerUni=u"分类： | 标签：";
```

使用：

```
foundCatZhcn = re.search(u"分类：(?P<catName>.+)|", footerUni);
print "foundCatZhcn=",foundCatZhcn;
if(foundCatZhcn):
    print "foundCatZhcn.group(0)=",foundCatZhcn.group(0);
    print "foundCatZhcn.group(1)=",foundCatZhcn.group(1);
    catName = foundCatZhcn.group("catName");
    print "catName=",catName;
```

所得到的结果却是：

```
foundCatZhcn= <_sre.SRE_Match object at 0x027E3C20>
foundCatZhcn.group(0)=
foundCatZhcn.group(1)= None
catName= None
```

其中group(0)，不是所期望的整个匹配的字符串，且group(1)应该是一个空格的字符，而不是None。

调试了半天，最后终于找到原因了，原来是在正则搜索中，竖线"|"，是or的关系

“

'|'

A|B, where A and B can be arbitrary REs, creates a regular expression that will match either A or B. An arbitrary number of REs can be separated by the '|' in this way. This can be used inside groups (see below) as well. As the target string is scanned, REs separated by '|' are tried from left to right. When one pattern completely matches, that branch is accepted. This means that once A matches, B will not be tested further, even if it would produce a longer overall match. In other words, the '|' operator is never greedy. To match a literal '|', use \|, or enclose it inside a character class, as in [|].

”

所以此处匹配到的结果是空值

所以测试过程中，无论如何修改re中的表达式，也都会得到foundCatZhcn是非空的值

然后对应的解决办法是，给竖线加上反斜杠，表示竖线字符本身：

```
foundCatZhcn = re.search(u"分类：(?P<catName>.*?)\|", footerUni);
```

这样才能真正自己想要的效果。

6.2. re模块的search的含义和用法及查找后group的含义

参考这里[1]：

Match Object Methods	Description
group(num=0)	This methods returns entire match (or specific subgroup num)
groups()	This method return all matching subgroups in a tuple (empty if there weren’ t any)

知道了，原来group(0)，是所有匹配的内容，而group(N)指的是原先subgroup子组对应的内容，而subgroup是原先search等规则中，用括号()所括起来的。

举例1：

```
#!/usr/bin/python
import re
line = "Cats are smarter than dogs";
matchObj = re.search( r'(.*) are(\.*)', line, re.M|re.I)
if matchObj:
  print "matchObj.group() : ", matchObj.group()
  print "matchObj.group(1) : ", matchObj.group(1)
  print "matchObj.group(2) : ", matchObj.group(2)
else:
  print "No match!!"
```

输出是：

```
matchObj.group(): Cats are
 matchObj.group(1) : Cats
 matchObj.group(2) :
```

举例2：字符串：

```
var pre = [false,'', '','\/recommend_music/blog/item/.html'];
```

然后去search：

```
match = re.search(r"var pre = \[(.*?),.*?,.*?,'(.*?)'\]", page,
 re.DOTALL | re.IGNORECASE | re.MULTILINE)print "match(0)=",
 match.group(0),"match(1)=",match.group(1),"match(2)=",match.group(2),"match(3)=",match.group(3)
```

得到的输出是：

[1] http://www.tutorialspoint.com/python/python_reg_expressions.htm

```
match(0)= var pre = [false,'', '','\/recommend_music/blog/item/.html']
match(1)= false
match(2)= \/recommend_music/blog/item/.html
match(3)=
```

6.3. re模块的findall的模式（pattern）中是否加括号的区别

关于search的结果，第 6.2 节 “re模块的search的含义和用法及查找后group的含义”中已经解释过了。

下面详细给出关于findall中，对于pattern中，加括号，与不加括号，所查找到的结果的区别。

其中加括号，表示（）内的匹配的内容为一组，供得到结果，通过group（N）所获取的到，N从0开始。

下面是详细测试结果，看结果，就明白是否加括号之间的区别了：

```
# here blogContent contains following pic url link:
# http://hiphotos.baidu.com/againinput_tmp/pic/
item/069e0d89033b5bb53d07e9b536d3d539b400bce2.jpg
# http://hiphotos.baidu.com/recommend_music/pic/item/221ebedfa1a34d224954039e.jpg
# following is test result:
pic_pattern_no_parenthesis = r'http://hiphotos.baidu.com/\S+/[ab]{0,2}pic/item/[a-zA-Z0-9]
{24,40}\.\w{3}'
picList_no_parenthesis = re.findall(pic_pattern_no_parenthesis, blogContent) # findall result is
 a list if matched
print 'findall no()=',picList_no_parenthesis
print 'findall no() len=',len(picList_no_parenthesis)
#print 'findall no() group=',picList_no_parenthesis.group(0) # -> cause error
pic_pattern_with_parenthesis = r'http://hiphotos.baidu.com/(\S+)/([ab]{0,2})pic/item/([a-zA-
Z0-9]+)\.([a-zA-Z]{3})'
picList_with_parenthesis = re.findall(pic_pattern_with_parenthesis, blogContent) # findall result
 is a list if matched
print 'findall with()=',picList_with_parenthesis
print 'findall with() len=',len(picList_with_parenthesis)
#print 'findall with() group(0)=',picList_with_parenthesis.group(0) # -> cause error
#print 'findall with() group(1)=',picList_with_parenthesis.group(1) # -> cause error
print 'findall with() [0][0]=',picList_with_parenthesis[0][0]
print 'findall with() [0][1]=',picList_with_parenthesis[0][1]
print 'findall with() [0][2]=',picList_with_parenthesis[0][2]
print 'findall with() [0][3]=',picList_with_parenthesis[0][3]
#print 'findall with() [0][4]=',picList_with_parenthesis[0][4] # no [4] -> cause error
```

测试结果为：
findall no()= [u'http://hiphotos.baidu.com/againinput_tmp/pic/item/069e0d89033b5bb53d07e9b536d3d539b400bce2.jpg', u'http://hiphotos.baidu.com/recommend_music/pic/item/221ebedfa1a34d224954039e.jpg'] findall no() len= 2 findall with()= [(u'againinput_tmp', u'', u'069e0d89033b5bb53d07e9b536d3d539b400bce2', u'jpg'), (u'recommend_music', u'', u'221ebedfa1a34d224954039e', u'jpg')] findall with() len= 2 findall with() [0][0]= againinput_tmp findall with() [0][1]= findall with() [0][2]= 069e0d89033b5bb53d07e9b536d3d539b400bce2 findall with() [0][3]= jpg

6.4. 使用re.search需要注意的事情

