Python3 正则表达式

2017年4月13日 11:20

正则表达式是一个特殊的字符序列，它能帮助你方便的检查一个字符串是否与某种模式匹配。
Python 自1.5版本起增加了re 模块，它提供 Perl 风格的正则表达式模式。
re 模块使 Python 语言拥有全部的正则表达式功能。
compile 函数根据一个模式字符串和可选的标志参数生成一个正则表达式对象。该对象拥有一系列方法用于正则表达式匹配和替换。
re 模块也提供了与这些方法功能完全一致的函数，这些函数使用一个模式字符串做为它们的第一个参数。
本章节主要介绍Python中常用的正则表达式处理函数。

re.match函数

re.match 尝试从字符串的起始位置匹配一个模式，如果不是起始位置匹配成功的话，match()就返回none。

函数语法:

```
re.match(pattern, string, flags=0)
```

函数参数说明:

参数	描述
pattern	匹配的正则表达式
string	要匹配的字符串。
flags	标志位，用于控制正则表达式的匹配方式，如：是否区分大小写，多行匹配等等。

匹配成功re.match方法返回一个匹配的对象，否则返回None。

我们可以使用group(num) 或 groups() 匹配对象函数来获取匹配表达式。

匹配对象方法	描述
group(num=0)	匹配的整个表达式的字符串，group() 可以一次输入多个组号，在这种情况下它将返回一个包含那些组所对应值的元组。
groups()	返回一个包含所有小组字符串的元组，从 1 到 所含的小组号。

实例 1:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import re
print(re.match('www', 'www.runoob.com').span())  # 在起始位置匹配
print(re.match('com', 'www.runoob.com'))         # 不在起始位置匹配
```

以上实例运行输出结果为:

```
(0, 3)
None
```

实例 2:

```
#!/usr/bin/python3
import re

line = "Cats are smarter than dogs"

matchObj = re.match( 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!!")
```

以上实例执行结果如下：

```
matchObj.group() :  Cats are smarter than dogs
matchObj.group(1) :  Cats
matchObj.group(2) :  smarter
```

re.search方法

re.search 扫描整个字符串并返回第一个成功的匹配。

函数语法：

```
re.search(pattern, string, flags=0)
```

函数参数说明：

参数	描述
pattern	匹配的正则表达式
string	要匹配的字符串。
flags	标志位，用于控制正则表达式的匹配方式，如：是否区分大小写，多行匹配等等。

匹配成功re.search方法返回一个匹配的对象，否则返回None。

我们可以使用group(num) 或 groups() 匹配对象函数来获取匹配表达式。

匹配对象方法	描述
group(num=0)	匹配的整个表达式的字符串，group() 可以一次输入多个组号，在这种情况下它将返回一个包含那些组所对应值的元组。
groups()	返回一个包含所有小组字符串的元组，从 1 到 所含的小组号。

实例 1：

```
#!/usr/bin/python3

import re

print(re.search('www', 'www.runoob.com').span())  # 在起始位置匹配
print(re.search('com', 'www.runoob.com').span())         # 不在起始位置匹配
```

以上实例运行输出结果为：

```
(0, 3)
(11, 14)
```

实例 2：

```
#!/usr/bin/python3

import re

line = "Cats are smarter than dogs";

searchObj = re.search( r'(.*) are (.*?) .*', line, re.M|re.I)

if searchObj:
   print ("searchObj.group() : ", searchObj.group())
   print ("searchObj.group(1) : ", searchObj.group(1))
   print ("searchObj.group(2) : ", searchObj.group(2))
```

```
else:
   print ("Nothing found!!")
```

以上实例执行结果如下：

```
searchObj.group() :  Cats are smarter than dogs
searchObj.group(1) :  Cats
searchObj.group(2) :  smarter
```

re.match与re.search的区别

re.match只匹配字符串的开始，如果字符串开始不符合正则表达式，则匹配失败，函数返回None；而re.search匹配整个字符串，直到找到一个匹配。

实例：

```
#!/usr/bin/python3

import re

line = "Cats are smarter than dogs";

matchObj = re.match( r'dogs', line, re.M|re.I)
if matchObj:
   print ("match --> matchObj.group() : ", matchObj.group())
else:
   print ("No match!!")

matchObj = re.search( r'dogs', line, re.M|re.I)
if matchObj:
   print ("search --> matchObj.group() : ", matchObj.group())
else:
   print ("No match!!")
```

以上实例运行结果如下：

```
No match!!
search --> matchObj.group() :  dogs
```

检索和替换

Python 的re模块提供了re.sub用于替换字符串中的匹配项。

语法：

```
re.sub(pattern, repl, string, count=0)
```

参数：

- pattern : 正则中的模式字符串。
- repl : 替换的字符串，也可为一个函数。
- string : 要被查找替换的原始字符串。
- count : 模式匹配后替换的最大次数，默认 0 表示替换所有的匹配。

实例：

```
#!/usr/bin/python3
import re

phone = "2004-959-559 # 这是一个电话号码"

# 删除注释
num = re.sub(r'#.*$', "", phone)
print ("电话号码 : ", num)

# 移除非数字的内容
```

```
num = re.sub(r'\D', "", phone)
print ("电话号码 : ", num)
```

以上实例执行结果如下:

