



第五章 网络爬虫与信息提取

5.3 关键信息提取

同济大学 计算机基础教研室

引例

➤ 《三联生活周刊》官方微博 (<http://blog.sina.com.cn/lifeweek>)

✓ 观察网页

✓ 分析源码

✓ 数据获取



```
import re
import requests
url = "http://blog.sina.com.cn/lifeweek"
r=requests.get(url)
r.encoding=r.apparent_encoding
#获取所有包含文章标题的div标签信息
all_title=soup.find_all('div',class_='blog_title')
#获取所有文章链接和标题
for ti in all_title:
    p=r'<a.*?href="(.*?)".*?>(.*?)</a>'
    t2 = re.findall(p, str(ti))
    for t in t2:
        print(t[0],t[1])
```

http://blog.sina.com.cn/s/blog_470bf2570102xgi2.html 莼鲈之思：舌尖3文会宴中的花鲈故事

http://blog.sina.com.cn/s/blog_470bf2570102xb9a.html 读书 | 一个人睡是很自然的一件事

http://blog.sina.com.cn/s/blog_470bf2570102xb99.html 与C罗面对面：做任何事情都需要天赋

简单示例

```
import re #导入re模块
```

```
m = re.match(r'hello', 'hello world!') #r的意思是“原生字符串”
```

```
print(m.group())
```

hello

#或者，将正则表达式编译成Pattern对象

```
pattern = re.compile(r'hel*o') #*表示匹配前一个字符0或无限次
```

使用Pattern匹配文本，获得匹配结果，无法匹配时将返回None

```
m = pattern.match('heo world!')
```

heo

```
print(m.group(0))
```

```
m = pattern.match('hello world!')
```

hello

```
print(m.group(0))
```

```
m = pattern.match('helllo world!')
```

hello

```
print(m.group(0))
```

什么是正则表达式

- 正则表达式是对字符串操作的一种**逻辑公式**，就是**用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”**，这个“规则字符串”用来表达对字符串的一种**过滤逻辑**。
- 给定一个正则表达式和另一个字符串，可以达到如下的目的：
 - ✓ 给定的字符串是否符合正则表达式的过滤逻辑（“**匹配**”）；
 - ✓ 通过正则表达式，从文本字符串中获取我们想要的特定部分（“**过滤**”）。

re 模块的一般使用步骤

- 使用 **compile()** 函数将正则表达式的字符串形式编译为一个 **pattern 对象**
- 通过 pattern 对象提供的一系列方法对文本进行**匹配查找**，获得匹配结果，一个**match 对象**。
- 最后使用 match 对象提供的属性和方法获得信息，根据需要进行其他的操作

正则表达式匹配格式——字符

语法	说明	表达式实例	完整匹配的字符串
一般字符	匹配自身	abc	abc
.	匹配任意单个字符，除了换行符	a.c	abc
\	转义字符，使后一个字符改变原来意思	a\.c a\\c	a.c a\c
[...]	字符集。对应位置可以是字符集中任意字符。字符集中的字符可以逐个列出，也可以给出范围，如[abc]或[a-c]。	a[bcd]e a[b-d]e	abe ace ade
[^...]	表示取反。如[^abc]表示不是abc的其他字符	a[^bcd]e	afe

正则表达式匹配格式——预定义字符集

可以写在字符集[...]中

语法	说明	表达式实例	完整匹配的字符串
\d	数字：[0-9]任意数字	a\dc	a5c
\D	非数字：[^d]任意非数字	a\Dc	abc
\s	空白字符：[<空格>\t\r\n\f\v]任意空白字符	a\sc	a c
\S	非空白字符：[^s]任意非空字符	a\Sc	abc
\w	单词字符：[A-Za-z0-9_]字母数字下划线	a\wc	abc a6c
\W	非单词字符：[^w]非字母数字下划线	a\Wc	a c

正则表达式匹配格式——数量词

用在字符或者(...)之后

语法	说明	表达式实例	完整匹配的字符串
*	匹配前一个字符0或无限次	abc*	ab abcccc
+	匹配前一个字符1次或无限次	abc+	abc abcccc
?	匹配前一个子表达式0或1次 (跟在任何一个其他限制符 (*, +, ?, {n}, {n,}, {n,m}) 后面时, 非贪婪方式匹配)	ab(cd)?	ab abcd
{m}	匹配前一个字符m次	ab{2}c	abbc
{m,n}	匹配前一个字符/表达式m至n次 省略m, 匹配0至n次; 省略n, 匹配m至无限次 (贪婪模式)	ab{1,2}c	abc abbc

正则表达式匹配格式——边界匹配

不消耗待匹配字符串中的字符

语法	说明	表达式实例	完整匹配的字符串
^	匹配字符串的开头	<code>^abc</code>	abc abcdef
\$	匹配字符串的末尾	<code>abc\$</code>	abc defabc

正则表达式匹配格式——逻辑分组

语法	说明	表达式实例	完整匹配的字符串
	代表左右表达式任意匹配一个	abc def	abc def
(...)	被括起来的表达式将作为分组； 分组表达式作为一个整体，可以后接数量词； 表达式中的 仅在该组中有效	(abc){2} a(123 456)c	abcabc a456c

开源中国提供的正则表达式测试工具

<http://tool.oschina.net/regex/>

Pattern 对象的一些常用方法

方法	作用
re.match()	从字符串头开始匹配模式，一次匹配；如果没有匹配，返回空
re.search()	从全文(任何位置开始)搜索并返回第一个匹配子串
re.findall()	全文搜索并找到所有匹配的子串，返回一个列表
re.finditer()	全文搜索并找到所有匹配的子串，返回一个可迭代对象
re.compile()	生成一个正则表达式对象实例，实例方法与类相同（不需要重新指定模式）
re.split()	按全文中符合模式的子串作为分隔符分割字符串，返回一个列表
re.sub()	检索并替换匹配子串

