正则表达式

常见匹配模式

模式	描述
\w	匹配字母数字及下划线
\W	匹配非字母数字下划线
\s	匹配任意空白字符，等价于 [\t\n\r\f].
\S	匹配任意非空字符
\d	匹配任意数字，等价于 [0-9]
\D	匹配任意非数字
\A	匹配字符串开始
\Z	匹配字符串结束，如果是存在换行，只匹配到换行前的结束字符串
\z	匹配字符串结束
\G	匹配最后匹配完成的位置
\n	匹配一个换行符
\t	匹配一个制表符
^	匹配字符串的开头
$	匹配字符串的末尾。
.	匹配任意字符，除了换行符，当re.DOTALL标记被指定时，则可以匹配包括换行符的任意字符。
[...]	用来表示一组字符,单独列出：[amk] 匹配 'a'，'m'或'k'
[^...]	不在[]中的字符：[^abc] 匹配除了a,b,c之外的字符。
*	匹配0个或多个的表达式。
+	匹配1个或多个的表达式。
?	匹配0个或1个由前面的正则表达式定义的片段，非贪婪方式
{n}	精确匹配n个前面表达式。
{n, m}	匹配 n 到 m 次由前面的正则表达式定义的片段，贪婪方式
a|b	匹配a或b
( )	匹配括号内的表达式，也表示一个组

RE模块中的函数说明

方法/属性	作用
match()	决定 RE 是否在字符串刚开始的位置匹配
fullmatch()	完全匹配string
search()	扫描字符串，找到这个 RE 匹配的位置
findall()	找到 RE 匹配的所有子串，并把它们作为一个列表返回
finditer()	找到 RE 匹配的所有子串，并把它们作为一个迭代器返回
purge()	清除正则表达式缓存
sub()	替换字符串中每一个匹配的子串后返回替换后的字符串
split()	以列表形式返回分割的字符串
escape()	正则运算符的字符进行转义的应用函数
compile()	将正则字符串编译成正则表达式对象

match()、seerch（）、finditer（）如果匹配成功则返回一个**Match Object**对象，该对象有以下属性、方法：

方法/属性	作用
group()	返回被 RE 匹配的字符串
start()	返回匹配开始的位置
end()	返回匹配结束的位置
span()	返回一个元组包含匹配 (开始,结束) 的位置

group() 返回re整体匹配的字符串，可以一次输入多个组号，对应组号匹配的字符串。

1. group（）返回re整体匹配的字符串，
2. group (n,m) 返回组号为n，m所匹配的字符串，如果组号不存在，则返回indexError异常
3. groups() 方法返回一个包含正则表达式中所有小组字符串的元组，从 1 到 所含的小组号，通常groups（）不需要参数，返回一个元组，元组中的元就是正则表达式中定义的组。

函数参数(pattern, string, flags)

1. *参数pattern是正则表达式，这里为"(\w+)\s"，如果匹配成功，则返回一个Match，否*

则返回一个None；

2. *参数string表示要匹配的字符串；*
3. *参数flags是标致位，用于控制正则表达式的匹配方式，如：是否区分大小写，多行匹配等等。*

---

re.match演示

函数原型：`re.match(pattern, string, flags=0)`

最常规的匹配

```
1  import re
2
3  content = 'Hello 123 4567 World_This is a Regex Demo'
4  print(len(content))
5  result = re.match('^Hello\s\d\d\d\s\d{4}\s\w{10}.*Demo$', content)
6  print(result)
7  print(result.group())
8  print(result.span())
```

```
1  41
2  <_sre.SRE_Match object; span=(0, 41), match='Hello 123 4567 World_Th
3  is is a Regex Demo'>
4  Hello 123 4567 World_This is a Regex Demo
   (0, 41)
```

