【整理】详解Python中re.sub

[2013 年 5 月 2 日 下午 6:29](http://www.crifan.com/python_re_sub_detailed_introduction/)[crifan](http://www.crifan.com/author/crifan/)[已有62963人围观](http://www.crifan.com/python_re_sub_detailed_introduction/)[5个评论](http://www.crifan.com/python_re_sub_detailed_introduction/#comments)

待完成，最近更新：2013-05-08

【背景】

Python中的正则表达式方面的功能，很强大。

其中就包括re.sub，实现正则的替换。

功能很强大，所以导致用法稍微有点复杂。

所以当遇到稍微复杂的用法时候，就容易犯错。

所以此处，总结一下，在使用re.sub的时候，需要注意的一些事情。

解释具体的注意事项之前，先把其具体的解释贴出来：

[re.sub](http://docs.python.org/2/library/re.html#re.sub)

|  |
| --- |
| **re.sub(*pattern*, *repl*, *string*, *count=0*, *flags=0*)**  Return the string obtained by replacing the leftmost non-overlapping occurrences of *pattern* in *string* by the replacement *repl*. If the pattern isn’t found, *string* is returned unchanged. *repl* can be a string or a function; if it is a string, any backslash escapes in it are processed. That is, \n is converted to a single newline character, \r is converted to a carriage return, and so forth. Unknown escapes such as \j are left alone. Backreferences, such as \6, are replaced with the substring matched by group 6 in the pattern. For example:  >>> re.sub(r'def\s+([a-zA-Z\_][a-zA-Z\_0-9]\*)\s\*\(\s\*\):',  ... r'static PyObject\*\npy\_\1(void)\n{',  ... 'def myfunc():')  'static PyObject\*\npy\_myfunc(void)\n{'  If *repl* is a function, it is called for every non-overlapping occurrence of *pattern*. The function takes a single match object argument, and returns the replacement string. For example:  >>> def dashrepl(matchobj):  ... if matchobj.group(0) == '-': return ' '  ... else: return '-'  >>> re.sub('-{1,2}', dashrepl, 'pro----gram-files')  'pro--gram files'  >>> re.sub(r'\sAND\s', ' & ', 'Baked Beans And Spam', flags=re.IGNORECASE)  'Baked Beans & Spam'  The pattern may be a string or an RE object.  The optional argument *count* is the maximum number of pattern occurrences to be replaced; *count* must be a non-negative integer. If omitted or zero, all occurrences will be replaced. Empty matches for the pattern are replaced only when not adjacent to a previous match, so sub('x\*', '-', 'abc') returns '-a-b-c-'.  In addition to character escapes and backreferences as described above, \g<name> will use the substring matched by the group named name, as defined by the (?P<name>...) syntax. \g<number>uses the corresponding group number; \g<2> is therefore equivalent to \2, but isn’t ambiguous in a replacement such as \g<2>0. \20 would be interpreted as a reference to group 20, not a reference to group 2 followed by the literal character '0'. The backreference \g<0> substitutes in the entire substring matched by the RE.  Changed in version 2.7: Added the optional flags argument. |

re.sub的功能

re是regular expression的所写，表示正则表达式

sub是substitute的所写，表示替换；

re.sub是个正则表达式方面的函数，用来实现通过正则表达式，实现比普通字符串的replace更加强大的替换功能；

举个最简单的例子：

如果输入字符串是：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1 | inputStr = "hello 111 world 111" |

那么你可以通过

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1 | replacedStr = inputStr.replace("111", "222") |

去换成

|  |
| --- |
| "hello 222 world 222" |

但是，如果输入字符串是：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1 | inputStr = "hello 123 world 456" |

而你是想把123和456，都换成222

（以及其他还有更多的复杂的情况的时候），

那么就没法直接通过字符串的replace达到这一目的了。

就需要借助于re.sub，通过正则表达式，来实现这种相对复杂的字符串的替换：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1 | replacedStr = re.sub("\d+", "222", inputStr) |

当然，实际情况中，会有比这个例子更加复杂的，其他各种特殊情况，就只能通过此re.sub去实现如此复杂的替换的功能了。

所以，re.sub的含义，作用，功能就是：

**对于输入的一个字符串，利用正则表达式（的强大的字符串处理功能），去实现（相对复杂的）字符串替换处理，然后返回被替换后的字符串**

其中re.sub还支持各种参数，比如count指定要替换的个数等等。

下面就是来详细解释其各个参数的含义。

re.sub的各个参数的详细解释

re.sub共有五个参数。

其中三个必选参数：*pattern*, *repl*, *string*

两个可选参数：*count*, *flags*

第一个参数：pattern

pattern，表示正则中的模式字符串，这个没太多要解释的。

需要知道的是：

* 反斜杠加数字（\N），则对应着匹配的组（matched group）
  + 比如\6，表示匹配前面pattern中的第6个group
  + 意味着，pattern中，前面肯定是存在对应的，第6个group，然后你后面也才能去引用

比如，想要处理：

|  |
| --- |
| hello crifan, nihao crifan |

且此处的，前后的crifan，肯定是一样的。

而想要把整个这样的字符串，换成crifanli

则就可以这样的re.sub实现替换：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1  2  3 | inputStr = "hello crifan, nihao crifan";  replacedStr = re.sub(r"hello (\w+), nihao \1", "crifanli", inputStr);  print "replacedStr=",replacedStr; #crifanli |

