# 正则表达式re模块

正则表达式是用来简洁表达一组字符串的表达式。

## re模块

re库是Python的标准库，主要用于字符串匹配。

re.search() 在一个字符串中搜索匹配正则表达式的第一个位置，返回match对象

re.match() 从一个字符串的开始位置起匹配正则表达式，返回match对象

re.findall() 搜索字符串，以列表类型返回全部能匹配的子串

re.split() 将一个字符串按照正则表达式匹配结果进行分割，返回列表类型

re.finditer() 搜索字符串，返回一个匹配结果的迭代类型，每个迭代元素是match对象

re.sub() 在一个字符串中替换所有匹配正则表达式的子串，返回替换后的字符串

（1）、 match函数

语法：re.match(pattern, string, flags=0)

参数：

pattern : 正则表达式的字符串或原生字符串表示

string : 待匹配字符串

flags : 正则表达式使用时的控制标记

re.I：忽略大小写

re.L：表示特殊字符集 \w, \W, \b, \B, \s, \S 依赖于当前环境

re.M：多行模式

re.S：'.'并且包括换行符在内的任意字符（注意：'.'不包括换行符）

re.U： 表示特殊字符集 \w, \W, \b, \B, \d, \D, \s, \S 依赖于 Unicode 字符属性数据库

实例：match.py

**import** re  
  
m = re.match(**r'hello'**, **'yeah helloworld'**)  
**if** m:  
 print(type(m))  
 print(m.group())  
**else**:  
 print(**'no match'**)  
  
n = re.match(**r'hello'**, **'helloworld'**)  
**if** n:  
 print(type(n))  
 print(n.group())  
**else**:  
 print(**'no match'**)

**注意：文件名不能命名为re.py，否则会和re模块冲突，因为它会把你自己的re.py文件当做re模块。**

（2）、 search函数

函数原型为： re.search(pattern, string, flags)   
会扫描整个字符串并返回第一个成功的匹配。

实例：search.py

**import** re  
  
n = re.search(**r'hello'**, **'yeah helloworld'**)  
**if** n:  
 print(type(n))  
 print(n.group())  
**else**:  
 print(**'no result'**)

（3）、findall函数

实例：findall.py

import re

ls = re.findall(r'[1-9]\d{5}', 'BIT100081 TSU100084')

ls

['100081', '100084']

[1-9]表示1-9中任意一个字符。\d表示一个数字。\d{5}表示5个连续的数字。

（4）、split函数

re.split(pattern, string, maxsplit=0, flags=0)

实例：split.py

import re  
  
url = 'http://localhost/test.py?a=hello&b=world'  
values = re.split(r'[?]',url)  
print(values[1])  
  
lst = re.split(r'[1-9]\d{5}', 'BIT100081 TSU100084CHI')  
print(lst)  
  
lst = re.split(r'[1-9]\d{5}', 'BIT100081 TSU100084CHI', maxsplit=1)  
print(lst)

（5）、sub函数  
在一个字符串中替换所有匹配正则表达式的子串, 返回替换后的字符串.

re.sub(pattern, repl, string, count=0, flags=0)

实例：sub.py

import re

info = re.sub(r'[1-9]\d{5}', ':zipcode', 'BIT100081 TSU100084')

print(info)

输出：

BIT:zipcode TSU:zipcode

## 常用规则

正则表达式常用的规则:

