正则表达式与**python**应用

1. 正则表达式概述

正则表达式是一个特殊的字符序列，它能帮助你方便的检查一个字符串是否与某种模式匹配。

2. python中的**re**模块

re 模块使 Python 语言拥有全部的正则表达式功能。

3. compile函数

compile 函数根据一个模式字符串和可选的标志参数生成一个正则表达式对象。该对象拥有一系列方法用于正则表达式匹配和替换。 python.org推荐的做法是，先准备好模式字符串，然后将其与re.flags 一起进行编译。编译返回的对象可以用于在目标字符串中查找或替换能够被匹配的部分。

flags : 可选，表示匹配模式，比如忽略大小写，多行模式等，具体参数为:

- 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 即为 . 并且包括换行符在内的任意字符（. 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
- re.X 为了增加可读性，忽略空格和 # 后面的注释

4. 查找函数

- def match(pattern, string, flags=0) 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。可以使用group(num) 或 groups() 匹配对象函数来获取匹配表达式。
- def search(pattern, string, flags=0) 扫描整个字符串并返回第一个成功的匹配。匹配成功re.search方法返回一个匹配的对象，否则返回None。可以使用group(num) 或 groups() 匹配对象函数来获取匹配表达式。

re.match只匹配字符串的开始，如果字符串开始不符合正则表达式，则匹配失败，函数返回None；而re.search匹配整个字符串，直到找到一个匹配。

- def findall(pattern, string, flags=0) 在字符串中找到正则表达式所匹配的所有子串，并返回一个列表，如果没有找到匹配的，则返回空列表。

注意： match 和 search 是匹配一次 findall 匹配所有。

5. 检索和替换

- def sub(pattern, repl, string, count=0, flags=0) 用于替换字符串中的匹配项。 pattern : 正则中的模式字符串。repl : 替换的字符串，也可为一个函数。string : 要被查找替换的原始字符串。count : 模式匹配后替换的最大次数，默认 0 表示替换所有的匹配。

6. 分割

- def split(pattern, string, maxsplit=0, flags=0) 通过给出的pattern对字符串进行拆分，返回包含匹配结果的字串列表。 如果pattern中包含捕获括号，那么模式中所有组的文本也将作为结果字串返回； 如果maxsplit为非0，则最多拆分为maxsplit项，甚于字串作为列表最后一项。

7. match对象

match对象是查找或替换方法返回的结果。如果存在匹配时，match对象中包含了匹配的结果和出现的位置；当没有匹配时，match() 和 search() 会返回None；可以使用下列语句进行测试 match = re.search(pattern, string) if match: process(match)

match对象支持以下方法和属性：

- group() Return the string matched by the RE
- start() Return the starting position of the match
- end() Return the ending position of the match
- span() Return a tuple containing the (start, end) positions of the match

常见的模式

一、校验数字的表达式

1. 数字：^[0-9]*$
2. n位的数字：^\d{n}$
3. 至少n位的数字：^\d{n,}$
4. m-n位的数字：^\d{m,n}$
5. 零和非零开头的数字：^(0|[1-9][0-9]*)$
6. 非零开头的最多带两位小数的数字：^([1-9][0-9]*)+(.[0-9]{1,2})?$
7. 带1-2位小数的正数或负数：^(-)?\d+(.\d{1,2})?$
8. 正数、负数、和小数：^(-|+)?\d+(.\d+)?$
9. 有两位小数的正实数：^[0-9]+(.[0-9]{2})?$
10. 有1~3位小数的正实数：^[0-9]+(.[0-9]{1,3})?$
11. 非零的正整数：^[1-9]\d*$ 或 ^([1-9][0-9]){1,3}$ 或 ^\+?[1-9][0-9]*$
12. 非零的负整数：^-[1-9][]0-9"$ 或 ^-[1-9]\d*$
13. 非负整数：^\d+$ 或 ^[1-9]\d*|0$
14. 非正整数：^-[1-9]\d*|0$ 或 ^((-\d+)|(0+))$
15. 非负浮点数：^\d+(.\d+)?$ 或 ^[1-9]\d*\.\d*|0\.\d*[1-9]\d*|0?\.0+|0$
16. 非正浮点数：^((-\d+(.\d+)?)|(0+(.0+)?))$ 或 ^(-([1-9]\d*\.\d*|0\.\d*[1-9]\d*))|0?\.0+|0$
17. 正浮点数：^[1-9]\d.\d|0.\d[1-9]\d*$ 或 ^(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*))$
18. 负浮点数：^-([1-9]\d.\d|0.\d[1-9]\d)$ 或 ^(-(([0-9]+\.[0-9]*[1-9][0-9]*)|([0-9]*[1-9][0-9]*\.[0-9]+)|([0-9]*[1-9][0-9]*)))$
19. 浮点数：^(-?\d+)(.\d+)?$ 或 ^-?([1-9]\d*\.\d*|0\.\d*[1-9]\d*|0?\.0+|0)$

