正则re解析

什么是正则表达式

通俗理解：按照一定的规则，从某个字符串中匹配出想要的数据。这个规则就是正则表达式。

正则表达式常用匹配规则

#1 匹配某个字符串

```
text = 'hello, he is a boy!'
ret = re.match('he',text)  #match匹配的要从字符开头寻找
print(ret.group())  # he
```

#2 点.可以匹配任意的字符，默认除了换行符

```
text = "\nabc\n"
ret = re.match('.',text)
print(ret.group())  #AttributeError: 'NoneType' object has no attribute 'group'
ret = re.search('.',text) # search方法匹配的字符串可以在任何地方
print(ret.group())  # a
```

注意：加上flags = re.DOTALL或者re.S可以使 "." 特殊字符完全匹配任何字符，包括换行；没有这个标志， "." 匹配除了换行外的任何字符

#3 \d匹配任意的数字

```
text = '喂，110, he is a boy!'
ret = re.search('\d\d\d',text)
print(ret.group())  # 110
```

#4 \D匹配任意的非数字

```
text = '喂，110, he is a boy!'
ret = re.search('\D',text)
print(ret.group())  # 喂
```

#5 \s匹配任意的空白字符（\n \r \t 空格）

```
text = '喂\t110, he is a boy!'
ret = re.search('\s',text)
print(ret.group())  #
```

#6 \S匹配任意的非空白字符

```
text = '喂\t110, he is a boy!'
ret = re.search('\S',text)
print(ret.group())  #喂
```

#7 \w匹配的是a-z A-Z以及数字和下划线_

```
text = '一1二2'
print(re.findall(r'\w',text,re.A))  # ['1', '2']
print(re.findall(r'\w',text))  # ['一', '1', '二', '2']
```

#8 \W匹配的是除了a-z A-Z以及数字和下划线_以为的其他字符

#9 [ ]组合的方式，只要满足中括号中的某一项都算匹配成功

之前讲到的几种匹配规则，其实可以使用中括号的形式来进行替代：

(1) \d：[0-9]
(2) \D：0-9
(3) \w：[0-9a-zA-Z_]
(4) \W：[^0-9a-zA-Z_]

匹配多个字符

*：可以匹配0或者任意多个字符。示例代码如下：

```
text = "0731"
ret = re.match('\d*',text)
print(ret.group())  #0731
```

+：可以匹配1个或者多个字符。最少一个。示例代码如下：

```
text = "abc"
ret = re.match('\w+',text)
print(ret.group())  #abc
```

?：匹配的字符可以出现一次或者不出现（0或者1）。示例代码如下：

```
text = "123"
ret = re.match('\d?',text)
print(ret.group())  # 1
```

{m}：匹配m个字符。示例代码如下：

```
text = "123"
ret = re.match('\d{2}',text)
print(ret.group())  #12
```

{m,n}：匹配m-n个字符。在这中间的字符都可以匹配到。示例代码如下：

```
text = "123"
ret = re.match('\d{1,2}',text)
print(ret.group())  #12
```

^（脱字号）：表示以...开始：

```
text = "hello"
ret = re.match('^h',text)
print(ret.group())  #h
```

注意：如果是在中括号中，那么代表的是取反操作.

$：表示以...结束：

```
# 匹配163.com的邮箱
text = "xxx@163.com"
ret = re.search('\w+@163\.com$',text)
print(ret.group())  #xxx@163.com
```

|：匹配多个表达式或者字符串：

小例子：验证URL：（注意下面一定要加表示分组的括号哦）

```
text = "http://www.baidu.com/"
ret = re.match('(http|https|ftp)://[^\s]+',text)
print(ret.group())  #http://www.baidu.com/
```

我们知道正常情况下，\w匹配字母数字及下划线，相当于[A-Za-z0-9_]。
在python 3中我们试下\w的匹配字 符串的时候,会发现匹配会匹配到中文汉字。
这是什么原因呢？在python 3里面,默认的是Unicode编码。正则也是默认的编码模式。
我们知道unicode编码由字母和数字构成。这就造成了\w可以匹配到中文。
那么怎么处理呢？
我们需要将正则的匹配模式修改为二进制匹配，就会得到正确的结果，即flags设置为re.A就可以了。