泛匹配

```
1  import re
2
3  content = 'Hello 123 4567 World_This is a Regex Demo'
4  result = re.match('^Hello.*Demo$', content)
5  print(result)
6  print(result.group())
7  print(result.span())
```

```
1  <_sre.SRE_Match object; span=(0, 41), match='Hello 123 4567 World_Th
2  is is a Regex Demo'>
3  Hello 123 4567 World_This is a Regex Demo
   (0, 41)
```

匹配目标

```
import re

content = 'Hello 1234567 World_This is a Regex Demo'
result = re.match('^Hello\s(\d+)\sWorld.*Demo$', content)
print(result)
print(result.group(1))
print(result.span())
```

```
<_sre.SRE_Match object; span=(0, 40), match='Hello 1234567 World_Thi
s is a Regex Demo'>
1234567
(0, 40)
```

贪婪匹配

```
import re

content = 'Hello 1234567 World_This is a Regex Demo'
result = re.match('^He.*(\d+).*Demo$', content)
print(result)
print(result.group(1))
```

```
<_sre.SRE_Match object; span=(0, 40), match='Hello 1234567 World_Thi
s is a Regex Demo'>
7
```

非贪婪匹配

```
import re

content = 'Hello 1234567 World_This is a Regex Demo'
result = re.match('^He.*?(\d+).*Demo$', content)
print(result)
print(result.group(1))
```

```
<_sre.SRE_Match object; span=(0, 40), match='Hello 1234567 World_Thi
s is a Regex Demo'>
1234567
```

匹配模式

```
import re

content = '''Hello 1234567 World_This
```

```
is a Regex Demo
'''
result = re.match('^He.*?(\d+).*?Demo$', content, re.S)
print(result.group(1))
```

```
1234567
```

转义

```
import re

content = 'price is $5.00'
result = re.match('price is $5.00', content)
print(result)
```

```
None
```

```
import re

content = 'price is $5.00'
result = re.match('price is \$5\.00', content)
print(result)
```

```
<_sre.SRE_Match object; span=(0, 14), match='price is $5.00'>
```

总结：尽量使用泛匹配、使用括号得到匹配目标、尽量使用非贪婪模式、有换行符就用re.S

---

re.search演示

函数原型：`re.search(pattern,string,flags)`

```
import re

content = 'Extra stings Hello 1234567 World_This is a Regex Demo Ext
ra stings'
result = re.match('Hello.*?(\d+).*?Demo', content)
print(result)
```

```
None
```

```
import re

content = 'Extra stings Hello 1234567 World_This is a Regex Demo Ext
```

```
ra stings'
result = re.search('Hello.*?(\d+).*?Demo', content)
print(result)
print(result.group(1))
```

```
<_sre.SRE_Match object; span=(13, 53), match='Hello 1234567 World_Th
is is a Regex Demo'>
1234567
```

