

python爬虫 基础

▶ 1, 基础知识 40

▼ 2, 正则表达式与Cookie的使用

▼ 1, 基础知识

import re正则表达式引用

▼ (1) 普通字符作为原子

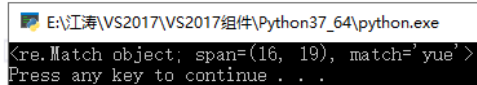
```
import urllib.request
import urllib.error
import re
```

```
pattern="yue"#普通字符串作为原子
string="http://yum.iqianyue.com"
result1=re.search(pattern,string)
print(result1)
#pattern在string中的位置, 若没有显示none
```

■

```
import urllib.request
import urllib.error
import re

pattern="yue"#普通字符串作为原子
string="http://yum.iqianyue.com"
result1=re.search(pattern,string)
print(result1)#pattern在string中的位置, 若没有显示none
```



```
E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(16, 19), match='yue'>
Press any key to continue . . .
```

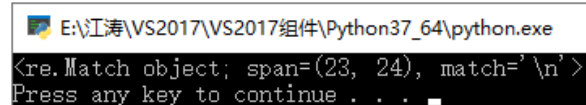
▼ (2) 非打印字符作为原子

/n 用于匹配一个换行符
/t 用于匹配一个制表符

```
pattern=("\\n")#非打印字符作为原子
string="http://yum.iqianyue.com
http://baidu.com"
result1=re.search(pattern,string)
print(result1)
```

■

```
pattern=("\\n")#非打印字符作为原子
string='''http://yum.iqianyue.com
http://baidu.com'''
result1=re.search(pattern,string)
print(result1)
```



```
E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(23, 24), match='\\n'>
Press any key to continue . . .
```

- ▼ (3) 通用字符作为原子
"."除换行符(\n)外任意字符

表5-2 常见的通用字符及其含义

符 号	含 义
\w	匹配任意一个字母、数字或下划线
\W	匹配除字母、数字和下划线以外的任意一个字符
\d	匹配任意一个十进制数
\D	匹配除十进制数以外的任意一个其他字符
\s	匹配任意一个空白字符
\S	匹配除空白字符以外的任意一个其他字符

- ▼

```
pattern = "\w\dpython\w"
string = "ddfdasdfasdf1212131pythond_ddd1"
result = re.search(pattern, string)
print(result)
```

■

```
pattern = "\w\dpython\w"
string = "ddfdasdfasdf1212131pythond_ddd1"
result = re.search(pattern, string)
print(result)
```

```
E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(17, 26), match='31pythond'>
Press any key to continue . . .
```

- ▼ (4) 原子表
原子表由[]表示，比如[xyz]就是一个原子表，这个原子表中定义了3个原子，这3个原子的地位平等，如，我们定义的正则表达式为"[xyz]py"，对应的源字符串是"xpython"，如果此时使用 re.search() 函数进行匹配，就可以匹配出结果"xy"，因为此时只要 py 前一位是 x y z 字母中的其中一个字母，就可以匹配成功。类似的，[^]代表的是除了中括号里面的原子均可以匹配，比如"[^xyz]py"能匹配"apy"，但是却不能匹配"xy"等。

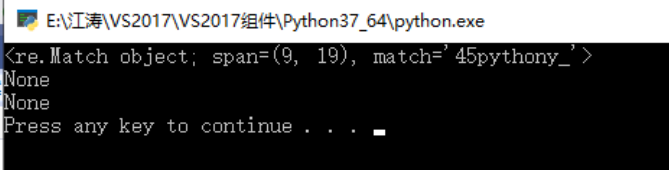
- ▼

```
pattern1 = "\w\dpython[xyz]\w"
pattern2 = "\w\dpython[^xyz]\w"
pattern3 = "\w\dpython[xyz]\W"
string = "abcdfphp345pythony_py"
result1 = re.search(pattern1, string)
result2 = re.search(pattern2, string)
result3 = re.search(pattern3, string)
print(result1)
print(result2)
print(result3)
```

```

pattern1="\w\dpython[xyz]\w"
pattern2="\w\dpython[^xyz]\w"
pattern3="\w\dpython[xyz]\w"
string="abcdfphp345pythony_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string)
result3=re.search(pattern3,string)
print(result1)
print(result2)
print(result3)

