1. 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，就是用实现定义好的一些特定的字符、及这些特定的字符的组合，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。

注意： 正则表达式并不是Python独有的，Python中使用re模块实现。

2. 常见匹配模式参考链接

元字符	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	匹配任意的空白符
\S	匹配任意非空白符
\d	匹配数字
\D	匹配任意非数字的字符
.	匹配除换行符以外的任意字符
^	匹配字符串的开始
$	匹配字符串的结束
*	重复零次或更多次
+	重复一次或更多次
?	重复零次或一次
{n}	重复n次
{n,}	重复n次或更多次
{n,m}	重复n到m次
a|b	匹配a或b
()	匹配括号内的表达式，也表示一个组

3. 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的话，match（）就会返回None

语法： re.match(正则表达式，要匹配的字符串,匹配方式)

常用匹配方式： re.I 使匹配对大小写不敏感 re.S 使 . 匹配包括换行在内的所有字符

```
str1 = 'www.kaikeba.com'
result = re.match('www',str1)

# 获取匹配的结果
print(result.group())

# 获取匹配字符串的长度范围
print(result.span())

# 不在起始位置匹配,返回None
print(re.match('kaikeba',str1))

# 默认大小写是敏感的
print(re.match('WWW',str1))

# 设置匹配模式，忽略大小写
print(re.match('WWW',str1,re.I))
```

开课吧 kaikeba.com

多种匹配方式

```
str2 = 'abc 123 def'
# 常规匹配
print(re.match('^abc\s\d\d\d\sdef$',str2).group())
print(re.match('^abc\s\d{3}\sdef$',str2).group())
# 范匹配
print(re.match('^abc\s.*\sdef$',str2).group())

# 获取指定的字符
print(re.match('^abc\s(.*)\sdef$',str2).group(1))
# 通过re.group()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
```

贪婪和非贪婪

```
# 贪婪匹配
content= "hello 1234567 world Demo"
result= re.match('^hello.*(\d+).*Demo',content)
print(result)
print(result.group(1))
# 这种情况的原因是前面的.* 给匹配掉了， .*在这里会尽可能的匹配多的内容，也就是我们所说的贪婪匹配
result= re.match('^hello.*?(\d+).*Demo',content)
print(result)
print(result.group(1))
```

匹配模式

```
# 很多时候匹配的内容是存在换行的问题的，这个时候的就需要用到匹配模式re.S来匹配换行的内容
content = """hello 123456 world
    demo
"""
result =re.match('^he.*?(\d+).*?demo$',content,re.S)
print(result)
print(result.group())
print(result.group(1))
```

转义

```
# 当我们要匹配的内容中存在特殊字符的时候，就需要用到转移符号\
content= "price is $5.00"
result = re.match('price is \$5\.00',content)
print(result)
print(result.group())
```

4. re.search()

re.search扫描整个字符串返回第一个成功匹配的结果

```
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.*?active.*?singer="(.*?)">(.*?)</a>',html,re.S)
print(result)
print(result.groups())
print(result.group(1))
print(result.group(2))
# 所以为了匹配方便，我们会更多的用search，不用match,match必须匹配头部，所以很多时候不是特别方便
```

5. re.findall()

搜索字符串，以列表的形式返回全部能匹配的子串

```
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])
```

6. re.compile()

将正则字符串编译成正则表达式对象，以便于复用该匹配模式。

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

7. 拓展（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

语法：re.sub(正则表达式，替换成的字符串，原字符串)

```
import re

content = "hello 123456 world"

# content = re.sub('\d+','',content)
# print(content)

#有些情况下我们替换字符的时候，还想获取我们匹配的字符串，然后在后面添加一些内容
#需要注意的一个问题是\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
content = re.sub('(\d+)',r'\1 7890',content)
print(content)
```

开课吧 kaikeba.com