python_re模块

Python中re模块主要功能是通过正则表达式是用来匹配处理字符串的

第一步：import re

导入该模块后，就可以使用该模块下的所有方法和属性

1、正则基本概念

常用的元字符	
代码	说明
.	匹配除换行符以外的任意字符
\w	匹配字母或数字或下划线或汉字
\s	匹配任意的空白符
\d	匹配数字
\b	匹配单词的开始或结束
^	匹配字符串的开始
$	匹配字符串的结束

常用的限定符	
代码	说明
*	重复零次或更多次，优先更多
+	重复一次或更多次，优先更多
?	重复零次后一次，优先一次
{n}	重复n次
{n,}	重复n次或更多次
{n,m}	重复n到m次

^元字符　以什么开头

```
import  re

str="匹配规则这个字符串是否匹配"

print(re.findall("^匹配规则",str)) #字符串开始位置与匹配规则符合就匹配且打印匹配内容，否则不匹配，返回值是list



打印内容：['匹配规则']
```

^元字符　如果写到[]字符集里就是反取

```
import  re

str="匹配s规则这s个字符串是否s匹配f规则则re则则则"

print(re.findall("[^a-z]",str)) #反取，匹配出除字母外的字符，返回值是list



D:\study\python\atp\venv\Scripts\python.exe D:/study/python/atp/lib/t.py

['匹', '配', '规', '则', '这', '个', '字', '符', '串', '是', '否', '匹', '配', '规', '则', '则', '则', '则', '则']
```

$元字符　以什么结尾

```
import  re

str="匹配s规则这s个字符串是否s匹配f规则则re则则则"

print(re.findall("则$",str)) #字符串结束位置与则符合就匹配，否则不匹配，返回值是list



D:\study\python\atp\venv\Scripts\python.exe D:/study/python/atp/lib/t.py

['则']
```

*元字符 匹配其前面的一个字符0次或多次

```
import  re

str="匹配s规则这s个字符串是否s匹配f规则则re则则则"

print(re.findall("则*",str)) #星号前面的一个字符可以是0次或多次，返回值是list

print(re.findall("规则*",str)) #星号前面的一个字符可以是0次或多次，返回值是list



D:\study\python\atp\venv\Scripts\python.exe D:/study/python/atp/lib/t.py

['', '', '', '', '则', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '则则', '', '', '则则则', '']

['规则', '规则则']
```

+元字符 匹配其前面的一个字符1次或多次

```
import  re

str="匹配s规则这s个字符串是否s匹配f规则则re则则则"

print(re.findall("则+",str)) #加号前面的一个字符可以是1次或多次，返回值是list

print(re.findall("规则+",str)) #加号配前面的一个字符可以是1次或多次，返回值是list



D:\study\python\atp\venv\Scripts\python.exe D:/study/python/atp/lib/t.py

['则', '则则', '则则则']

['规则', '规则则']
```

? 元字符 匹配其前面的一个字符0次或1次

```
import re

str="匹配s规则这s个字符串是否s匹配f规则则re则则则"

print(re.findall("则?",str)) #问号前面的一个字符可以是0次或1次，返回值是list

print(re.findall("规则?",str)) #问号前面的一个字符可以是0次或1次，返回值是list



D:\study\python\atp\venv\Scripts\python.exe D:/study/python/atp/lib/t.py

['', '', '', '', '则', '', '', '', '', '', '', '', '', '', '', '', '', '', '则', '则', '', '', '则', '则', '则', '']

['规则', '规则']
```

{}元字符,范围

{m}匹配前一个字符m次，{m,n}匹配前一个字符m至n次，若省略n，则匹配m至无限次

{0,}匹配前一个字符0或多次,等同于*元字符
{+,}匹配前一个字符1次或无限次,等同于+元字符
{0,1}匹配前一个字符0次或1次,等同于?元字符

```
import re

str="匹配s规则这s个字符串是否s匹配f规则则re则则则"

print(re.findall("则{2}",str)) #匹配前一个字符2次，返回值是list

print(re.findall("规则{1,2}",str)) #匹配前一个字符1-2次，返回值是list



D:\study\python\atp\venv\Scripts\python.exe D:/study/python/atp/lib/t.py

['则则', '则则']

['规则', '规则则']
```

[]元字符,字符集

需要字符串里完全符合，匹配规则，就匹配，（规则里的 [] 元字符）对应位置是[]里的任意一个字符就匹配

```
import re

str="匹配s规则这s个字符串是否s匹配f规则则re则则则"

print(re.findall("匹配[s,f]规则",str)) #匹配字符后，只有符合[]中任意字符均可，返回值是list



D:\study\python\atp\venv\Scripts\python.exe D:/study/python/atp/lib/t.py

['匹配s规则', '匹配f规则']
```