```



▼ (5) 元字符

表5-3 常见的元字符及其含义

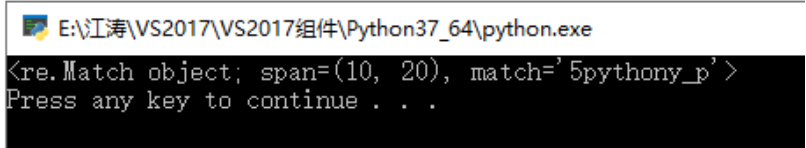
符 号	含 义
.	匹配除换行符以外的任意字符
^	匹配字符串的开始位置
\$	匹配字符串的结束位置
*	匹配 0 次、1 次或多次前面的原子
?	匹配 0 次或 1 次前面的原子
+	匹配 1 次或多次前面的原子
{n}	前面的原子恰好出现 n 次
{n,}	前面的原子至少出现 n 次
{n, m}	前面的原子至少出现 n 次, 至多出现 m 次
	模式选择符
()	模式单元符

▪ (1) 任意匹配元字符

```

import re
pattern=".python..."
string="abcdfphp345pythony_py"
result1=re.search(pattern,string)
print(result1)

```



▼ (2) 边界限制元字符

```

pattern1="^abd" #边界限制元字符
pattern2="^abc"
pattern3="py$"
pattern4="ay$"
string="abcdfphp345pythony_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string)
result3=re.search(pattern3,string)
result4=re.search(pattern4,string)
print(result1)
print(result2)
print(result3)
print(result4)

```

```

pattern1="^abd" #边界限制元字符
pattern2="^abc"
pattern3="py$"
pattern4="ay$"
string="abcdphp345python_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string)
result3=re.search(pattern3,string)
result4=re.search(pattern4,string)
print(result1)
print(result2)
print(result3)
print(result4)

```

```

E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
None
<re.Match object; span=(0, 3), match='abc'>
<re.Match object; span=(19, 21), match='py'>
None
Press any key to continue . . .

```

▼ (3) 限定符

```

pattern1="py.*n"
pattern2="cd{2}"
pattern3="cd{3}"
pattern4="cd{2,}"
string="abcdphp345python_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string)
result3=re.search(pattern3,string)
result4=re.search(pattern4,string)
print(result1)
print(result2)
print(result3)
print(result4)

```

```

pattern1="py.*n"
pattern2="cd{2}"
pattern3="cd{3}"
pattern4="cd{2,}"
string="abcdphp345python_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string)
result3=re.search(pattern3,string)
result4=re.search(pattern4,string)
print(result1)
print(result2)
print(result3)
print(result4)

```

```

E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(13, 19), match='python'>
<re.Match object; span=(2, 5), match='cdd'>
<re.Match object; span=(2, 6), match='cddd'>
<re.Match object; span=(2, 6), match='cddd'>
Press any key to continue . . .

```

▼ (4) 模式选择符 "|"

正则表达式“python|php”中，字符串“python”和“php”均满足匹配条件第一个满足的就输出结果。

```

import re
pattern="python|php" #模式选择符
string="abpythoncdphp345y_py"
result1=re.search(pattern,string)
print(result1)

```

▪

```
pattern="python|php" #模式选择符
string="abpythoncdfphp345y_py"
result1=re.search(pattern,string)
print(result1)
```

```
E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(2, 8), match='python'>
Press any key to continue . . .
```

▼ (5) 模式单元符“() ”
字符整体不分割 (ab)

```
import re
pattern1="(cd){1,}"
pattern2="cd{1,}"
string="abcdcdcdcdfphp345pythony_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string)
print(result1)
print(result2)
```

▪

```
import re
pattern1="(cd){1,}"
pattern2="cd{1,}"
string="abcdcdcdcdfphp345pythony_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string)
print(result1)
print(result2)
```

```
E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(2, 10), match='cdcdcdcd'>
<re.Match object; span=(2, 4), match='cd'>
Press any key to continue . . .
```

▼ (6) 可选字符集
[.com|.cn] .com或.cn中有一个满足
[a-zA-Z] [0-5 7-9]

```
import re
pattern="[a-zA-Z]+://[^\s]*[.com|.cn]"
string="<a href='http://www.baidu.com'>百度首页</a>"
result=re.search(pattern,string)
print(result)
```