正则表达式匹配的流程

正则表达式引擎

编译

正则表达式文本

正则表达式对象

匹配

需要匹配的文本

匹配结果

正则表达式引擎编译表达式文本得到的对象，包含了如何进行匹配的信息

正则表达式对象对需要匹配的文本进行匹配后得到的记过，如匹配到的字符串、分组以及索引

match引例

- `match()`方法尝试从字符串的**起始位置**匹配一个模式，如果不是起始位置匹配的话就会返回None

```
m = re.match(r'dog', 'dog cat dog')
```

```
m.group(0)
```

```
m.start()
```

```
m.end()
```

```
pattern=re.compile(r'cat')
```

```
m=pattern.match( 'dog cat dog')
```

```
m.group(0)
```

'dog'

0

3

子串最后一个字符的索引+1

Match方法实例

```
import re
pattern = re.compile(r'\d+') # 用于匹配至少一个数字
m = pattern.match('one12twothree34four') # 查找头部，没有匹配
print(m)
```

None

```
m = pattern.match('one12twothree34four', 3, 10)
# 从'1'的位置开始匹配，正好匹配
print(m)
```

<_sre.SRE_Match object at 0x10a42aac0>

- `group([group1, ...])` 方法用于获得一个或多个分组匹配的字符串，当要获得整个匹配的子串时，可直接使用 `group()` 或 `group(0)`；
- `start([group])` 方法用于获取分组匹配的子串在整个字符串中的起始位置（子串第一个字符的索引），参数默认值为 0；
- `end([group])` 方法用于获取分组匹配的子串在整个字符串中的结束位置（子串最后一个字符的索引+1），参数默认值为 0；
- `span([group])` 方法返回 `(start(group), end(group))`。

match 方法

- **用于查找字符串的头部（也可以指定起始位置）**
- 一次匹配，只要找到一个匹配的结果就返回，而不是查找所有匹配的结果。匹配成功时，返回一个 Match 对象，如果没有匹配上，则返回 None。

```
p=re.compile(pattern[,flags])  
p.match(string[, pos[, endpos]])
```

待匹配的字符串

字符串的起始位置
默认为0（匹配头部）

字符串的终点位置
默认为len(字符串长度)

- **等价于：**

```
re.match(pattern, string[, flags])
```

编译pattern时
指定匹配模式

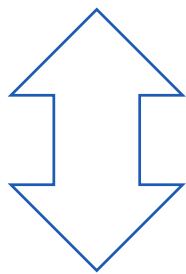
（括号内是完整写法）

- (1)re.I(re.IGNORECASE): 忽略大小写，[A-Z]能够匹配小写
- (2)re.M(re.MULTILINE): 允许多行模式，^可将每行当作匹配开始
- (3)re.S(re.DOTALL): 支持.匹配所有字符

RE库的等价用法

```
pattern = re.compile(r'\d+') # 用于匹配至少一个数字  
m = pattern.match('one12twothree34four')
```

```
p.match(string[, pos[, endpos]])
```



```
m = re.match(r'\d+', 'one12twothree34four')
```

```
re.match(pattern, string[, flags])
```

Match方法实例

```
import re
pattern = re.compile(r'([a-z]+) ([a-z]+)', re.I) # re.I 表示忽略大小写
m = pattern.match('Hello World Wide Web')
# 匹配成功，返回一个 Match 对象
print(m)
```

```
<re.Match object; span=(0, 11), match='Hello World'>
```

```
m.group(0) # 返回匹配成功的整个子串
m.span(0)  # 返回匹配成功的整个子串的索引
m.group(1) # 返回第一个分组匹配成功的子串
m.span(1)  # 返回第一个分组匹配成功的子串的索引
m.group(2) # 返回第二个分组匹配成功的子串
m.span(2)  # 返回第二个分组匹配成功的子串
m.groups() # 等价于 (m.group(1), m.group(2), ...)
m.group(3) # 不存在第三个分组
```

'Hello World'	
(0, 11)	
'Hello'	
	(0, 5)
World'	
(6, 11)	
'Hello' , 'World'	

```
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: no such group
```

常规匹配

- 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的话, `match ()` 就会返回None
- **`re.match(pattern,string,flags=0)`**

```
import re
```

```
content= "hello 2022 0428 Do_you_know?This is a Demo by XY"
```

```
result = re.match('^hello\s\d\d\d\d\s\d{4}\s\w{11}.*XY$',content)
```

```
print(result)
```

```
print(result.group())    #获取匹配的结果
```

```
print(result.span())     #获取匹配字符串的长度范围
```

- 将上述的正则规则进行简化

```
import re
```

```
content= "hello 2022 0428 Do_you_know?This is a Demo by XY"
```

```
result = re.match("^hello.*XY$",content)
```

```
print(result)
```

```
print(result.group())
```

```
print(result.span())
```


匹配目标

- 如果为了匹配字符串中具体的目标，则需要通过 () 括起来

```
import re
content= "hello 20220428 Do_you_know?This is a Demo by XY"
result = re.match('^hello\s(\d+)\sDo.*XY$',content)
print(result)
print(result.group())
print(result.group(1))
```