| **语法** | **意义** | **说明** |
| --- | --- | --- |
| . | 任意字符 |  |
| ^ | 字符串开始 | ‘^hello’匹配’helloworld’而不匹配’aaaahellobbb’ |
| $ | 字符串结尾 | abc$表示abc且在一个字符串的结尾 |
| \* | 0 个或多个字符（贪婪匹配） | <\*>匹配<\title>chinaunix<\title> |
| + | 1 个或多个字符（贪婪匹配） | 与上同理 |
| ? | 0 个或多个字符（贪婪匹配） | 与上同理 |
| \*?  +?  ?? | 以上三个取第一个匹配结果（非贪婪匹配） | <\*>匹配<\title> |
| {m,n} | 对于前一个字符重复m到n次，{m}亦可 | a{6}匹配6个a、a{2,4}匹配2到4个a |
| {m,n}? | 对于前一个字符重复m到n次，并取尽可能少 | ‘aaaaaa’中a{2,4}只会匹配2个 |
| [] | 表示一个字符集 | [0-9]、[a-z]、[A-Z]、[^0]  [abc]表示a、b、c，  [^abc]表示非a或b或c的单个字符 |
| (…) | 匹配括号中任意表达式 |  |
| (?<=…) | 前向界定 | 括号中 ’…’ 代表你希望匹配的字符串的前面应该出现的字符串。 |
| (?=…) | 后向界定 | 括号中的 ’…’ 代表你希望匹配的字符串后面应该出现的字符串。 |
| \d | 数字，等价于[0‐9] |  |
| \w | 单词字符，等价于[A‐Za‐z0‐9\_] |  |
| \b | 空格字符 |  |

## 贪婪格式和非贪婪格式

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，而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。  
实例：mode.py

import re  
a = re.match('<(.\*)>','<H1>title<H1>').group()

print(a)

b = re.match('<(.\*?)>','<H1>title<H1>').group()

print(b)

执行结果：

<H1>title<H1>

<H1>

## 原生字符串

import re

# “\b”在ASCII 字符中代表退格键，\b”在正则表达式中代表“匹配一个单词边界”

print(re.findall("\bblow", "Jason blow cat")) # 这里\b代表退格键,所以没有匹配到

print(re.findall("\\bblow", "jason blow cat")) # 用\转义后这里就匹配到了 ['blow']

print(re.findall(r"\bblow", "jason blow cat")) # 用原生字符串后就**不需要转义**了 ['blow']

#执行结果：

[]

['blow']

['blow']

## 爬虫程序中利用re分析HTML标签常用技巧

（1）、获取<tr></tr>标签之间内容

实例：tr.py

import re  
  
con = "<html><body><table><tr><td>123</td></tr></table></body></html>"  
res\_tr = r'<tr>(.\*?)</tr>'  
m\_tr = re.findall(res\_tr, con, re.S|re.M)  
for item in m\_tr:  
 print(item)

（2）、获取超链接<a href=..></a>之间内容

实例：a.py

import re  
  
content = '''  
<td>  
<a href="https://www.baidu.com/articles/zj.html" title="浙江省">浙江省主题介绍</a>  
<a href="https://www.baidu.com//articles/gz.html" title="贵州省">贵州省主题介绍</a>  
</td>  
'''  
  
*# 获取<a href></a>之间的内容*print('获取链接文本内容:')  
res = r'<a .\*?>(.\*?)</a>'  
mm = re.findall( res, content, re.S | re.M)  
for value in mm:  
 print(value)  
  
*# 获取所有<a href></a>链接所有内容*print('\n获取完整链接内容:')  
urls = re.findall(r"<a.\*?href=.\*?</a>", content, re.I | re.S | re.M)  
for i in urls:  
 print(i)  
  
*# 获取<a href></a>中的URL*print('\n获取链接中URL:')  
res\_url = "(?<=href=\").+?(?=\")|(?<=href=\').+?(?=\')"  
link = re.findall(res\_url, content, re.I | re.S | re.M)  
for url in link:  
 print(url)

获取链接中的URL利用了(?<=…)前向界定和(?=…)后向界定。

（3）、过滤html标签

实例：killHtml.py

**import** re  
  
value = **"<span>abc</br>123</span>"**value = re.sub(**'<[^>]+>'**,**''**, value) *#过滤HTML标签*print(value)

（4）、获取*img的src*

实例：img.py

import re  
  
test = '''<img alt="头像" src="../images/Kuomintang.png" width="19" height="19" border="0" />'''  
print(re.findall(r'src="(.\*?)"',test))