输出：

```
<re.Match object; span=(9, 29), match='http://www.baidu.com'>
Press any key to continue . . .
```

```
import re
pattern="[a-zA-Z]+://[^\s]*[.com|.cn]"
string="<a href='http://www.baidu.com'>百度首页</a>"
result=re.search(pattern,string)
print(result)
```

```
E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(9, 29), match='http://www.baidu.com'>
Press any key to continue . . .
```

▼ (6)模式修正

符 号	含 义
I	匹配时忽略大小写
M	多行匹配
L	做本地化识别匹配
U	根据 Unicode 字符及解析字符
S	让 . 匹配包括换行符，即用了该模式修正后，"." 匹配就可以匹配任意的字符了

```
▼ import re #不区分大小写
pattern1="python"
pattern2="python"
string="abcdfphp345Pythony_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string,re.I)
print(result1)
print(result2)
```

```
import re #不区分大小写
pattern1="python"
pattern2="python"
string="abcdfphp345Pythony_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string,re.I)
print(result1)
print(result2)
```

```
E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
None
<re.Match object; span=(11, 17), match='Python'>
Press any key to continue . . .
```

▼ (7) 贪婪模式与懒惰模式

总的来说，贪婪模式的核心点就是尽可能多地匹配，而懒惰模式的核心点就是尽可能少地匹配。

▼ 转化为懒惰模式，需要后面加上“？”，方可转化为懒惰模式。

例：

表达式 .* 就是单个字符匹配任意次,即贪婪匹配。

表达式 .*? 是满足条件的情况只匹配一次,即最小匹配。

```

▼ import re
pattern1="p.*y"#贪婪模式
pattern2="p.*?y"#懒惰模式
string="abcdphp345python_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string)
print(result1)
print(result2)

```

```

import re
pattern1="p.*y"#贪婪模式
pattern2="p.*?y"#懒惰模式
string="abcdphp345python_py"
result1=re.search(pattern1,string)
result2=re.search(pattern2,string)
print(result1)
print(result2)

```

```

E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(5, 21), match='php345python_py'>
<re.Match object; span=(5, 13), match='php345py'>
Press any key to continue . . .

```

▼ (8) 正则表达式常见函数

▼ 1.re.match () 从头匹配函数

re.match(pattern,string,flag)

第一个参数代表对应的正确表达式，第二个参数代表对应的源字符，第三个参数是可选参数，代表对应的标志位，可以放模式修正符 等信息

函数从头开始匹配，若第一个字符不符合输出none

```

import re
string="apythonhellomypythonhispythonourpythonend"
pattern=".python."
result=re.match(pattern,string)
print(result)

```

```

E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(0, 8), match=' apythonh'>
Press any key to continue . . .

```

```

import re
string="1apythonhellomypythonhispythonourpythonend"
pattern=".python."
result=re.match(pattern,string)
print(result)

```

```

E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
None
Press any key to continue . . .

```

▼ 2.re.search () 整个字符串函数

re.search(pattern,string,flag)

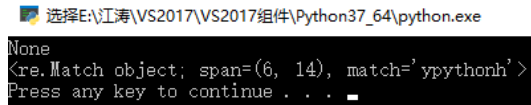
第一个参数代表对应的正确表达式，第二个参数代表对应的源字符串，第三个参数是可选参数，代表对应的标志位，可以放模式修正符 等信息

re.search () 函数进行匹配，使用该函数进行匹配，会扫描整个字符串并进行对应的匹配。该函数与 re.match () 函数最大的不同是，re.match () 函数从源字符串的开头 进行匹配，而re.search () 函数会在全文中进行检索并匹配。

▼ import re string="hellomypythonhispythonourpythonend" pattern=".python."
result=re.match(pattern,string) result2=re.search(pattern,string) print(result)
print(result2)

■

```
import re
string="hellomypythonhispythonourpythonend"
pattern=".python."
result=re.match(pattern,string)
result2=re.search(pattern,string)
print(result)
print(result2)
```



```
选择E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
None
<re.Match object; span=(6, 14), match='ypythonh'>
Press any key to continue . . .
```

▼ 3.全局匹配函数

处理字符串内多个符合要求的结果

▼ 思路如下：

- 1) 使用re.compile () 对正则表达式进行预编译。
- 2) 编译后，使用findall () 根据正则表达式从源字符串中将匹配的结果全部找出。

▼ import re
string="hellomypythonhispythonourpythonend"
pattern=re.compile(".python.")#预编译
result=pattern.findall(string)#找出符合模式的所有结果
print(result)