- 获得结果后，如果正则表达式中有括号，则re.group(1)获取的就是第一个括号中匹配的结果

贪婪匹配

```
import re  
content= "hello 20220428 Do_you_know?This is a Demo by XY"  
result= re.match('^hello.*(\d+).*XY',content)  
print(result.group(1))
```

- 只匹配到了6，并没有匹配到20220428
- 因为前面的.*会尽可能的匹配多的内容，即贪婪匹配
- 要得到20220428，可将.*修改为非贪婪模式

非贪婪模式

```
import re  
content= "hello 20220428 Do_you_know?This is a Demo by XY"  
result= re.match('^hello.*?(\d+).*XY',content)  
print(result.group(1))
```

匹配模式

- 匹配的内容存在换行时，需要用匹配模式`re.S`来匹配换行的内容

```
import re
content= "hello 20220428 Do_you_know?
This is a Demo by XY
"

result =re.match('^he.*?(\d+).*?XY$',content,re.S)

print(result.group())
print(result.group(1))
```

转义

- 当要匹配的内容中存在特殊字符时，需要用到转义符号\

```
import re
```

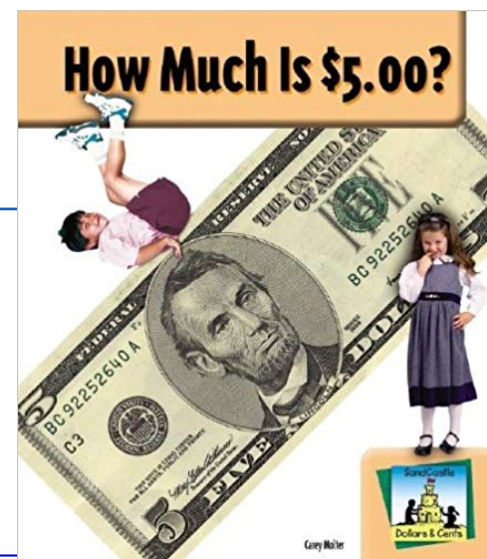
```
content= "How Much Is $5.00?"
```

```
result = re.match('How Much Is \$5\.00\?',content)
```

```
print(result)
```

```
print(result.group())
```

- 使用match方法，正则表达式中的^可省略



小结

- 尽量使用泛匹配
- 使用括号得到匹配目标时，尽量使用非贪婪模式
- 换行符可用`re.S`匹配模式
- 注意`re.match`是从字符串的起始位置匹配一个模式

经典正则表达式

表达式	说明
<code>^[A-Za-z]+\$</code>	由26个字母组成的字符串
<code>^[A-Za-z0-9]+\$</code>	由26个字母和数字组成的字符串
<code>^-?d+\$</code>	整数字符串
<code>^[1-9]\d*\$</code>	正整数字符串
<code>[1-9]\d{5}</code> 第一位1-9，共6位	中国境内邮政编码
<code>[\u4e00 - \u9fa5]</code>	中文字符
<code>^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,10}\$</code>	校验密码强度（密码的强度必须是包含大小写字母和数字的组合，不能使用特殊字符，长度在8-10之间）
<code>^[1-9]\d{5}[1-9]\d{3}((0\d) (1[0-2]))(((0 1 2)\d) 3[0-1])\d{3}([0-9] X)\$</code>	校验身份证号码
<code>^[0-9]+(\.[0-9]{2})?\$</code>	校验金额（精确到2位小数）
<code>^(13[0-9] 14[5 7] 15[0 1 2 3 5 6 7 8 9] 18[0 1 2 3 5 6 7 8 9])\d{8}\$</code>	校验手机号（13、15、18开头的手机号）

compile 函数

- compile 函数用于编译正则表达式
生成一个 Pattern 对象

```
import re  
# 将正则表达式编译成 Pattern 对象  
pattern = re.compile(r'\d+')
```

- 原始类型字符串可以简单地在普通字符串前面加一个字符'r'来创建。当一个字符串是原始类型时，Python编译器不会对其尝试做任何的替换。
- 接下来，利用 pattern 的一系列方法对文本进行匹配查找

- 将正则表达式编译成正则表达式对象，方便复用该正则表达式

```
import re  
  
content= "hello 20211201 Do_you_know?  
This is a Demo by XY  
"  
  
pattern =re.compile("hello.*Demo",re.S)  
  
result = re.match(pattern,content)  
print(result)  
print(result.group())
```

search引例

- re.search扫描整个字符串返回第一个成功匹配的结果
- search()方法不会限制只从字符串的开头查找匹配
- 但是search()方法会在它查找到一个匹配项之后停止继续查找

```
m = re.search(r'cat', 'dog cat dog')
```

```
m.group(0)
```

'cat'

```
m = re.search(r'dog', 'dog cat dog')
```

```
m.group(0)
```

'dog'

search 方法

- **search 方法用于查找字符串的任何位置**
- 是一次匹配，只要找到了一个匹配的结果就返回，而不是查找所有匹配的结果，一般使用形式如下

```
p.search(string[, pos[, endpos]])
```

待匹配的字符串

字符串的起始位置
默认为0（匹配头部）

字符串的终点位置
默认为len(字符串长度)

- 当匹配成功时，返回一个 Match 对象，如果没有匹配上，则返回 None。

等价于：

```
re.search(pattern, string[, flags])
```

Search实例

```
import re
```

```
# 将正则表达式编译成 Pattern 对象
```

```
pattern = re.compile('\d+')
```

```
m = pattern.search('one12twothree34four') # 此处若使用 match 方法  
则不匹配
```

```
m
```

```
<_sre.SRE_Match object at 0x10cc03ac0>
```

```
m.group()
```

```
'12'
```

```
m = pattern.search('one12twothree34four', 10, 30) # 指定字符串区间
```

```
m
```

```
<_sre.SRE_Match object at 0x10cc03b28>
```

```
m.group()
```

```
'34'
```

```
m.span()
```

```
(13, 15)
```