```
电话号码 :  2004-959-559
电话号码 :  2004959559
```

repl 参数是一个函数

以下实例中将字符串中的匹配的数字乘于 2:

```
#!/usr/bin/python

import re

# 将匹配的数字乘于 2
def double(matched):
    value = int(matched.group('value'))
    return str(value * 2)

s = 'A23G4HFD567'
print(re.sub('(?P<value>\d+)', double, s))
```

执行输出结果为:

```
A46G8HFD1134
```

正则表达式修饰符 - 可选标志

正则表达式可以包含一些可选标志修饰符来控制匹配的模式。修饰符被指定为一个可选的标志。多个标志可以通过按位 OR(|) 它们来指定。如 re.I | re.M 被设置成 I 和 M 标志:

修饰符	描述
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。这个标志影响 \w, \W, \b, \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得更易于理解。

正则表达式模式

模式字符串使用特殊的语法来表示一个正则表达式:

字母和数字表示他们自身。一个正则表达式模式中的字母和数字匹配同样的字符串。

多数字母和数字前加一个反斜杠时会拥有不同的含义。

标点符号只有被转义时才匹配自身，否则它们表示特殊的含义。

反斜杠本身需要使用反斜杠转义。

由于正则表达式通常都包含反斜杠，所以你最好使用原始字符串来表示它们。模式元素(如 r'/t'，等价于'//t')匹配相应的特殊字符。

下表列出了正则表达式模式语法中的特殊元素。如果你使用模式的同时提供了可选的标志参数，某些模式元素的含义会改变。

模式	描述
^	匹配字符串的开头
$	匹配字符串的末尾。
.	匹配任意字符，除了换行符，当re.DOTALL标记被指定时，则可以匹配包括换行符的任意字

	符。
[...]	用来表示一组字符,单独列出：[amk] 匹配 'a'，'m'或'k'
[^...]	不在[]中的字符：[^abc] 匹配除了a,b,c之外的字符。
re*	匹配0个或多个的表达式。
re+	匹配1个或多个的表达式。
re?	匹配0个或1个由前面的正则表达式定义的片段，非贪婪方式
re{ n}	
re{ n,}	精确匹配n个前面表达式。
re{ n, m}	匹配 n 到 m 次由前面的正则表达式定义的片段，贪婪方式
a| b	匹配a或b
(re)	G匹配括号内的表达式，也表示一个组
(?imx)	正则表达式包含三种可选标志：i, m, 或 x 。只影响括号中的区域。
(?-imx)	正则表达式关闭 i, m, 或 x 可选标志。只影响括号中的区域。
(?: re)	类似 (...), 但是不表示一个组
(?imx: re)	在括号中使用i, m, 或 x 可选标志
(?-imx: re)	在括号中不使用i, m, 或 x 可选标志
(?#...)	注释.
(?= re)	前向肯定界定符。如果所含正则表达式，以 ... 表示，在当前位置成功匹配时成功，否则失败。但一旦所含表达式已经尝试，匹配引擎根本没有提高；模式的剩余部分还要尝试界定符的右边。
(?! re)	前向否定界定符。与肯定界定符相反；当所含表达式不能在字符串当前位置匹配时成功
(?> re)	匹配的独立模式，省去回溯。
\w	匹配字母数字
\W	匹配非字母数字
\s	匹配任意空白字符，等价于 [\t\n\r\f].
\S	匹配任意非空字符
\d	匹配任意数字，等价于 [0-9].
\D	匹配任意非数字
\A	匹配字符串开始
\Z	匹配字符串结束，如果是存在换行，只匹配到换行前的结束字符串。c
\z	匹配字符串结束
\G	匹配最后匹配完成的位置。
\b	匹配一个单词边界，也就是指单词和空格间的位置。例如， 'er\b' 可以匹配"never" 中的 'er'，但不能匹配 "verb" 中的 'er'。
\B	匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'，但不能匹配 "never" 中的 'er'。
\n, \t, 等.	匹配一个换行符。匹配一个制表符。等
\1...\9	匹配第n个分组的子表达式。
\10	匹配第n个分组的子表达式，如果它经匹配。否则指的是八进制字符码的表达式。

正则表达式实例

字符匹配

实例	描述
python	匹配 "python".

字符类

实例	描述
[Pp]ython	匹配 "Python" 或 "python"
rub[ye]	匹配 "ruby" 或 "rube"
[aeiou]	匹配中括号内的任意一个字母
[0-9]	匹配任何数字。类似于 [0123456789]
[a-z]	匹配任何小写字母
[A-Z]	匹配任何大写字母
[a-zA-Z0-9]	匹配任何字母及数字
[^aeiou]	除了aeiou字母以外的所有字符
[^0-9]	匹配除了数字外的字符

特殊字符类

实例	描述
.	匹配除 "\n" 之外的任何单个字符。要匹配包括 '\n' 在内的任何字符，请使用象 '[.\n]' 的模式。
\d	匹配一个数字字符。等价于 [0-9]。
\D	匹配一个非数字字符。等价于 [^0-9]。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
\w	匹配包括下划线的任何单词字符。等价于'[A-Za-z0-9_]'。
\W	匹配任何非单词字符。等价于 '[^A-Za-z0-9_]'。