正则re解析

贪婪模式和非贪婪模式

贪婪模式：正则表达式会匹配尽量多的字符。默认是贪婪模式。

示例代码如下：

```
1.  text = "0123456"
2.  ret = re.match('\d+',text)
3.  print(ret.group())  #0123456
4.  # 因为默认采用贪婪模式，所以会输出0123456
```

非贪婪模式：正则表达式会尽量少的匹配字符。

只需要在* + ?或者{}这些符号后面再加个?就可以了

```
1.  text = "0123456"
2.  ret = re.match('\d+?',text)
3.  print(ret.group())  #0
```

在爬虫中，多半都是非贪婪模式哦

转义字符和原生字符串

在正则表达式中，有些字符是有特殊意义的字符。因此如果想要匹配这些字符，那么就必须使用反斜杠进行转义。

```
1.  text = "apple price is \$99,orange paice is $88"
2.  ret = re.search('\$(\d+)',text)
3.  print(ret.group())  # $99
```

在正则表达式中，\是专门用来做转义的。在Python中\也是用来做转义的。
因此如果想要在普通的字符串中匹配出\，那么要给出四个\。

```
1.  text = "apple \c"
2.  ret = re.search('\\\\c',text)
3.  print(ret.group())  #\c
```

要使用原生字符串就可以解决这个问题：

```
1.  text = "apple \c"
2.  ret = re.search(r'\\c',text)
3.  print(ret.group())  #\c
```

re模块中常用函数

match

从开始的位置进行匹配。如果开始的位置没有匹配到。就直接失败了。

```
1.  text = 'hello'
2.  ret = re.match('h',text)
3.  print(ret.group())  #h
```

如果第一个字母不是h，那么就会失败。示例代码如下：

```
1.  text = 'ahello'
2.  ret = re.match('h',text)
3.  print(ret.group())
4.  >> AttributeError: 'NoneType' object has no attribute 'group'
```

search

在字符串中找满足条件的字符。如果找到，就返回。

```
1.  text = 'apple price $99 orange price $88'
2.  ret = re.search('\d+',text)
3.  print(ret.group())  #99
```

group分组

在正则表达式中，可以对过滤到的字符串进行分组。分组使用圆括号()的方式。
group()：它和group(0)是等价的，返回的是整个满足条件的字符串。
groups：返回的是里面的子组，用元组返回。
group(1)：返回的是第一个子组，可以传入多个。

举例

```
1.  text = 'apple is $5, orange is $17.'
2.  ret = re.search(r'.*(\$\d+).*(\$\d+).*',text)
3.  print(ret.group()) # apple is $5, orange is $17.
4.  print(ret.group(0)) # apple is $5, orange is $17.
5.  print(ret.group(1)) # $5
6.  print(ret.group(2)) # $17
7.  print(ret.group(1,2)) # ('$5', '$17') 返回了一个元组
8.  print(ret.groups()) # ('$5', '$17')
```

findall

找出所有满足条件的，返回的是一个列表。

```
1.  text = 'apple price $99 orange price $88'
2.  ret = re.findall('\d+',text)
3.  print(ret)
4.  # ['99', '88']
```

sub

用来替换字符串。将匹配到的字符串替换为其他字符串。

```
1.  text = 'apple price $99 orange price $88'
2.  ret = re.sub('\d+','0',text)
3.  print(ret)  #>> apple price $0 orange price $0
```

split

使用正则表达式来分割字符串。

```
1.  text = "hello world ni hao"
2.  ret = re.split('\W',text)
3.  print(ret)
4.  # ["hello","world","ni","hao"]
```

compile

对于一些经常要用到的正则表达式，可以使用compile进行编译，后期再使用的时候可以直接拿过来用，执行效率会更快。

```
1.  import re
2.  r = re.compile(r'.{2}')
3.  text = '123456\n789abc'
4.  ret = re.search(r,text)  #12
```

注意：如果传入的是compile对象，那么在search,match,findall等方法里面不允许有flags这个参数，因为此时正则表达式已经被编译完成了储存在内存中了。