- re.search扫描整个字符串返回第一个成功匹配的结果

```
import re  
content = "Follow me say hello 20220505 Do_you_know?This is a  
Demo by XY"  
result = re.search("hello.*?(\\d+).*?Demo",content)  
print(result.group())  
print(result.group(1))
```

- 此时不需再写^以及\$, 因为search是扫描整个字符串
- 为了匹配方便, 相比match, 会更多使用search

Search例

```
html = "<div id='songs-list'>
<h2 class='title'>经典老歌</h2>
<ul id='list' class='list-group'>
<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>
</ul>
</div>"
```

```
result = re.search('<li.*?active.*?singer="(.*?)">(.*?)</a>',html,re.S)
print(result.group(1),result.group(2))
result = re.search('<li.*?data-view=.*?singer="(.*?)">(.*?)</a>',html,re.S)
print(result.group(1),result.group(2))
```

findall引例

- **以列表形式返回全部能匹配的子串**
- 结果直接是列表，而不是match对象

```
['cat']
```

✓ `re.findall(r'cat', 'dog cat dog')`

✓ `re.findall(r'dog', 'dog cat dog')`

```
['dog', 'dog']
```

✓ `re.findall(r'cat\d*', 'dog cat dog cat1 cat12')`

```
['cat', 'cat1', 'cat12']
```

findall 方法

- findall 以**列表**形式返回**全部**能匹配的子串
- 如果没有匹配，则返回一个空列表。

```
p.findall(string[, pos[, endpos]])
```

待匹配的字符串

字符串的起始位置
默认为0（匹配头部）

字符串的终点位置
默认为len(字符串长度)

等价于：

```
re.findall(pattern, string[, flags])
```

findall实例

```
import re
pattern = re.compile(r'\d+') # 查找数字
r1 = pattern.findall('hello 123456 789')
r2 = pattern.findall('one1two2three3four4', 0, 10)
print(r1)
print(r2)
```

```
['123456', '789']
['1', '2']
```

findall实例

```
import re
```

```
pattern = re.compile(r'\d+\.\d*')
```

```
#通过pattern.findall()方法全部匹配得到的字符串
```

```
result = pattern.findall("123.141593, 'bigcat', 232312, 3.15")
```

```
#findall 以 列表形式 返回全部能匹配的子串给result
```

```
for item in result:
```

```
    print(item)
```

```
123.141593  
3.15
```

re.findall

- 搜索字符串，以列表的形式返回全部能匹配的子串

```
results = re.findall('<li.*?href="(.*?)".*?singer="(.*?)">(.*?)</a>', html,  
re.S)
```

```
print(type(results))  
for result in results:  
    print(result[0], result[1], result[2])
```

```
results = re.findall('<li.*?>\s*?(<a.*?>)?(\w+)(</a>)?\s*?</li>',html,re.S)
```

```
for result in results:  
    #print(result[0])  
    print(result[1])
```

- (<a.*?>)? 因为html中有的有a标签，有的没有，? 表示匹配一个或0个，正好可以用于匹配

finditer 方法


- **finditer 方法搜索整个字符串，获得所有匹配的结果。**
- 返回一个顺序访问每一个匹配结果（Match 对象）的迭代器。

```
p.finditer(string[, pos[, endpos]])
```

待匹配的字符串

字符串的起始位置
默认为0（匹配头部）

字符串的终点位置
默认为len(字符串长度)



```
import re
pattern = re.compile(r'\d+')
result_iter1 = pattern.finditer('hello 123456 789')
result_iter2 = pattern.finditer('one1two2three3four4', 0, 10)
type(result_iter1)
type(result_iter2)
print('result1...')
for m1 in result_iter1: # m1 是 Match 对象
    print('matching string: {}, position: {}'.format(m1.group(), m1.span()))
print('result2...')
for m2 in result_iter2:
    print('matching string: {}, position: {}'.format(m2.group(), m2.span()))
```

```
<type 'callable-iterator'>
```

```
<type 'callable-iterator'>
```

```
result1...
```

```
matching string: 123456, position: (6, 12)
```

```
matching string: 789, position: (13, 16)
```

```
result2...
```

```
matching string: 1, position: (3, 4)
```

```
matching string: 2, position: (7, 8)
```

split 方法

- **split** 方法按照能够匹配的子串将字符串分割后返回列表

`split(string[, maxsplit])`

待匹配的字符串

指定最大分割次数，不指定将全部分割

```
import re
```

```
p = re.compile(r'[\s\,\,;]+')
```

```
print p.split('a,b;; c d')
```

`['a', 'b', 'c', 'd']`

- 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.sub(正则表达式, 替换成的字符串, 原字符串)`

```
content = "Follow me say hello 20220505 Do_you_know?This is a Demo  
by XY"  
content = re.sub('\d+', "", content) #将数字替换为空  
print(content)
```

- 有些情况下替换字符时，还想获取我们匹配的字符串，并在后面添加一些内容

```
content = "Follow me say hello 20220505 Do_you_know?This is a Demo  
by XY"  
content = re.sub('(\d+)', r'\1 16:00', content)  
print(content)
```