第二个参数：repl

repl，就是replacement，被替换，的字符串的意思。

repl可以是字符串，也可以是函数。

**repl是字符串**

如果repl是字符串的话，其中的任何反斜杠转义字符，都会被处理的。

即：

* \n：会被处理为对应的换行符；
* \r：会被处理为回车符；
* 其他不能识别的转移字符，则只是被识别为普通的字符：
  + 比如\j，会被处理为j这个字母本身；
* 反斜杠加g以及中括号内一个名字，即：\g<name>，对应着命了名的组，named group

接着上面的举例：

想要把对应的：

|  |
| --- |
| hello crifan, nihao crifan |

中的crifan提取出来，只剩：

|  |
| --- |
| crifan |

就可以写成：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1  2  3 | inputStr = "hello crifan, nihao crifan";  replacedStr = re.sub(r"hello (\w+), nihao \1", "\g<1>", inputStr);  print "replacedStr=",replacedStr; #crifan |

对应的带命名的组（named group）的版本是：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1  2  3 | inputStr = "hello crifan, nihao crifan";  replacedStr = re.sub(r"hello (?P<name>\w+), nihao (?P=name)", "\g<name>", inputStr);  print "replacedStr=",replacedStr; #crifan |

**repl是函数**

举例说明：

比如输入内容是：

|  |
| --- |
| hello 123 world 456 |

想要把其中的数字部分，都加上111，变成：

|  |
| --- |
| hello 234 world 567 |

那么就可以写成：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | #!/usr/bin/python  # -\*- coding: utf-8 -\*-  """  Function:  【整理】详解Python中re.sub  <http://www.crifan.com/python_re_sub_detailed_introduction>    Version:    2013-05-02  Author:     Crifan  Contact:    admin (at) crifan.com  """    import re;    def pythonReSubDemo():      """          demo Pyton re.sub      """      inputStr = "hello 123 world 456";        def \_add111(matched):          intStr = matched.group("number"); #123          intValue = int(intStr);          addedValue = intValue + 111; #234          addedValueStr = str(addedValue);          return addedValueStr;        replacedStr = re.sub("(?P<number>\d+)", \_add111, inputStr);      print "replacedStr=",replacedStr; #hello 234 world 567    ###############################################################################  if \_\_name\_\_=="\_\_main\_\_":      pythonReSubDemo(); |

第三个参数：string

string，即表示要被处理，要被替换的那个string字符串。

没什么特殊要说明。

第四个参数：count

举例说明：

继续之前的例子，假如对于匹配到的内容，只处理其中一部分。

比如对于：

|  |
| --- |
| hello 123 world 456 nihao 789 |

只是像要处理前面两个数字：123,456，分别给他们加111，而不处理789，

那么就可以写成：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | #!/usr/bin/python  # -\*- coding: utf-8 -\*-  """  Function:  【整理】详解Python中re.sub  <http://www.crifan.com/python_re_sub_detailed_introduction>    Version:    2013-05-02  Author:     Crifan  Contact:    admin (at) crifan.com  """    import re;    def pythonReSubDemo():      """          demo Pyton re.sub      """      inputStr = "hello 123 world 456 nihao 789";        def \_add111(matched):          intStr = matched.group("number"); #123          intValue = int(intStr);          addedValue = intValue + 111; #234          addedValueStr = str(addedValue);          return addedValueStr;        replacedStr = re.sub("(?P<number>\d+)", \_add111, inputStr, 2);      print "replacedStr=",replacedStr; #hello 234 world 567 nihao 789    ###############################################################################  if \_\_name\_\_=="\_\_main\_\_":      pythonReSubDemo(); |

第五个参数：flags

关于re.sub的注意事项

然后再来整理一些，关于re.sub的注意事项，常见的问题及解决办法：

要注意，被替换的字符串，即参数repl，是普通的字符串，不是pattern

注意到，语法是：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1 | re.sub(pattern, repl, string, count=0, flags=0) |

即，对应的第二个参数是repl。

需要你指定对应的r前缀，才是pattern：

r"xxxx"

不要误把第四个参数flag的值，传递到第三个参数count中了

否则就会出现我这里：

[【已解决】Python中，（1）re.compile后再sub可以工作，但re.sub不工作，或者是（2）re.search后replace工作，但直接re.sub以及re.compile后再re.sub都不工作](http://www.crifan.com/python_re_complie_then_sub_work_but_re_sub_not_work/)

遇到的问题：

当传递第三个参数，原以为是flag的值是，

结果实际上是count的值

所以导致re.sub不功能，

所以要参数指定清楚了：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1 | replacedStr = re.sub(replacePattern, orignialStr, replacedPartStr, flags=re.I); # can omit count parameter |

或：

[?](http://www.crifan.com/python_re_sub_detailed_introduction/)

|  |  |
| --- | --- |
| 1 | replacedStr = re.sub(replacePattern, orignialStr, replacedPartStr, 1, re.I); # must designate count parameter |

才可以正常工作。