■

```
import re
string="hellomypythonhispythonourpythonend"
pattern=re.compile(".python.")#预编译
result=pattern.findall(string)#找出符合模式的所有结果
print(result)
```



```
E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
['ypythonh', 'spythono', 'rpythone']
Press any key to continue . . .
```

▼ import re #整合
string="hellomypythonhispythonourpythonend"
pattern=".python."
result=re.compile(pattern).findall(string)
print(result)


```
import re #整合
string="hellomypythonhispythonourpythonend"
pattern=".python."
result=re.compile(pattern).findall(string)
print(result)
```

```
选择E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
['ypythonh', 'spythono', 'rpythone']
Press any key to continue . . .
```

▼ 4.re.sub () 匹配并替换函数 re.sub(pattern,rep,string,max)

其中，第一个参数为对应的正则表达式，第二个参数为要替换成的字符串，第三个参数为源字符串，第四个参数为可选项，代表最多 替换的次数，如果忽略不写，则会将符合模式的结果全部替换。使用re.sub () 这个函数，会根据正则表达式pattern，从源字符串string查找出符合模式的结果，并替换为字符串rep，最多可替换 max次

```
▼ 用php 代替python字符
import re
string="hellomypythonhispythonourpythonend"
pattern="python."
result1=re.sub(pattern,"php",string) #全部替换
result2=re.sub(pattern,"php",string,2) #最多替换两次
print(result1)
print(result2)
```

输出：
hellomyphpisphpurphpnd
hellomyphpisphpurpythonend
Press any key to continue . . .

```
#print(result)

import re
string="hellomypythonhispythonourpythonend"
pattern="python."
result1=re.sub(pattern,"php",string) #全部替换
result2=re.sub(pattern,"php",string,2) #最多替换两次
print(result1)
print(result2)
```

```
选择E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
hellomyphpisphpurphpnd
hellomyphpisphpurpythonend
Press any key to continue . . .
```

▼ (9) 实例解析

▼ 实例1：匹配.com或.cn后缀的URL网址

实例目的：将一串字符串里面以.com或.cn为域名后缀的URL网 址匹配出来，过滤掉其他的无关信息。

```

▼ import re
pattern="[a-zA-Z]+://[^\s]*[.com|.cn]"
string="<a href='http://www.baidu.com'>百度首页</a>"
result=re.search(pattern,string)
print(result)

```

输出：

```

<re.Match object; span=(9, 29), match='http://www.baidu.com'>
Press any key to continue . . .

```

■

```

import re
pattern="[a-zA-Z]+://[^\s]*[.com|.cn]"
string="<a href='http://www.baidu.com'>百度首页</a>"
result=re.search(pattern,string)
print(result)

```

```

E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(9, 29), match='http://www.baidu.com'>
Press any key to continue . . .

```

▼ 实例2：匹配电话号码

实例目的：将一串字符串里面出现的电话号码信息提取出来，过滤掉其他无关信息。

```

▼ import re #匹配电话号码
pattern="\d{4}-\d{7}|\d{3}-\d{8}"
string="<a href='http://www.bai92698876g033-45645677ggg'"
result=re.search(pattern,string)
print(result)

```

▼

```

import re #匹配电话号码
pattern="\d{4}-\d{7}|\d{3}-\d{8}"
string="<a href='http://www.baidu.com'>百度首页</a>92698876g033-45645677ggg"
result=re.search(pattern,string)
print(result)

```

```

E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
<re.Match object; span=(48, 60), match='033-45645677'>
Press any key to continue . . .

```

■

```

import re #匹配电话号码
pattern="\d{4}-\d{7}|\d{3}-\d{8}"
string="<a href='http:// www.baidu.com'>百度首页</a>032-92698876g033-45645677ggg"
result=re.compile(pattern).findall(string)
print(result)

```

```

E:\江涛\VS2017\VS2017组件\Python37_64\python.exe
['032-92698876', '033-45645677']
Press any key to continue . . .