- `\1`是获取第一个匹配的结果，为了防止转义字符的问题，我们需要在前面加上`r`

sub 方法

➤ sub 方法用于替换

替换字符串中每一个匹配的子串后返回替换后的字符串

```
p.sub(repl, string[, count])
```

要被查找替换的
原始待匹配字符串

指定最多替换次数，不
指定时全部替换

替换的字符串或者一个函数。

字符串：使用 repl 去替换字符串中每一个匹配的子串，并返回替换后的字符串；

函数：这个方法应当只接受一个参数（Match 对象），并返回一个字符串用于替换

```
re.sub(pattern, repl, string)
```

sub 方法实例

```
import re
p = re.compile(r'(\w+) (\w+)') # \w = [A-Za-z0-9]
s = 'hello 123, hello 456'
print(p.sub(r'hello world', s)) # 使用 'hello world' 替换 'hello 123' 和
'hello 456'
#对符合p模式的每一个匹配子串，用'\2加油'代替该子串
print(p.sub(r'\2加油', s))
print(p.sub(r'\2 \1', s)) # 引用分组
def func(m):
    return 'hi' + ' ' + m.group(2)
print(p.sub(func, s))
print(p.sub(func, s, 1))      # 最多替换一次
```

```
hello world, hello world
123加油, 456加油
123 hello, 456 hello
hi 123, hi 456
hi 123, hello 456
```

贪婪模式与非贪婪模式

- **贪婪模式：**在整个表达式匹配成功的前提下，尽可能多的匹配 (*)
- **非贪婪模式：**在整个表达式匹配成功的前提下，尽可能少的匹配 (?)
- Python里数量词默认是贪婪的

✓ `m=re.search(r'ab*','abbbc')`

✓ `m.group(0)`

尽可能多匹配 b

abbb

✓ `m=re.search(r'ab*?','abbbc')`

✓ `m.group(0)`

尽可能少匹配 b

a

贪婪/非贪婪 实例

➤ `m=re.search(r'<div>.*</div>','aa<div>test1</div>bb<div>test2</div>cc')`

尽可能向右尝试匹配

➤ `m.group(0)`


`<div>test1</div>bb<div>test2</div>`

➤ `m=re.search(r'<div>.*?</div>','aa<div>test1</div>bb<div>test2</div>cc')`

匹配一个后结束匹配

➤ `m.group(0)`

`<div>test1</div>`



```
import re
data='X1Y2051001X2Y2052028X3Y2051052'
n1=re.findall('X(.*)Y',data,re.S)  #提取X、Y之间的内容
n2=re.findall('Y(.*)X2',data,re.S) #提取Y与X2之间的内容
print(n1)
print(n2)
```

```
data='X1Y2051001Z-X2Y2052028Z-X3Y2051052Z'
n3=re.findall('X(.*)Y(.*)Z',data,re.S)
print(n3)
```


实例-抓取title标签间的内容

```
import re
import requests
r= requests.get("http://news.baidu.com/")
r.encoding=r.apparent_encoding
title = re.findall(r'<title>(.*?)</title>', r.text)
print(title[0])
```

百度新闻——全球最大的中文新闻平台

实例-抓取超链接标签间的内容

```
import re
import requests
r= requests.get("http://news.baidu.com/")
r.encoding=r.apparent_encoding
```

#获取完整超链接

```
res = r"<a.*?href=.*?</a>"
urls = re.findall(res, r.text)
for t in urls[:8]:
    print(t)
```

抓取超链接标签的href和内容

➤ 获取href

```
res = r"<a.*?(href=.*?) .*?</a>"
urls = re.findall(res, r.text)
for t in urls[:8]:
    print(t)
```

```
href="https://www.baidu.com/"
href="http://tieba.baidu.com/"
href="https://zhidao.baidu.com/"
href="http://music.baidu.com/"
href="http://image.baidu.com/"
href="http://v.baidu.com/"
href="http://map.baidu.com/"
href="http://wenku.baidu.com/"
```

➤ 获取超链接<a>和之间内容

```
res = r'<a .*?>(.*?)</a>'
texts = re.findall(res, r.text, re.S|re.M)
for t in texts[:8]:
    print(t)
```

网页
贴吧
知道
音乐
图片
视频
地图
文库

抓取图片超链接标签的src

- HTML插入图片使用标签的基本格式为 “”

```
html = "<img...src=“...610077661n.jpg” ...>”
```

```
urls = re.findall('src="(.*?)"', html, re.I|re.S|re.M)
```

```
print(urls)
```

- 若要获取URL中最后一个参数

```
name = str.split(urls[0],'/')[-1]
```

```
print(name)
```

抓取tr\td标签间的内容

#获取<tr></tr>间内容

```
res = r'<tr>(.*?)</tr>'  
texts = re.findall(res, html, re.S|re.M)  
for m in texts:  
    print(m)
```

#获取<th></th>间内容

```
for m in texts:  
    res_th = r'<th>(.*?)</th>'  
    m_th = re.findall(res_th, m, re.S|re.M)  
    for t in m_th:  
        print(t)
```

#直接获取<td></td>间内容

```
res = r'<td>(.*?)</td><td>(.*?)</td>'  
texts = re.findall(res, html, re.S|re.M)  
for m in texts:  
    print(m[0],m[1])
```

```
<html>  
<head><title>表格</title></head>  
<body>  
    <table border=1>  
        <tr><th>学号</th><th>姓名</th></tr>  
        <tr><td>18001</td><td>同舟  
        </td></tr>  
        <tr><td>18002</td><td>共济  
        </td></tr>  
    </table>  
</body>  
</html>
```