二、校验字符的表达式

1. 汉字：^[\u4e00-\u9fa5]{0,}$
2. 英文和数字：^[A-Za-z0-9]+$ 或 ^[A-Za-z0-9]{4,40}$
3. 长度为3-20的所有字符：^.{3,20}$
4. 由26个英文字母组成的字符串：^[A-Za-z]+$
5. 由26个大写英文字母组成的字符串：^[A-Z]+$

6. 由26个小写英文字母组成的字符串：^[a-z]+$
7. 由数字和26个英文字母组成的字符串：^[A-Za-z0-9]+$
8. 由数字、26个英文字母或者下划线组成的字符串：^\w+$ 或 ^\w{3,20}$
9. 中文、英文、数字包括下划线：^[\u4E00-\u9FA5A-Za-z0-9_]+$
10. 中文、英文、数字但不包括下划线等符号：^[\u4E00-\u9FA5A-Za-z0-9]+$ 或 ^[\u4E00-\u9FA5A-Za-z0-9]{2,20}$
11. 可以输入含有^%&',;=?$\"等字符：[^%&',;=?$\x22]+
12. 禁止输入含有~的字符：[^~\x22]+

三、特殊需求表达式

1. Email地址：^\w+([-+.]\w+)@\w+([-.]\w+).\w+([-.]\w+)*$
2. 域名：[a-zA-Z0-9][-a-zA-Z0-9]{0,62}(/.[a-zA-Z0-9][-a-zA-Z0-9]{0,62})+/.?
3. InternetURL：[a-zA-z]+://[^\s] 或 ^http://([\w-]+\.)+[\w-]+(/[\w-./?%&=])?$
4. 手机号码：^(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}$
5. 电话号码("XXX-XXXXXXX"、"XXXX-XXXXXXXX"、"XXX-XXXXXXX"、"XXX-XXXXXXXX"、"XXXXXXX"和"XXXXXXXX)：^((\d{3,4}-)|\d{3.4}-)?\d{7,8}$
6. 国内电话号码(0511-4405222、021-87888822)：\d{3}-\d{8}|\d{4}-\d{7}
7. 身份证号(15位、18位数字)：^\d{15}|\d{18}$
8. 短身份证号码(数字、字母x结尾)：^([0-9]){7,18}(x|X)?$ 或 ^\d{8,18}|[0-9x]{8,18}|[0-9X]{8,18}?$
9. 帐号是否合法(字母开头，允许5-16字节，允许字母数字下划线)：^[a-zA-Z][a-zA-Z0-9_]{4,15}$
10. 密码(以字母开头，长度在6~18之间，只能包含字母、数字和下划线)：^[a-zA-Z]\w{5,17}$
11. 强密码(必须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)：^(?=.\d)(?=.[a-z])(?=.*[A-Z]).{8,10}$
12. 日期格式：^\d{4}-\d{1,2}-\d{1,2}
13. 一年的12个月(01～09和1～12)：^(0?[1-9]|1[0-2])$
14. 一个月的31天(01～09和1～31)：^((0?[1-9])|((1|2)[0-9])|30|31)$
15. 钱的输入格式：
16. 1.有四种钱的表示形式我们可以接受:"10000.00" 和 "10,000.00", 和没有 "分" 的 "10000" 和 "10,000"：^[1-9][0-9]*$
17. 2.这表示任意一个不以0开头的数字,但是,这也意味着一个字符"0"不通过,所以我们采用下面的形式：^(0|[1-9][0-9]*)$
18. 3.一个0或者一个不以0开头的数字.我们还可以允许开头有一个负号：^(0|-?[1-9][0-9]*)$
19. 4.这表示一个0或者一个可能为负的开头不为0的数字.让用户以0开头好了.把负号的也去掉,因为钱总不能是负的吧.下面我们要加的是说明可能的小数部分：^[0-9]+(.[0-9]+)?$
20. 5.必须说明的是,小数点后面至少应该有1位数,所以"10."是不通过的,但是 "10" 和 "10.2" 是通过的：^[0-9]+(.[0-9]{2})?$
21. 6.这样我们规定小数点后面必须有两位,如果你认为太苛刻了,可以这样：^[0-9]+(.[0-9]{1,2})?$
22. 7.这样就允许用户只写一位小数.下面我们该考虑数字中的逗号了,我们可以这样：^[0-9]{1,3}(,[0-9]{3})*(.[0-9]{1,2})?$
23. 8.1到3个数字,后面跟着任意个 逗号+3个数字,逗号成为可选,而不是必须：^([0-9]+|[0-9]{1,3}(,[0-9]{3})*)(.[0-9]{1,2})?$
24. 备注：这就是最终结果了,别忘了"+"可以用"*"替代如果你觉得空字符串也可以接受的话(奇怪,为什么?)最后,别忘了在用函数时去掉去掉那个反斜杠,一般的错误都在这里
25. xml文件：^([a-zA-Z]+-?)+[a-zA-Z0-9]+\.[x|X][m|M][l|L]$
26. 中文字符的正则表达式：[\u4e00-\u9fa5]
27. 双字节字符：[^\x00-\xff] (包括汉字在内，可以用来计算字符串的长度(一个双字节字符长度计2，ASCII字符计1))
28. 空白行的正则表达式：\n\s*\r (可以用来删除空白行)