\d 匹配任何十进制数，它相当于类[0-9]

```
import  re

str="匹配s规则这s个字符串4是否s匹配3f规则则re则则2则"

print(re.findall("\d",str)) #匹配字符串所有的数字，返回值是list



/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 /Users/dongyf/Documents/python/besttest_study/ryg.py

['4', '3', '2']
```

\d+如果需要匹配一位或者多位数的数字时用

```
import  re

str="匹配s规则这s个字符串455是否s匹配3f规则则re则则2则"

print(re.findall("\d+",str)) #匹配字符串中一位或多位数字，返回值是list



/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 /Users/dongyf/Documents/python/besttest_study/ryg.py

['455', '3', '2']
```

\D匹配任何非数字字符，它相当于类[^0-9]

```
import  re

str="匹配s规则这s个字符串455是否s匹配3f规则则re则则2则"

print(re.findall("\D",str)) #匹配字符串中非数字，返回值是list



/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 /Users/dongyf/Documents/python/besttest_study/ryg.py

['匹', '配', 's', '规', '则', '这', 's', '个', '字', '符', '串', '是', '否', 's', '匹', '配', 'f', '规', '则', '则', 'r', 'e',
'则', '则', '则']
```

\s匹配任何空白字符，它相当于类[\t\n\r\f\v]

```
import  re

str="匹配s规则这s个字  符  串  \n \t \f \v455是否s匹配3f规则则re则则2则"

print(re.findall("\s",str)) #匹配字符串空白字符（\t\n\r\f\v），返回值是list



/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 /Users/dongyf/Documents/python/besttest_study/ryg.py

[' ', ' ', ' ', '\n', ' ', '\t', ' ', '\x0c', ' ', '\x0b']
```

\S匹配任何非空白字符，它相当于类[^\t\n\r\f\v]

```
import  re

str="匹配s规则这s个字 符 串 \n \t \f \v455是"

print(re.findall("\S",str)) #匹配字符串非空白字符（\t\n\r\f\v），返回值是list


/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 /Users/dongyf/Documents/python/besttest_study/ryg.py

['匹', '配', 's', '规', '则', '这', 's', '个', '字', '符', '串', '4', '5', '5', '是']
```

\w匹配包括下划线在内任何字母数字汉字字符

```
import  re

str="匹配s规则这s个_字 S符 串-455是"

print(re.findall("\w",str)) #匹配字符串下划线，汉字，字母，数字，返回值是list


/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 /Users/dongyf/Documents/python/besttest_study/ryg.py

['匹', '配', 's', '规', '则', '这', 's', '个', '_', '字', 'S', '符', '串', '4', '5', '5', '是']
```

\W匹配非任何字母数字汉字字符包括下划线在内

```
import  re

str="匹配s规则这s个_字 S符 串-455是"

print(re.findall("\W",str)) #匹配字符串非下划线，汉字，字母，数字，返回值是list


/Library/Frameworks/Python.framework/Versions/3.7/bin/python3.7 /Users/dongyf/Documents/python/besttest_study/ryg.py

[' ', ' ', '-']
```

()元字符，分组

也就是分组匹配，()里面的为一个组也可以理解成一个整体

如果()后面跟的是特殊元字符如 (adc)* 那么*控制的前导字符就是()里的整体内容，不再是前导一个字符

```
import  re

str="a3a3ddd"

print(re.search("(a3)+",str).group()) #匹配一个或多个a3


C:\Users\zhaow\AppData\Local\Programs\Python\Python37\python.exe D:/study/python/test/uu.py

a3a3
```

|元字符，或

|或，或就是前后其中一个符合就匹配

```
import  re

str="a3死a3d有dd"

print(re.findall(r"死|有+",str)) #匹配|前后一个字符均可



C:\Users\zhaow\AppData\Local\Programs\Python\Python37\python.exe D:/study/python/test/uu.py

['死', '有']
```

r原生字符

将在python里有特殊意义的字符如\b，转换成原生字符（就是去除它在python的特殊意义），不然会给正则表达式有冲突，为了避免这种冲突可以在规则前加原始字符r