```
pattern = re.compile(r'HTTP Error ([0-9]{3}):.*')
matched = re.search(pattern, errStr)
if matched : #注意，此处运行时候会直接出错！！！因为search查找后，应该用
matched.group(0),matched.group(1)等方式查看查找出来的结果
    print 'is http type error'
    isHttpError = True
else :
    print 'not http type error'
    isHttpError = False
```

用re.search后，想要查看结果，如果直接用返回值matched的话，运行的时候会直接出错！！！ 因为search查找后，应该用matched.group(0),matched.group(1)等方式查看查找出来的结果。 这点，需要特别注意。

【后记】

后来的测试结果表明上面的判断是错误的。

上面的错误实际上是由于当时search的时候所传入的参数errStr实际上是个对象类型，而不是普通的str或者unicode字符类型，所以导致上面的search会直接运行出错。

而如果在search之前，用errStr = str(errStr)后，search的结果，则是可以直接拿来判断是否为空，或者用来打印的。

相应的打印出来的结果，是类似这样的：

```
matched= <_sre.SRE_Match object at 0x02B4F1E0>
```

而对应的，matched.group(0)是对应的匹配此次查找的全部的字符：

```
HTTP Error 500: ( The specified network name is no longer available. )
```

【总结】

在调用类似于re.search等函数的时候，要确保传入的所要查找的变量，是字符类型（str或者是unicode），否则，像我这里，传入的是一个对象，而不是字符，就会导致运行出错了。

6.5. Python正则表达式的一些疑惑和未解决的问题

6.5.1. 搜索内容包含斜杠时，必须加上反斜杠才可以搜索到，原因未知

字符串变量respPostJson为：

```
,url : 'http:\/\/hi.baidu.com\/shuisidezhuyi\/item\/d32cc02e598460c50e37f967',
```

使用代码：

```
foundUrlList = re.findall("url\s*?:\s*?'(?P<url>http:\\/\\/hi\.baidu\.com\\/.+?\\/item\\/\w+?)'",
 respPostJson);
logging.info("foundUrlList=%s", foundUrlList);
```

却搜不到对应的字符串，结果为：

```
foundUrlList=[]
```

而只有给斜杠前面加上反斜杠：

```
foundUrlList = re.findall("url\s*?:\s*?'(?P<url>http:\\\/\\\/hi\.baidu\.com\\\/.+?\\\/item\\\/\w
+?)'", respPostJson);
logging.info("foundUrlList=%s", foundUrlList);
```

才可以搜索到结果：

```
foundUrlList=['http:\\/\\/hi.baidu.com\\/shuisidezhuyi\\/item\\/d32cc02e598460c50e37f967']
```

很是奇怪。目前不知道为何会这样，等待高手给解释解释。

参考书目

[1] 【总结】关于（C#和Python中的）正则表达式[1]

[2] perl regex: m//[2]

[3] perl regex: s///[3]

[4] perl regex: qr/STRING/[4]

[5] Perl Regexp-Quote-Like-Operators[5]

[6] [issue14258] Better explain re.LOCALE and re.UNICODE for \S and \W[6]

[7] Regular Expression Options[7]

[8] 【已解决】Perl中的正则表达式的替换和后向引用[8]

[9] ActionScript[9]

[1] http://www.crifan.com/summary_regular_expression_csharp_python/
[2] http://perldoc.perl.org/functions/m.html
[3] http://perldoc.perl.org/functions/s.html
[4] http://perldoc.perl.org/functions/qr.html
[5] http://perldoc.perl.org/perlop.html#Regexp-Quote-Like-Operators
[6] http://python.6.x6.nabble.com/issue14258-Better-explain-re-LOCALE-and-re-UNICODE-for-S-and-W-td4568904.html
[7] http://msdn.microsoft.com/en-us/library/yd1hzczs%28v=vs.71%29.aspx
[8] http://www.crifan.com/perl_regex_replace_backreference/
[9] http://zh.wikipedia.org/wiki/ActionScript