BS+正则实现豆瓣图书信息提取

```
import requests
import re
from bs4 import BeautifulSoup
r= requests.get('https://book.douban.com/')
soup = BeautifulSoup(r.text,"html.parser")
all_li=soup.find_all('li')
for li in all_li:
    p='<li.*?cover.*?href="(.*?)".*?title="(.*?)".*?more-  
meta.*?author">(.*?)</span>.*?year">(.*?)</span>.*?</li>'
    results = re.findall(p, str(li), re.I|re.S|re.M)
    for result in results:
        url,name,author,date = result
        author = re.sub('\s',"",author)
        date = re.sub('\s',"",date)
        print(url,name,author,date)
```

个人博客爬取

1. 博客网址的标题 (title) 内容。
2. 爬取博客首页中所有文章的标题、超链接

```
import re
import requests
```

```
url = "http://blog.sina.com.cn/s/articlelist_1191258123_0_1.html"
r=requests.get(url)
r.encoding='utf-8'
title = re.findall(r'<title>(.*?)</title>', r.text) #爬取标题
print(title[0])
```

```
pattern='<a.*?target="_blank"
href="(.*?)">(.*?)</a>'
t1 = re.findall(pattern,r.text, re.S|re.M)
for t in t1:
    print(t[0],t[1])
```

猫眼电影热映口碑榜

#%%电影标题提取

```
import requests
url="https://maoyan.com/board"
headers = {'User-Agent':"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.108 Safari/537.36"}
r = requests.get(url,headers=headers)
Data=re.findall('<p class="name"><a .*?>(.*)</a></p>',r.text,re.S)
print(Data)
```

['金刚川', '野性的呼唤', '疯狂原始人2', '流浪地球：飞跃2020特别版', '热血合唱团', '除暴', '一秒钟', '气·球', '隐形人', '汪汪队立大功之超能救援']

猫眼电影Top100榜

```
import requests
from bs4 import BeautifulSoup
url="https://maoyan.com/board/4?offset=10"
r = requests.get(url,headers=headers)
Data=re.findall('<p
class="name"><a .*?>(.*?)</a></p>',r.text,re.S)
print(Data)
```

['怦然心动', '星际穿越', '千与千寻', '阿甘正传', '触不可及',
'楚门的世界', '寻梦环游记', '辛德勒的名单', '小丑', '摔跤
吧！爸爸']

#向服务器发送请求并获取html源代码的函数

```
def get_page(url):  
    try:  
        headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1;  
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/58.0.3029.110 Safari/537.36'}  
        resp=requests.get(url,headers=headers)  
        resp.raise_for_status()  
        resp.encoding=resp.apparent_encoding  
        return resp.text    #返回网页源代码  
    except Exception as e:  
        print(e)
```

url='https://maoyan.com/board/4?'

get_info1(get_page(url))

```
{ 'rank': '1' }  
{ 'rank': '2' }  
{ 'rank': '3' }  
{ 'rank': '4' }  
{ 'rank': '5' }  
{ 'rank': '6' }  
{ 'rank': '7' }  
{ 'rank': '8' }  
{ 'rank': '9' }  
{ 'rank': '10' }
```

#提取电影的排名信息

```
def get_info1(page):  
    items=re.findall('board-index.*?>(\d+)</i>',page,re.S)  
    for item in items:  
        data={}  
        data['rank']=item  
        print(data)
```

#提取电影的名称

```
def get_info2(page):
```

```
    items=re.findall('board-  
index .*?>(\d+)</i>.*?class="name"><.*?>(.*)</a></p>',  
page,re.S)
```

```
    for item in items:
```

```
        data={}
```

```
        data['rank']=item[0]
```

```
        data['title']=item[1]
```

```
        print(data)
```

#主程序

```
url='https://maoyan.com/board/4?'  
get_info2(get_page(url))
```

```
{'rank': '1', 'title': '我不是药神'}  
{'rank': '2', 'title': '肖申克的救赎'}  
{'rank': '3', 'title': '绿皮书'}  
{'rank': '4', 'title': '海上钢琴师'}  
{'rank': '5', 'title': '哪吒之魔童降世'}  
{'rank': '6', 'title': '小偷家族'}  
{'rank': '7', 'title': '霸王别姬'}  
{'rank': '8', 'title': '美丽人生'}  
{'rank': '9', 'title': '盗梦空间'}  
{'rank': '10', 'title': '这个杀手不太冷'}
```

#提取主演的名称

```
def get_info3(page):  
    items=re.findall('board-  
index .*?>(\d+)</i>.*?class="name"><.*?>(.*?)</a></p>.*?<p  
class="star">.*?' '主演: (.*?) .*?</p>',page,re.S)
```

```
    for item in items:
```

```
        data={}
```

```
        data['rank']=item[0]
```

```
        data['title']=item[1]
```

```
        actors=re.sub('\n',"",item[2])
```

```
        data['actors']=actors
```

```
        print(data)
```

```
    ...: get_info3(get_page(url))  
{'rank': '1', 'title': '我不是药神', 'actors': '徐峥,周一围,王传君'}  
{'rank': '2', 'title': '肖申克的救赎', 'actors': '蒂姆·罗宾斯,摩根·弗里曼,  
鲍勃·冈顿'}  
{'rank': '3', 'title': '绿皮书', 'actors': '维果·莫腾森,马赫沙拉·阿里,琳达  
·卡德里尼'}  
{'rank': '4', 'title': '海上钢琴师', 'actors': '蒂姆·罗斯,比尔·努恩,克兰伦  
斯·威廉姆斯三世'}  
{'rank': '5', 'title': '哪吒之魔童降世', 'actors': '吕艳婷,囡森瑟夫,瀚墨'}  
{'rank': '6', 'title': '小偷家族', 'actors': '中川雅也,安藤樱,松冈茉优'}  
{'rank': '7', 'title': '霸王别姬', 'actors': '张国荣,张丰毅,巩俐'}  
{'rank': '8', 'title': '美丽人生', 'actors': '罗伯托·贝尼尼,朱斯蒂诺·杜拉  
诺,赛尔乔·比尼·布斯特里克'}  
{'rank': '9', 'title': '盗梦空间', 'actors': '莱昂纳多·迪卡普里奥,渡边谦,约  
瑟夫·高登-莱维特'}  
{'rank': '10', 'title': '这个杀手不太冷', 'actors': '让·雷诺,加里·奥德曼,  
娜塔莉·波特曼'}
```