模块方法：

match()函数（以后常用）

match，从头匹配一个符合规则的字符串，从起始位置开始匹配，匹配成功返回一个对象，未匹配成功返回None
match(pattern, string, flags=0)
# pattern： 正则模型
# string ： 要匹配的字符串
# falgs ： 匹配模式

```
import  re

str="hello egon bcd egon lge egon acd 19"

r=re.match("h\w+",str) #match，从起始位置开始匹配，匹配成功返回一个对象，未匹配成功返回None,非字母，汉字，数字及下划线分割

print(r.group()) # 获取匹配到的所有结果，不管有没有分组将匹配到的全部拿出来

print(r.groups()) # 获取模型中匹配到的分组结果，只拿出匹配到的字符串中分组部分的结果

print(r.groupdict())  # 获取模型中匹配到的分组结果，只拿出匹配到的字符串中分组部分定义了key的组结果



hello

()

{}




r2=re.match("h(\w+)",str) #match，从起始位置开始匹配，匹配成功返回一个对象，未匹配成功返回None

print(r2.group())

print(r2.groups())

print(r2.groupdict())



hello

('ello',)

{}



r3=re.match("(?P<n1>h)(?P<n2>\w+)",str)  #?P<>定义组里匹配内容的key(键)，<>里面写key名称，值就是匹配到的内容

print(r3.group())

print(r3.groups())

print(r3.groupdict())



hello

('h', 'ello')

{'n1': 'h', 'n2': 'ello'}
```

search()函数
search,浏览全部字符串，匹配第一符合规则的字符串，浏览整个字符串去匹配第一个，未匹配成功返回None
search(pattern, string, flags=0)
 # pattern： 正则模型
 # string ： 要匹配的字符串
 # falgs ： 匹配模式

注意：match()函数 与 search()函数基本是一样的功能，不一样的就是match()匹配字符串开始位置的一个符合规则的字符串，search()是在字符串全局匹配第一个合规则的字符串

```
import  re

str="hello egon bcd egon lge egon acd 19"

r=re.search("h\w+",str) #match，从起始位置开始匹配，匹配成功返回一个对象，未匹配成功返回None,非字母，汉字，数字及下划线分割

print(r.group()) # 获取匹配到的所有结果，不管有没有分组将匹配到的全部拿出来

print(r.groups()) # 获取模型中匹配到的分组结果，只拿出匹配到的字符串中分组部分的结果

print(r.groupdict())  # 获取模型中匹配到的分组结果，只拿出匹配到的字符串中分组部分定义了key的组结果



hello

()

{}




r2=re.search("h(\w+)",str) #match，从起始位置开始匹配，匹配成功返回一个对象，未匹配成功返回None

print(r2.group())

print(r2.groups())

print(r2.groupdict())


hello

('ello',)

{}


r3=re.search("(?P<n1>h)(?P<n2>\w+)",str)  #?P<>定义组里匹配内容的key(键)，<>里面写key名称，值就是匹配到的内容

print(r3.group())

print(r3.groups())

print(r3.groupdict())


hello

('h', 'ello')

{'n1': 'h', 'n2': 'ello'}
```

findall()函数

findall(pattern, string, flags=0)
 # pattern： 正则模型
 # string ： 要匹配的字符串
 # falgs ： 匹配模式

浏览全部字符串，匹配所有合规则的字符串，匹配到的字符串放到一个列表中，未匹配成功返回空列表

注意：一旦匹配成，再次匹配，是从前一次匹配成功的，后面一位开始的，也可以理解为匹配成功的字符串，不在参与下次匹配

```
import re

r=re.findall("\d+\w\d+","a2b3c4d5") #浏览全部字符串，匹配所有合规则的字符串，匹配到的字符串方到一个列表中

print(r)



['2b3', '4d5'] #匹配成功的字符串，不再参与下次匹配，所以3c4也符合规则但是没有匹配到
```

注意：如果没写匹配规则，也就是空规则，返回的是一个比原始字符串多一位的，空字符串列表

```
import re

r=re.findall("","a2b3c4d5") #浏览全部字符串，匹配所有合规则的字符串，匹配到的字符串方到一个列表中

print(r)



['', '', '', '', '', '', '', '', ''] #如果没有写匹配规则，也就是空规则，返回的是一个比原始字符串多一位的空字符串列表，如上是8个字符，返回是9个空字符
```

注意：正则匹配到空字符的情况，如果规则里只有一个组，而组后面是*就表示组里的内容可以是0个或者多过，这样组里就有了两个意思，一个意思是匹配组里的内容，二个意思是匹配组里0内容（即是空白）所以尽量避免用*否则会有可能匹配出空字符串

注意：正则只拿组里最后一位，如果规则里只有一个组，匹配到的字符串里在拿组内容是，拿的是匹配到的内容最后一位

```
import re

r=re.findall("(ca)*","ca2b3caa4d5") #浏览全部字符串，匹配所有合规则的字符串，匹配到的字符串方到一个列表中

print(r)



['ca', '', '', '', 'ca', '', '', '', '', '']#用*号会匹配出空字符
```