总结：为匹配方便，能用search就不用match

匹配演练

```
import re

html = '''<div id="songs-list">
    <h2 class="title">经典老歌</h2>
    <p class="introduction">
        经典老歌列表
    </p>
    <ul id="list" class="list-group">
        <li data-view="2">一路上有你</li>
        <li data-view="7">
            <a href="/2.mp3" singer="任贤齐">沧海一声笑</a>
        </li>
        <li data-view="4" class="active">
            <a href="/3.mp3" singer="齐秦">往事随风</a>
        </li>
        <li data-view="6"><a href="/4.mp3" singer="beyond">光辉岁月</
a></li>
        <li data-view="5"><a href="/5.mp3" singer="陈慧琳">记事本</a><
/li>
        <li data-view="5">
            <a href="/6.mp3" singer="邓丽君"><i class="fa fa-user"></
i>但愿人长久</a>
        </li>
    </ul>
</div>'''
result = re.search('<li.*?active.*?singer="(.*?)">(.*?)</a>', html,
re.S)
if result:
    print(result.group(1), result.group(2))
```

```
齐秦 往事随风
```

```python
import re

html = '''<div id="songs-list">
    <h2 class="title">经典老歌</h2>
    <p class="introduction">
        经典老歌列表
    </p>
    <ul id="list" class="list-group">
        <li data-view="2">一路上有你</li>
        <li data-view="7">
            <a href="/2.mp3" singer="任贤齐">沧海一声笑</a>
        </li>
        <li data-view="4" class="active">
            <a href="/3.mp3" singer="齐秦">往事随风</a>
        </li>
        <li data-view="6"><a href="/4.mp3" singer="beyond">光辉岁月</
a></li>
        <li data-view="5"><a href="/5.mp3" singer="陈慧琳">记事本</a><
/li>
        <li data-view="5">
            <a href="/6.mp3" singer="邓丽君">但愿人长久</a>
        </li>
    </ul>
</div>'''
result = re.search('<li.*?singer="(.*?)">(.*?)</a>', html, re.S)
if result:
    print(result.group(1), result.group(2))
```

```
任贤齐 沧海一声笑
```

```python
import re

html = '''<div id="songs-list">
    <h2 class="title">经典老歌</h2>
    <p class="introduction">
        经典老歌列表
    </p>
    <ul id="list" class="list-group">
        <li data-view="2">一路上有你</li>
        <li data-view="7">
            <a href="/2.mp3" singer="任贤齐">沧海一声笑</a>
        </li>
        <li data-view="4" class="active">
            <a href="/3.mp3" singer="齐秦">往事随风</a>
```

```
        </li>
        <li data-view="6"><a href="/4.mp3" singer="beyond">光辉岁月</
a></li>
        <li data-view="5"><a href="/5.mp3" singer="陈慧琳">记事本</a><
/li>
        <li data-view="5">
            <a href="/6.mp3" singer="邓丽君">但愿人长久</a>
        </li>
    </ul>
</div>'''
result = re.search('<li.*?singer="(.*?)">(.*?)</a>', html)
if result:
    print(result.group(1), result.group(2))
```

```
beyond 光辉岁月
```

re.findall演示

函数原型：`re.findall(pattern,string,flags)`

```
import re

html = '''<div id="songs-list">
    <h2 class="title">经典老歌</h2>
    <p class="introduction">
        经典老歌列表
    </p>
    <ul id="list" class="list-group">
        <li data-view="2">一路上有你</li>
        <li data-view="7">
            <a href="/2.mp3" singer="任贤齐">沧海一声笑</a>
        </li>
        <li data-view="4" class="active">
            <a href="/3.mp3" singer="齐秦">往事随风</a>
        </li>
        <li data-view="6"><a href="/4.mp3" singer="beyond">光辉岁月</
a></li>
        <li data-view="5"><a href="/5.mp3" singer="陈慧琳">记事本</a><
/li>
        <li data-view="5">
            <a href="/6.mp3" singer="邓丽君">但愿人长久</a>
        </li>
    </ul>
```

```
</div>'''
results = re.findall('<li.*?href="(.*?)".*?singer="(.*?)">(.*?)</a>'
, html, re.S)
print(results)
print(type(results))
for result in results:
    print(result)
    print(result[0], result[1], result[2])
```

```
[('/2.mp3', '任贤齐', '沧海一声笑'), ('/3.mp3', '齐秦', '往事随风'), ('/
4.mp3', 'beyond', '光辉岁月'), ('/5.mp3', '陈慧琳', '记事本'), ('/6.mp3
', '邓丽君', '但愿人长久')]
<class 'list'>
('/2.mp3', '任贤齐', '沧海一声笑')
/2.mp3 任贤齐 沧海一声笑
('/3.mp3', '齐秦', '往事随风')
/3.mp3 齐秦 往事随风
('/4.mp3', 'beyond', '光辉岁月')
/4.mp3 beyond 光辉岁月
('/5.mp3', '陈慧琳', '记事本')
/5.mp3 陈慧琳 记事本
('/6.mp3', '邓丽君', '但愿人长久')
/6.mp3 邓丽君 但愿人长久
```

```
import re

html = '''<div id="songs-list">
    <h2 class="title">经典老歌</h2>
    <p class="introduction">
        经典老歌列表
    </p>
    <ul id="list" class="list-group">
        <li data-view="2">一路上有你</li>
        <li data-view="7">
            <a href="/2.mp3" singer="任贤齐">沧海一声笑</a>
        </li>
        <li data-view="4" class="active">
            <a href="/3.mp3" singer="齐秦">往事随风</a>
        </li>
        <li data-view="6"><a href="/4.mp3" singer="beyond">光辉岁月</
a></li>
        <li data-view="5"><a href="/5.mp3" singer="陈慧琳">记事本</a><
/li>
        <li data-view="5">
```

```
            <a href="/6.mp3" singer="邓丽君">但愿人长久</a>
        </li>
    </ul>
</div>'''
results = re.findall('<li.*?>\s*?(<a.*?>)?(\w+)(</a>)?\s*?</li>', ht
ml, re.S)
print(results)
for result in results:
    print(result[1])
```

```
[('', '一路上有你', ''), ('<a href="/2.mp3" singer="任贤齐">', '沧海一声
笑', '</a>'), ('<a href="/3.mp3" singer="齐秦">', '往事随风', '</a>'),
 ('<a href="/4.mp3" singer="beyond">', '光辉岁月', '</a>'), ('<a href
="/5.mp3" singer="陈慧琳">', '记事本', '</a>'), ('<a href="/6.mp3" sin
ger="邓丽君">', '但愿人长久', '</a>')]
一路上有你
沧海一声笑
往事随风
光辉岁月
记事本
但愿人长久
```