#主程序

```
url='https://maoyan.com/board/4?'  
get_info3(get_page(url))
```

```

def get_info(page):
    items=re.findall( 'board-
index .*?>(\d+)</i>.*?class= "name" ><.*?>(.*)</a></p>.*?<p
class= "star" >.*? 主演: (.*) .*?</p>.*?<p
class="releasetime">(.*)</p>.*?<p class="score"><i class="integer">
(.*)</i><i class="fraction">(\d+)</i></p>',page,re.S)
    page_all=[]
    for item in items:
        data={}
        data['rank']=item[0]
        data['title']=item[1]
        actors=re.sub('\n',"",item[2])
        data['actors']=actors
        data['date']=item[3]
        data['score']=str(item[4])+str(item[5])
        print(data)
        page_all.append(data)
    return page_all

```

```

In [297]: get_info(get_page(url))
{'rank': '1', 'title': '我不是药神', 'actors': '徐峥,周一围,王传君',
'date': '上映时间: 2018-07-05', 'score': '9.6'}
{'rank': '2', 'title': '肖申克的救赎', 'actors': '蒂姆·罗宾斯,摩根·弗里曼,
鲍勃·冈顿', 'date': '上映时间: 1994-09-10(加拿大)', 'score': '9.5'}
{'rank': '3', 'title': '绿皮书', 'actors': '维果·莫腾森,马赫沙拉·阿里,琳达
·卡德里尼', 'date': '上映时间: 2019-03-01', 'score': '9.5'}
{'rank': '4', 'title': '海上钢琴师', 'actors': '蒂姆·罗斯,比尔·努恩,克兰伦
斯·威廉姆斯三世', 'date': '上映时间: 2019-11-15', 'score': '9.3'}
{'rank': '5', 'title': '哪吒之魔童降世', 'actors': '吕艳婷,囡森瑟夫,瀚墨',
'date': '上映时间: 2019-07-26', 'score': '9.6'}
{'rank': '6', 'title': '小偷家族', 'actors': '中川雅也,安藤樱,松冈茉优',
'date': '上映时间: 2018-08-03', 'score': '8.1'}
{'rank': '7', 'title': '霸王别姬', 'actors': '张国荣,张丰毅,巩俐',
'date': '上映时间: 1993-07-26', 'score': '9.5'}

```

#主程序

```

url='https://maoyan.com/board/4?'
get_info(get_page(url))

```

#获取不同页面网址

```
urls=['https://maoyan.com/board/4?offset={}'.format(i*10) for i in range(10)]
```

#主程序

```
All=[]
```

```
for url in urls:
```

```
    page=get_page(url)
```

```
    datas=get_info(page)
```

```
    All.append(datas)
```

```
print(All)
```

```
[[{'rank': '1',  
  'title': '我不是药神',  
  'actors': '徐峥,周一围,王传君',  
  'date': '上映时间: 2018-07-05',  
  'score': '9.6'},  
 {'rank': '2',  
  'title': '肖申克的救赎',  
  'actors': '蒂姆·罗宾斯,摩根·弗里曼,鲍勃·冈顿',  
  'date': '上映时间: 1994-09-10(加拿大)',  
  'score': '9.5'},  
 {'rank': '3',  
  'title': '绿皮书',  
  'actors': '维果·莫腾森,马赫沙拉·阿里,琳达·卡德琳',  
  'date': '上映时间: 2019-03-01',  
  'score': '9.5'},  
 {'rank': '4',  
  'title': '海上钢琴师',  
  'actors': '蒂姆·罗斯,比尔·努恩,克兰伦斯·威廉姆',  
  'date': '上映时间: 2019-11-15',  
  'score': '9.3'},  
 ]]
```

诗词名句网排行榜 (https://www.shicimingju.com/paiming)



诗词排行榜

1 《将进酒·君不见黄河之水天上来》

[唐] 李白
君不见黄河之水天上来，奔流到海不复回。
君不见高堂明镜悲白发，朝如青丝暮成雪。

[展开全文](#)



2 《江城子·乙卯正月二十日夜记梦》

[宋] 苏轼
十年生死两茫茫，不思量，自难忘。千里孤坟，无处话凄凉。纵使相逢应不识，尘满面，鬓如霜。
夜来幽梦忽还乡，小轩窗，正梳妆。相顾无言，惟有泪千行。料得年年肠断处，明月夜，短松冈。



3 《声声慢·寻寻觅觅》

[宋] 李清照
寻寻觅觅，冷冷清清，凄凄惨惨戚戚。
乍暖还寒时候，最难将息。

[展开全文](#)



4 《青玉案·元夕》

[宋] 辛弃疾
东风夜放花千树，更吹落，星如雨。

[展开全文](#)