无分组：匹配所有合规则的字符串，匹配到的字符串放到一个列表中

```
import re

r=re.findall("a\w+","ca2b3 caa4d5") #浏览全部字符串，匹配所有合规则的字符串，匹配到的字符串方到一个列表中

print(r)

['a2b3', 'aa4d5']#匹配所有合规则的字符串，匹配到的字符串放入列表
```

有分组：只将匹配到的字符串里，组的部分放到列表里返回，相当于groups()方法

```
import re

r=re.findall("a(\w+)","ca2b3 caa4d5") #有分组：只将匹配到的字符串里，组的部分放到列表里返回

print(r)

['2b3', 'a4d5']#返回匹配到组里的内容返回
```

多个分组：只将匹配到的字符串里，组的部分放到一个元组中，最后将所有元组放到一个列表里返

相当于在group()结果里再将组的部分，分别，拿出来放入一个元组，最后将所有元组放入一个列表返回

```
import re

r=re.findall("(a)(\w+)","ca2b3 caa4d5") #有多分组：只将匹配到的字符串里，组的部分放到一个元组中，最后将所有元组放到一个列表里返回

print(r)

[('a', '2b3'), ('a', 'a4d5')]#返回的是多维数组
```

分组中有分组：只将匹配到的字符串里，组的部分放到一个元组中，先将包含有组的组，看作一个整体也就是一个组，把这个整体组放入一个元组里，然后在把组里的组放入一个元组，最后将所有组放入一个列表返回

```
import re

r=re.findall("(a)(\w+(b))","ca2b3 caa4b5") #分组中有分组：只将匹配到的字符串里，组的部分放到一个元组中，先将包含有组的组，看作一个整体也就是一个组，把这个整体组放入一个元组里，然后在把组里的组放入一个元组，最后将所有组放入一个列表返回

print(r)

[('a', '2b', 'b'), ('a', 'a4b', 'b')]#返回的是多维数组
```

?:在有分组的情况下findall()函数，不只拿分组里的字符串，拿所有匹配到的字符串，注意?:只用于不是返回正则对象的函数如findall()

```
import re

r=re.findall("a(?:\w+)","a2b3 a4b5 edd") #?:在有分组的情况下，不只拿分组里的字符串，拿所有匹配到的字符串，注意?:只用于不是返回正则对象的函数如findall()

print(r)



['a2b3', 'a4b5']
```

split()函数

根据正则匹配分割字符串，返回分割后的一个列表

```
split(pattern, string, maxsplit=0, flags=0)

# pattern:  正则模型
# string :  要匹配的字符串
# maxsplit: 指定分割个数
# flags  :  匹配模式
```

```
import re

r=re.split("a\w","sdfadfdfadsfsfafsff")

print(r)

r2=re.split("a\w","sdfadfdfadsfsfafsff",maxsplit=2)

print(r2)



C:\Users\zhaow\AppData\Local\Programs\Python\Python37\python.exe D:/study/python/atp/lib/t.py

['sdf', 'fdf', 'sfsf', 'sff']

['sdf', 'fdf', 'sfsfafsff']
```

sub()函数

替换匹配成功的指定位置字符串

```
sub(pattern, repl, string, count=0, flags=0)

# pattern:  正则模型
# repl   :  要替换的字符串
# string :  要匹配的字符串
# count  :  指定匹配个数
# flags  :  匹配模式
```

```
import re

r=re.sub("a\w","替换","sdfadfdfadsfsfafsff")

print(r)



C:\Users\zhaow\AppData\Local\Programs\Python\Python37\python.exe D:/study/python/atp/lib/t.py

sdf替换fdf替换sfsf替换sff
```

subn()函数

替换匹配成功的指定位置字符串,并且返回替换次数，可以用两个变量分别接受

```
subn(pattern, repl, string, count=0, flags=0)

# pattern:  正则模型
# repl   :  要替换的字符串
# string :  要匹配的字符串
# count  :  指定匹配个数
# flags  :  匹配模式
```

```
import  re

a,b=re.subn("a\w","替换","sdfadfdfadsfsfafsff") #替换匹配成功的指定位置字符串,并且返回替换次数，可以用两个变量分别接受

print(a) #返回替换后的字符串

print(b) #返回替换次数



C:\Users\zhaow\AppData\Local\Programs\Python\Python37\python.exe D:/study/python/atp/lib/t.py

sdf替换fdf替换sfsf替换sff

3
```

备注：参考网站 https://www.cnblogs.com/zjltt/p/6955965.html