re.findter演示

函数原型：`re.findter(pattern,string,flags)`

在前面学习了findall()函数，它可以一次性找到多个匹配的字符串，但是不能提供所在的位置，并且是一起返回的，如果有数万个一起返回来，就不太好处理了，因此要使用finditer()函数来实现每次只返回一个，并且返回所在的位置。

```
import re

html = '''<div id="songs-list">
    <h2 class="title">经典老歌</h2>
    <p class="introduction">
        经典老歌列表
    </p>
    <ul id="list" class="list-group">
        <li data-view="2">一路上有你</li>
        <li data-view="7">
            <a href="/2.mp3" singer="任贤齐">沧海一声笑</a>
        </li>
        <li data-view="4" class="active">
            <a href="/3.mp3" singer="齐秦">往事随风</a>
```

```
        </li>
        <li data-view="6"><a href="/4.mp3" singer="beyond">光辉岁月</
a></li>
        <li data-view="5"><a href="/5.mp3" singer="陈慧琳">记事本</a><
/li>
        <li data-view="5">
            <a href="/6.mp3" singer="邓丽君">但愿人长久</a>
        </li>
    </ul>
</div>'''
results = re.finditer('<li.*?href="(.*?)".*?singer="(.*?)">(.*?)</a>
', html, re.S)
for result in results:
    print(result.groups())
    start = result.start() #开始位置
    end = result.end()   #结束为止
    print('原字符串切片: \n{0}\n索引位置:{1}:{2}'.format(html[start:end
],start,end))
```

```
('/2.mp3', '任贤齐', '沧海一声笑')
原字符串切片:
<li data-view="2">一路上有你</li>
        <li data-view="7">
            <a href="/2.mp3" singer="任贤齐">沧海一声笑</a>
索引位置:153:260
('/3.mp3', '齐秦', '往事随风')
原字符串切片:
<li data-view="4" class="active">
            <a href="/3.mp3" singer="齐秦">往事随风</a>
索引位置:283:366
('/4.mp3', 'beyond', '光辉岁月')
原字符串切片:
<li data-view="6"><a href="/4.mp3" singer="beyond">光辉岁月</a>
索引位置:389:448
('/5.mp3', '陈慧琳', '记事本')
原字符串切片:
<li data-view="5"><a href="/5.mp3" singer="陈慧琳">记事本</a>
索引位置:462:517
('/6.mp3', '邓丽君', '但愿人长久')
原字符串切片:
<li data-view="5">
            <a href="/6.mp3" singer="邓丽君">但愿人长久</a>
索引位置:531:601
```