推荐作者

李白	苏轼	诗经
毛泽东	李清照	杜甫
辛弃疾	李煜	白居易
王维	李商隐	纳兰性德
陆游	柳永	陶渊明
杜牧	曹雪芹	曹操
屈原	李贺	孟浩然
欧阳修	刘禹锡	王安石
秦观	唐寅	晏殊
鲁迅	曹植	温庭筠
王昌龄	范仲淹	王勃
杨万里	张九龄	仓央嘉措
岑参	吴几道	元稹
柳宗元	韩愈	岳飞
骆宾王	鱼玄机	贺知章
范成大	姜夔	张若虚
黄庭坚	文天祥	韦应物
李世民	王国维	王之涣
朱淑真	周邦彦	龚自珍
周思来	元好问	罗隐

```

<div class="card shici_card">
<div class="list_num_info">
<span>1</span><br/>[唐]<br/><a href="/chaxun/zuozhe/1.html">李白</a>
</div>
<div class="shici_list_main">
<h3><a href="/chaxun/list/25739.html">《将进酒·君不见黄河之水天上来》</a> </h3>
<div class="shici_content">
君不见黄河之水天上来，奔流到海不复回。<br />君不见高堂明镜悲白发，朝如青丝暮成雪。<br />
<div style="display: none">
人生得意须尽欢，莫使金樽空对月。<br />天生我材必有用，千金散尽还复来。<br />烹羊宰牛且为乐，
<a href="javascript:void(0)" class="hide_more_shici">收起</a>
</div>
</div>
<div class="shici_list_pic">
<a href="/chaxun/list/25739.html"></a>
</div>

```


引入所需模块/库

```
import requests
from bs4 import BeautifulSoup
import os
import re
import matplotlib.pyplot as plt
plt.rcParams['font.sans-serif'] = ['SimHei']
```

向url地址发送请求并获取源代码

```
def get_page(url):
    try:
        headers = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110
Safari/537.36'}
        r = requests.get(url, headers=headers, verify=False)
        r.raise_for_status()
        r.encoding = r.apparent_encoding
        return r.text # 返回网页源代码
    except Exception as e:
        print(e)
```


根据soup获取、显示和保存网页相关数据

```
def get_info(soup):
    all_divs = soup.find_all('div', class_='card shici_card')
    for a in all_divs:
        # 排名
        info = a.find('div', class_='list_num_info')
        no = info.find('span').get_text()
        print("第{}名:".format(no), end="")
        # 标题
        titles = a.h3.get_text()
        print("{}".format(titles))
        # 朝代
        time = re.search('<br/?>(.*?)<br/?>', str(info), re.S | re.M).group(1)
        print("朝代: ", time[1:-1]) # 去除朝代两端的中括号
        if time[1:-1] != "": # 判断朝代信息是否为空
            dynasty.append(time[1:-1])
        else:
            print("第{}名无朝代".format(no), end="")
        # 作者
        name = info.find('a').get_text().strip()
        print("诗人: {}".format(name))
```

未完待续...

接上页函数...

图片保存

```
picDiv = a.find('div', class_='shici_list_pic')
if not picDiv:
    print("无图片")
else:
    pic_url = picDiv.find('img').get("src")
    root = r"c:/t/"    #所有图片存放在C:/t/目录下
    path = root + str(no) + '.' + pic_url.split('.')[1]
    try:
        if not os.path.exists(root):
            os.mkdir(root)
        if not os.path.exists(path):
            r = requests.get("https://www.shicimingju.com/{}".format(pic_url))
            r.raise_for_status()
            with open(path, 'wb') as f:
                f.write(r.content)
                print("文件保存成功")
        else:
            print("文件已存在")
    except:
        print("产生异常")
return dynasty
```

统计各朝代的诗词数量、绘图

统计各朝代的诗词数量、绘图

```
def disp(dynasty):
```

```
    # 统计
```

```
    dyn_n = []
```

```
    count_n = []
```

```
    for who in set(dynasty):
```

```
        dyn_n.append(who)
```

```
        count_n.append(dynasty.count(who))
```

```
        print(who, ': ', dynasty.count(who))
```

```
    # 绘制
```

```
    plt.figure()
```

```
    plt.title('各朝代诗词')
```

```
    plt.xlabel('朝代')
```

```
    plt.ylabel('诗词数量')
```

```
    plt.bar(x=dyn_n, height=count_n, width=0.8)
```

```
    for i, j in zip(dyn_n, count_n):
```

```
        plt.text(i, j, j, va='bottom', ha='center')
```

主程序

#主程序

```
dynasty = [] #存放2000首诗词的朝代数据
```

```
for num in range(1,101):
```

```
    url = 'https://www.shicimingju.com/paiming?p=' + str(num)    #生成每页的url地址
```

```
    content = get_page(url) #获取每页的源代码
```

```
    if content:
```

```
        soup = BeautifulSoup(content,'html.parser') #对每页源代码进行解析
```

```
        get_info(soup) #获取、显示、保存指定的数据
```

```
    disp(dynasty) #根据朝代进行统计、绘制图形
```

1.jpg
JPG 文件
143 KB

6.jpg
JPG 文件
71.3 KB

11.jpg
JPG 文件
170 KB

16.jpg
JPG 文件
49.1 KB

2.jpg
JPG 文件
108 KB

7.jpg
JPG 文件
162 KB

12.jpg
JPG 文件
184 KB

19.jpg
JPG 文件
94.6 KB

3.jpg
JPG 文件
200 KB

8.jpg
JPG 文件
230 KB

13.png
PNG 文件
316 KB

20.jpg
JPG 文件
52.3 KB

5.jpg
JPG 文件
147 KB

9.jpg
JPG 文件
306 KB

14.jpg
JPG 文件
125 KB

23.jpg
JPG 文件
112 KB