```

▼ 实例3：匹配电子邮件地址

实例目的：将一串字符串里面出现的电子邮件信息提取出来，过滤掉其他无关信息。



```
import re
pattern="\w+([.-]\w+)*@\w+([.-]\w+)*\.\w+([.-]\w+)*" #匹配电子邮件的正则表达式
string="<a href='http://a><br><a href='mailto:c-e+o@iqi-anyue.com.cn'>电子邮件地址</a>"
result=re.search(pattern,string)
print(result)
```



```
#print(result)

import re
pattern="\w+([.-]\w+)*@\w+([.-]\w+)*\.\w+([.-]\w+)*" #匹配电子邮件的正则表达式
string="<a href='http:// www.baidu.com'>百度首页</a><br><a href='mailto:c-e+o@iqi-anyue.com.cn'>电子邮件地址</a>"
result=re.search(pattern,string)
print(result)
```

▼ 2, Cookie

▼ 1 , http.cookiejar

如果希望登录状态一直保持，则需要进行Cookie处理。 进行Cookie处理的一种常用思路如下：

- 1) 导入Cookie处理模块http.cookiejar。
- 2) 使用http.cookiejar.CookieJar () 创建CookieJar对象。
- 3) 使用HTTPCookieProcessor创建cookie处理器，并以其为参数 构建opener对象。
- 4) 创建全局默认的opener对象

▼ #cookie使用

```
import urllib.request
import urllib.parse
import http.cookiejar
url = "http:// bbs.chinaunix.net/member.php?
mod=logging&action=login&loginsubmit =yes&loginhash=L768q"
postdata =urllib.parse.urlencode({ "username":"weisuen", "password":"aA123456"
}).encode('utf-8')
req = urllib.request.Request(url,postdata)
req.add_header('User-Agent', 'Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/ 537.36 (KHTML, like Gecko) Chrome/38.0.2125.122 Safari/537.36
SE 2.X MetaSr 1.0')
#使用http.cookiejar.CookieJar()创建CookieJar对象
cjar=http.cookiejar.CookieJar()
#使用HTTPCookieProcessor创建cookie处理器，并以其为参数构建opener对象
opener = urllib.request.build_opener(urllib.request.HTTPCookieProcessor(cjar))
#将opener安装为全局
urllib.request.install_opener(opener)
file=opener.open(req)
data=file.read()
file=open("D:/Python10.html","wb")
file.write(data)
file.close()
url2="http:// bbs.chinaunix.net/"
data2=urllib.request.urlopen(url2).read()
fhandle=open("D:/Python11.html","wb")
fhandle.write(data2)
fhandle.close()
```

▼

```
#cookie使用
import urllib.request
import urllib.parse
import http.cookiejar
#url地址
url = "http:// bbs.chinaunix.net/member.php?mod=logging&action=login&loginsubmit =yes&loginhash=L768q"
#post 名称 密码
postdata =urllib.parse.urlencode({ "username":"weisuen", "password":"aA123456" }).encode('utf-8')
#组合成Request
req = urllib.request.Request(url,postdata)
#增加头文件，模拟访问
req.add_header('User-Agent', 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/ 537.36 (KHTML, like Gecko) Chrome/38.0.2125.122 Safari/537.36 SE 2.X MetaSr 1.0')
#使用http.cookiejar.CookieJar()创建CookieJar对象
cjar=http.cookiejar.CookieJar()
#使用HTTPCookieProcessor创建cookie处理器，并以其为参数构建opener对象
opener = urllib.request.build_opener(urllib.request.HTTPCookieProcessor(cjar))
#将opener安装为全局
urllib.request.install_opener(opener)
file=opener.open(req)
data=file.read()
file=open("D:/Python10.html","wb")
file.write(data)
file.close()
url2="http:// bbs.chinaunix.net/"
data2=urllib.request.urlopen(url2).read()
fhandle=open("D:/Python11.html","wb")
fhandle.write(data2)
fhandle.close()
```

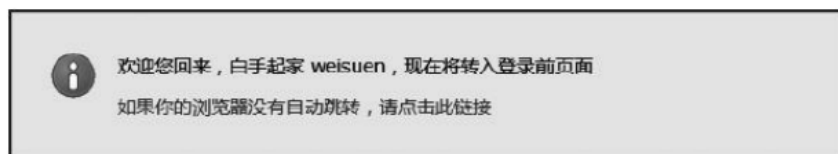


图5-9 登录成功



图5-10 保持爬虫在对应站点的登录状态

- ▶ 3,手写Python爬虫 16
- 4, fiddler 监听软件使用
- ▶ 5, 爬虫伪装 17
- ▶ 6,爬虫的定向爬取技术 4
- ▶ 7, scrapy基础 62
- ▼ 8, scrapy
CrawlSpider (自动爬取网页) 实例

extract()==getall()

▪