re.sub演示

函数原型：`re.sub(pattern,string,flags)`

```
import re

content = 'Extra stings Hello 1234567 World_This is a Regex Demo Ext
ra stings'
content = re.sub('\d+', '', content)
print(content)
```

```
Extra stings Hello  World_This is a Regex Demo Extra stings
```

```
import re

content = 'Extra stings Hello 1234567 World_This is a Regex Demo Ext
ra stings'
content = re.sub('\d+', 'Replacement', content)
print(content)
```

```
Extra stings Hello Replacement World_This is a Regex Demo Extra stin
gs
```

```
import re

content = 'Extra stings Hello 1234567 World_This is a Regex Demo Ext
ra stings'
content = re.sub('(\d+)', r'\1 8910', content)
print(content)
```

```
Extra stings Hello 1234567 8910 World_This is a Regex Demo Extra sti
ngs
```

```
import re

html = '''<div id="songs-list">
    <h2 class="title">经典老歌</h2>
    <p class="introduction">
        经典老歌列表
    </p>
    <ul id="list" class="list-group">
        <li data-view="2">一路上有你</li>
        <li data-view="7">
            <a href="/2.mp3" singer="任贤齐">沧海一声笑</a>
        </li>
        <li data-view="4" class="active">
```

```
            <a href="/3.mp3" singer="齐秦">往事随风</a>
        </li>
        <li data-view="6"><a href="/4.mp3" singer="beyond">光辉岁月</
a></li>
        <li data-view="5"><a href="/5.mp3" singer="陈慧琳">记事本</a><
/li>
        <li data-view="5">
            <a href="/6.mp3" singer="邓丽君">但愿人长久</a>
        </li>
    </ul>
</div>'''
```

```
import re

html = '''<div id="songs-list">
    <h2 class="title">经典老歌</h2>
    <p class="introduction">
        经典老歌列表
    </p>
    <ul id="list" class="list-group">
        <li data-view="2">一路上有你</li>
        <li data-view="7">
            <a href="/2.mp3" singer="任贤齐">沧海一声笑</a>
        </li>
        <li data-view="4" class="active">
            <a href="/3.mp3" singer="齐秦">往事随风</a>
        </li>
        <li data-view="6"><a href="/4.mp3" singer="beyond">光辉岁月</
a></li>
        <li data-view="5"><a href="/5.mp3" singer="陈慧琳">记事本</a><
/li>
        <li data-view="5">
            <a href="/6.mp3" singer="邓丽君">但愿人长久</a>
        </li>
    </ul>
</div>'''
html = re.sub('<a.*?>|</a>', '', html)
print(html)
results = re.findall('<li.*?>(.*?)</li>', html, re.S)
print(results)
for result in results:
    print(result.strip())
```

```
<div id="songs-list">
    <h2 class="title">经典老歌</h2>
    <p class="introduction">
        经典老歌列表
    </p>
    <ul id="list" class="list-group">
        <li data-view="2">一路上有你</li>
        <li data-view="7">
            沧海一声笑
        </li>
        <li data-view="4" class="active">
            往事随风
        </li>
        <li data-view="6">光辉岁月</li>
        <li data-view="5">记事本</li>
        <li data-view="5">
            但愿人长久
        </li>
    </ul>
</div>
['一路上有你', '\n                沧海一声笑\n            ', '\n                往事随风\n            ', '光辉岁月', '记事本', '\n                但愿人长久\n            ']
一路上有你
沧海一声笑
往事随风
光辉岁月
记事本
但愿人长久
```

re.split演示

函数原型：`split(string [, maxsplit = 0])`

你可以通过设置 maxsplit 值来限制分片数。当 maxsplit 非零时，最多只能有 maxsplit 个分片，字符串的其余部分被做为列表的最后部分返回。在下面的例子中，定界符可以是非数字字母字符的任意序列。

```
>>> p = re.compile(r'\W+')
>>> p.split('This is a test, short and sweet, of split().')
['This', 'is', 'a', 'test', 'short', 'and', 'sweet', 'of', 'split', '']
>>> p.split('This is a test, short and sweet, of split().', 3)
['This', 'is', 'a', 'test, short and sweet, of split().']
```

