Python 正则表达式高级语法实战

第一部分：Python 正则表达是基本语法

常用的元字符			常用限定符	
代码	说明		代码	说明
.	匹配除换行符以外的任意字符		*	重复零次或更多次
\w	匹配字母或数字或下划线		+	重复一次或更多次
\s	匹配任意的空白符		?	重复零次或一次
\d	匹配数字		{n}	重复 n 次
\b	匹配单词的开始或结束		{n,}	重复 n 次或更多次
^	匹配字符串的开始		{n, m}	重复 n 到 m 次
$	匹配字符串的结束			
(xyz)	匹配小括号内的 xyz(作为一个整体去匹配)			
x|y	|表示或，匹配 x 或 y			
[…]	表示一组字符如：[amk] 匹配’ a’, ’m’或’ k’			
[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			
常用反义词				
代码	说明			
\W	匹配任意不是字母，数字，下划线，汉字的字符			
\S	匹配任意不是空白符的字符			
\D	匹配任意非数字的字符			
\B	匹配不是单词开头或结束的位置			
[^x]	匹配除了 x 以外的任意字符			
[^aeiou]	匹配除了 aeiou 这几个字母以外的任意字符			
[^0-9]	匹配除了数字外的字符			

第二部分：Python 正则表达式高级语法

复杂组装的正则语法	
.+?	表示匹配任意字符一个或则多个
.*?	表示匹配任意字符 0 个或多个
[^,，；;：:。？！?!]*?	表示匹配中括弧中 0 个或多个
.{0,3}	表示匹配 0 个到 3 个任意字符
[0-9.]+	表示匹配至少一个数字和点号

第三部分：常用的案例

案例 1：贪婪和非贪婪模式匹配(.+?)	
匹配的正则	r"a(.+?)b"
输入数据	"a123b456b"
输出数据	['123']
案例点评	**点评 1**：?控制只匹配 0 或 1 个,所以只会输出和最近的 b 之间的匹配情况 **点评 2**：一个字符串”a123b456b”，如果我们想匹配 a 和最后一个 b 之间的所有值而非 a 和第一个出现的 b 之间的值,可以用?来控制正则贪婪和非贪婪匹配

案例 2：贪婪和非贪婪模式匹配(.+)	
匹配的正则	r"a(.+)b"
输入数据	"a123b456b"
输出数据	['123b456']
案例点评	**点评 1**：一个字符串”a123b456b”，如果我们想匹配 a 和最后一个 b 之间的所有值而非 a 和第一个出现的 b 之间的值,可以用?来控制正则贪婪和非贪婪匹配

案例 3：贪婪和非贪婪模式匹配(.*)	
匹配的正则	r"a(.*)b"
输入数据	"a123b456b"
输出数据	['123b456']
案例点评	**点评 1**：一个字符串”a123b456b”，如果我们想匹配 a 和最后一个 b 之间的所有值而非 a 和第一个出现的 b 之间的值,可以用?来控制正则贪婪和非贪婪匹配

案例 4：贪婪和非贪婪模式匹配(.*?)	
匹配的正则	r"a(.*?)b"
输入数据	"a123b456b"
输出数据	['123']
案例点评	**点评 1**：一个字符串”a123b456b”，如果我们想匹配 a 和最后一个 b 之间的所有值而非 a 和第一个出现的 b 之间的值,可以用?来控制正则贪婪和非贪婪匹配

案例 5：贪婪和非贪婪模式匹配(含?号)	
匹配的正则	r'.*?(增加).{0,3}?([0-9.点,]+)(%|成|倍).*?'
输入数据	增加 1,234 倍
输出数据	[('增加', '1,234', '倍')]
案例点评	

案例 6：贪婪和非贪婪模式匹配(不含?号)

匹配的正则	r'.*?(增加).{0,3}([0-9.点,]+)(%|成|倍).*?'
输入数据	增加 1,234 倍
输出数据	[('增加', '34', '倍')]
案例点评	

案例 7：多行匹配	
匹配的正则	re.findall(r"a(\d+)b.+a(\d+)b", str)
输入数据	"a23b\na34b"
输出数据	[]
案例点评	**点评 1：**如果你要多行匹配，那么需要加上 re.S 和 re.M 标志. 加上 re.S 后, . 将会匹配换行符，默认.不会匹配换行符

案例 8：多行匹配	
匹配的正则	re.findall(r"a(\d+)b.+a(\d+)b", str, re.S)
输入数据	"a23b\na34b"
输出数据	[('23', '34')]
案例点评	**点评 1：**如果你要多行匹配，那么需要加上 re.S 和 re.M 标志. 加上 re.S 后, . 将会匹配换行符，默认.不会匹配换行符

