layout: post
title: "正则表达式"
date: 2020-07-01
description: "python常用技巧"

tag: python常用技巧

在编程中，经常会涉及到字符串的操作，一个常用的策略就是利用`split`函数，然后对于特定的字符串进行匹配，但是这种方法格式复杂，可复用性较差。

正则表达式是处理字符串匹配一个必不可少的方法，定义一个语义规则，来进行特定的字符字符串的规则。

[更多技术文章请点击查看]

正则表达式速查表

1 元字符，用特殊符号表示一类元素

语法	说明	语法示例	匹配字符
.	匹配除换行符以外的任意字符（这是一个小红点儿）	a.c	abc
\w	匹配字母或数字或下划线 (可以记成 word)	a\wc	a2c
\b	匹配一个单词的边界	foo\b	匹配foo,foo. 但不匹配foobar
\B	匹配空字符串，但 不 能在词的开头或者结尾	py\B	匹配python py2 py3, 但不匹配py.
^	匹配字符串的开始	^abc	abc
\W	匹配非字母数字下划线	a\Wb	a b
\d	匹配数字 (可记成 digit)	a\dc	a2c
\D	匹配非数字	a\Dc	abc
\s	匹配任意的空白符 (可以记成 space)	a\sb	a b
\S	匹配非空白符	a\Sb	acb
a|b	匹配字符a或字符b。（长的写前面，短的写后面）	abc|def	abc 或def
()	匹配括号内的表达式，也表示一个组	(abc)	abc
[...]	匹配字符组中的字符	a[bcd]e	abe 或ace或ade
[^...]	匹配除了字符组中字符的所有字符	a[bcd]e	ase

2 字符组 []，限制范围，某个位置只能出现这个范围内的某个元素。

语法	说明	语法示例	匹配字符
[0123456789]	只能匹配0-9数字，可写成[0-9]	[0123456789]	0或1或2...
[a-z]	只能匹配小写字母	[a-z]	d
[A-Z]	只能匹配大写字母	[A-Z]	D
[0-9a-fA-F]	可以匹配16进制的某个数	[0-9a-fA-F]	D

3 量词，表示数量，约束前面元字符出现的次数。

语法	说明	语法示例	匹配字符
*	重复前一个字符零次或多次	abc*	ab 或abccc
+	重复前一个字符一次或多次	abc+	abc 或abccc
?	重复前一个字符零次或一次	abc?	ab 或abc
{n}	重复前一个字符n次	abc{3}	abccc
{n,}	重复前一个字符至少n次	abc{3,}	abccccc
{n,m}	重复前一个字符n到m次	abc{1,5}	abc或abcc或abccccc

re模块包函数

```
# 导入包
import re
```

```
#################re.compile() 编译
# 将正则表达式编译成一个正则表表达式对象。
# 如果一个正则表达式在程序中只用一次，就没必要编译了。
# 如果同一个正则表达式要被多次使用时，就需要对表达式进行编译，以便后续使用。
例如：匹配www.baidu.com
pattern = r'w*\.[a-z]{5}\.[a-z]*'
prog = re.compile(pattern)
string = 'www.baidu.com'
result = prog.match(string)
# 如果匹配成功返回match对象，不成功返回None
```

```
###################re.findall()  返回列表

# 一、re.findall(正则表达式，待匹配的字符串，flags=0)

# 二、findall默认只显示分组中的，分组有优先级。
pattern = r'w{3}\.(baidu|oldboy)\.com'
prog = re.compile(pattern)
ret = re.findall(prog, 'www.baidu.com')
```

```
print(ret)
# 结果：['baidu']

# 三、加上 ?: 取消分组的优先
# ret = re.findall('www\.(?:baidu|oldboy)\.com', 'www.baidu.com')
# print(ret)
# 结果：['www.baidu.com']
```

```
####################re.match()
#从头匹配
re.match(pattern, string, flags=0)
#如果 string 开始的0或者多个字符匹配到了正则表达式样式，就返回一个相应的 匹配对象 。 如果没有匹配，就返回 None ；
#需要group(),才能返回值
```

```
####################re.search()
#扫描整个 字符串 找到匹配样式的第一个位置，并返回一个相应的 匹配对象。如果没有匹配，就返回一个 None ； 注
#同样需要group(),才能的返回值
```

```
####################re.sub()、re.subn()
# re.sub()
re.sub(r'\sAND\s', ' & ', 'Baked Beans And Spam', flags=re.IGNORECASE)  # 【表达式表示替换内容，要替换的新元素，替换对象，替换次数】

# 结果为：'Baked Beans & Spam'
# re.subn()
re.subn(r'\sAND\s', ' & ', 'Baked Beans And Spam', flags=re.IGNORECASE)   # 【返回元组，前面是替换完的结果，后面是替换的次数】

结果是('Baked Beans & Spam', 1)
```