有时，你不仅对定界符之间的文本感兴趣，也需要知道定界符是什么。定界符可以是非数字字母字符的任意序列，如果捕获括号在 RE中使用，那么它们（定界符）的值也会当作列表的一部分返回。比较下面的调用：

```
re.split("([ab])","carbs")  #   ['c', 'a', 'r', 'b', 's']     定界符是a或b，结果返回界定符a、b。

re.split("([ab]#)", "carbs")  #  ['carbs']   定界符是a#或b#，结果 ['carbs']
```

re.escape演示

在使用python的过程中，你肯定对转义字符的使用苦恼过，因为有的时候我们需要使用一些特殊符号如 `"$ * . ^"` 等的原意，有时候需要被转义后的功能，并且转义字符地使用很繁琐，容易出错，那拯救你的就非re.escape莫属了。

`re.escape(pattern)` 可以对字符串中所有可能被解释为正则运算符的字符进行转义的应用函数。如果字符串很长且包含很多特殊技字符，而你又不想输入一大堆反斜杠，或者字符串来自于用户(比如通过raw_input函数获取输入的内容)，且要用作正则表达式的一部分的时候，可以使用这个函数。

```
print(re.escape('www.python.org'))
print(re.findall(re.escape('w.py'),"jw.pyji w.py.f"))
#这里的re.escape('w.py')作为了函数re.findall函数的正则表达式部分。
```

```
'www\\.python\\.org'
['w.py', 'w.py']
```

re.compile演示

函数原型：`re.compile(pattern,flags)`

将一个正则表达式串编译成正则对象，以便于复用该匹配模式

```
import re

content = '''Hello 1234567 World_This
is a Regex Demo'''
pattern = re.compile('Hello.*Demo', re.S)
result = re.match(pattern, content)
#result = re.match('Hello.*Demo', content, re.S)
print(result)
```

```
<_sre.SRE_Match object; span=(0, 40), match='Hello 1234567 World_This\nis a Regex Demo'>
```

实战练习

```
import requests
import re
content = requests.get('https://book.douban.com/').text
pattern = re.compile('<li.*?cover.*?href="(.*?)".*?title="(.*?)".*?more-meta.*?author">(.*?)</span>.*?year">(.*?)</span>.*?</li>', re.S)
results = re.findall(pattern, content)
for result in results:
    url, name, author, date = result
    author = re.sub('\s', '', author)
    date = re.sub('\s', '', date)
    print(url, name, author, date)
```

基础须知

re.match

[🍓]*re.match 尝试从字符串的起始位置匹配一个模式，如果不是起始位置匹配成功的话，match()就返回none。*

re.search

[🍓] *re.search 扫描整个字符串并返回第一个成功的匹配。*

re.findall

[🍓] *搜索字符串，以列表形式返回全部能匹配的子串。*

re.finditer

[🍓] *搜索字符串，以迭代器返回全部能匹配的子串。*

re.sub

[🍓] *替换字符串中每一个匹配的子串后返回替换后的字符串。*

re.compile

[🍓] *将正则字符串编译成正则表达式对象*

个人经验

尽量使用泛匹配、使用括号得到匹配目标、尽量使用非贪婪模式、有换行符就用re.S，为匹配方便，能用search就不用match，将一个正则表达式串编译成正则对象，以便于复用该匹配模式。

通用模版

```
pattern = re.compile('正则表达式', re.S)  #将正则字符串编译成正则表达式对象方便复用
results = re.findall(pattern, content)   #全文匹配
for result in results:  #遍历result取得匹配
    print(result)
```