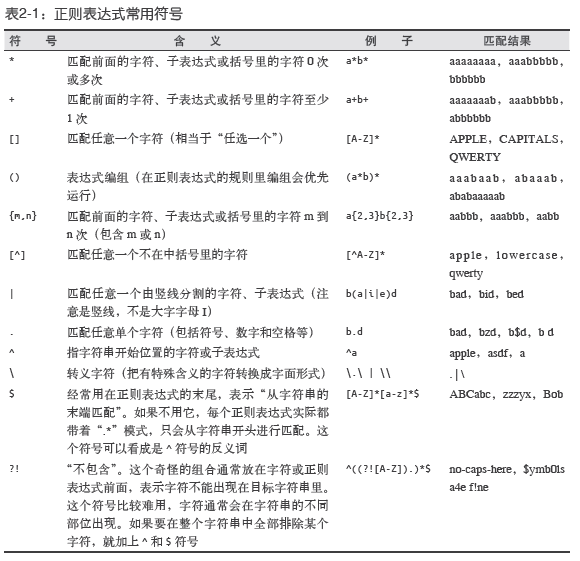
《Python 网络数据采集》 第二章

正则表达式



<http://www.cnblogs.com/vamei/archive/2012/08/31/2661870.html>

正则表达式的函数

m = re.search(pattern, string) # 搜索整个字符串，直到发现符合的子字符串。 m = re.match(pattern, string) # 从头开始检查字符串是否符合正则表达式。必须从字符串的第一个字符开始就相符。

可以从这两个函数中选择一个进行搜索。上面的例子中，我们如果使用re.match()的话，则会得到None，因为字符串的起始为‘a’， 不符合'[0-9]'的要求。

对于返回的m, 我们使用m.group()来调用结果。（我们会在后面更详细解释m.group()）

我们还可以在搜索之后将搜索到的子字符串进行替换：

str = re.sub(pattern, replacement, string)

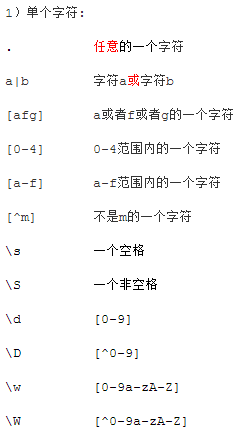
# 在string中利用正则变换pattern进行搜索，对于搜索到的字符串，用另一字符串replacement替换。返回替换后的字符串。

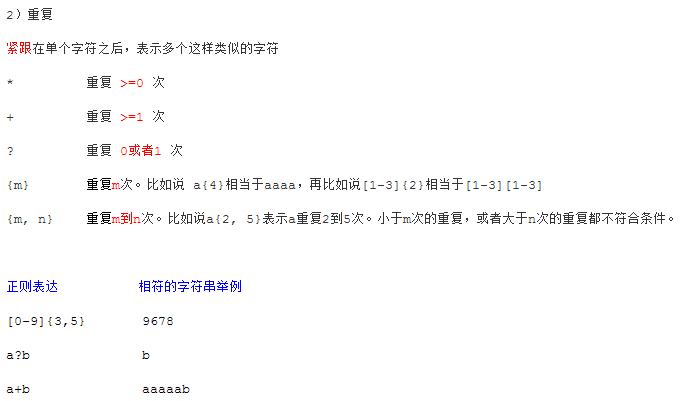
此外，常用的正则表达式函数还有

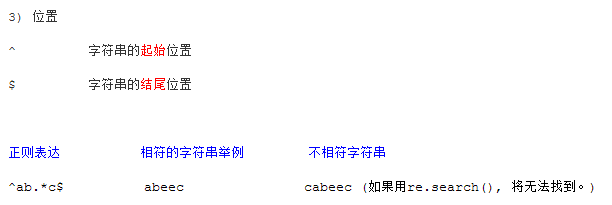
re.split()    # 根据正则表达式分割字符串， 将分割后的所有子字符串放在一个表(list)中返回

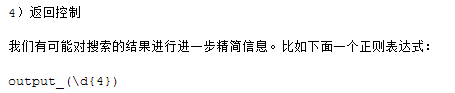
re.findall()  # 根据正则表达式搜索字符串，将所有符合的子字符串放在一给表(list)中返回

(在熟悉了上面的函数后，可以看一下re.compile()，以便于提高搜索效率。)









该正则表达式用括号()包围了一个小的正则表达式，\d{4}。 这个小的正则表达式被用于从结果中筛选想要的信息（在这里是四位数字）。这样被括号圈起来的正则表达式的一部分，称为群(group)。

我们可以m.group(number)的方法来查询群。group(0)是整个正则表达的搜索结果，group(1)是第一个群……

import re m = re.search("output\_(\d{4})", "output\_1986.txt")print(m.group(1))

我们还可以将群命名，以便更好地使用m.group查询:

import re m = re.search("output\_(?P<year>\d{4})", "output\_1986.txt") #(?P<name>...) 为group命名print(m.group("year"))