案例 9：多行匹配(匹配字符串的开始或者结束)	
匹配的正则	re.findall(r"^a(\d+)b", str)
输入数据	"a23b\na34b"
输出数据	['23']
案例点评	**点评 1：**加上 re.M 后,^$标志将会匹配每一行，默认^和$只会匹配第一行

案例 10：多行匹配(匹配字符串的开始或者结束)	
匹配的正则	re.findall(r"^a(\d+)b", str, re.M)
输入数据	"a23b\na34b"
输出数据	['23', '34']
案例点评	**点评 1：**加上 re.M 后,^$标志将会匹配每一行，默认^和$只会匹配第一行

案例 11：[^abc].*?和[^abc]*?的区别	
输入数据	"dh 中国你好 ah 中国你好"
匹配的正则	re.findall(r"[^abc].*?中国", str)
输出数据	['dh 中国', '你好 ah 中国']
匹配的正则	re.findall(r"[^abc]*?中国", str)
输出数据	['dh 中国', 'h 中国']
案例点评	**点评 1：**.表示匹配任意字符，[^abc]表示匹配除了 a,b,c 以外的任意一个字符，.*?表达单独匹配一个和多个字符 **点评 2：***是限定词表示 0 次或者多次出现，[^abc]*?表示匹配除 a,b,c 以外的任意一个字符。 **点评 3：**[^;；::。？！?!\s\t].*?(死|亡|死亡)，比如这个正则，除[]中的所有字符的 0 个和多个，之后就自然会把[死]匹配上

正则表达式中 findall()方法详解

第一部分：findall()函数的定义

它在 re.py 中有定义：

```
def findall(pattern, string, flags=0):
    """Return a list of all non-overlapping matches in the string.
    If one or more capturing groups are present in the pattern, return
    a list of groups; this will be a list of tuples if the pattern
    has more than one group.
    Empty matches are included in the result."""
    return _compile(pattern, flags).findall(string)
```

返回 string 中所有与 pattern 匹配的全部字符串,返回形式为数组。

第二部分：代码列举

```
import re
str = 'aabbabaabbaa'
#一个"."就是匹配除 \n (换行符)以外的任意一个字符
print(re.findall(r'a.b',str))#['aab', 'aab']
#*前面的字符出现 0 次或以上
print(re.findall(r'a*b',str))#['aab', 'b', 'ab', 'aab', 'b']
#贪婪，匹配从.*前面为开始到后面为结束的所有内容
print(re.findall(r'a.*b',str))#['aabbabaabb']
#非贪婪，遇到开始和结束就进行截取，因此截取多次符合的结果，中间没有字符也会被截取
print(re.findall(r'a.*?b',str))#['aab', 'ab', 'aab']
#非贪婪，与上面一样，只是与上面的相比多了一个括号，只保留括号的内容
print(re.findall(r'a(.*?)b',str))#['a', '', 'a']

str = '''aabbab
        aabbaa
        bb'''#后面多加了 2 个 b
#没有把最后一个换行的 aab 算进来
print(re.findall(r'a.*?b',str))#['aab', 'ab', 'aab']
#re.S 不会对\n 进行中断
print(re.findall(r'a.*?b',str,re.S))#['aab', 'ab', 'aab', 'aa\n        b']
```

第三部分：re.findall 中正则表达式(.*?)理解

字符串是
str = 'aabbabaabbaa'

3.1、一个 . 就 是匹配除 \n (换行符)以外的任意一个字符

print(re.findall(r'a.b',str))
['aab', 'aab']

3.2、一个*前面的字符出现 0 次或以上

print(re.findall(r'a*b',str))
['aab', 'b', 'ab', 'aab', 'b']

3.3、.* 贪婪，匹配从.*前面为开始到后面为结束的所有内容

print(re.findall(r'a.*b',str))
['aabbabaabb']

3.4、.*? 非贪婪，遇到开始和结束就进行截取，因此截取多次符合的结果，中间没有字符也会被截取

print(re.findall(r'a.*?b',str))
['aab', 'ab', 'aab']

3.5、(.*?) 非贪婪，与上面一样，只是与上面的相比多了一个括号，只保留括号的内容

print(re.findall(r'a(.*?)b',str))
['a', '', 'a']

第四部分：re.findall 中参数 re.S 的意义

4.1、字符串变为(后面多加了 2 个 b)

```
str = '''aabbab
        aabbaa
        bb'''
```

4.2、参数无 re.S，没有把最后一个换行的 aab 算进来

```
print(re.findall(r'a.*?b',str))
['aab', 'ab', 'aab']
```

4.3、参数有 re.S，不会对\n 进行中断

```
print(re.findall(r'a.*?b',str,re.S))
['aab', 'ab', 'aab', 'aa\n        b']
```