29. HTML标记的正则表达式：<(\S?)[^>]>.?</\1>|<.? /> (网上流传的版本太糟糕，上面这个也仅仅能部分，对于复杂的嵌套标记依旧无能为力)
30. 首尾空白字符的正则表达式：^\s|\s$或(^\s*)|(\s*$) (可以用来删除行首行尾的空白字符(包括空格、制表符、换页符等等)，非常有用的表达式)
31. 腾讯QQ号：[1-9][0-9]{4,} (腾讯QQ号从10000开始)
32. 中国邮政编码：[1-9]\d{5}(?!\d) (中国邮政编码为6位数字)
33. IP地址：\d+.\d+.\d+.\d+ (提取IP地址时有用)
34. IP地址：((?:(?:25[0-5]|2[0-4]\d|[01]?\d?\d)\.){3}(?:25[0-5]|2[0-4]\d|[01]?\d?\d))

参考资料

- https://docs.python.org/3.6/howto/regex.html#regular-expression-howto

In [12]:
```
""" First, run the Python interpreter, import the re module, and compile a RE"""
import re
p = re.compile('[a-z]+')
p
```

In [14]:
```
"""使用re FLAGS 指定匹配方式"""
import re
p = re.compile('[a-z]+',re.IGNORECASE)
p
```

Out[14]:
```
re.compile(r'[a-z]+', re.IGNORECASE|re.UNICODE)
```

In [24]:
```
"""使用match方法，匹配字符串"""
import re
p = re.compile('[a-z]+')
a = p.match("")
print(a)

b = p.match("ab13cd123")
b.group()
```

```
None
```

Out[24]:
```
'ab'
```

In [9]:
```
"""正则表达式 初试"""
import re

thePattern = 'abc'
srcStr = 'wertyuio123pdfghjklasdabc345fghjkabcrtyuiopabcgh345657-jkl'

theMatch = re.match(thePattern,srcStr)
print(theMatch)

theMatch = re.search(thePattern,srcStr)
print(theMatch)

theMatch = re.findall(thePattern,srcStr)
print(theMatch)

theMatch = re.sub(thePattern,'==ABC==',srcStr)
print(theMatch)

theMatch = re.split(thePattern,srcStr)
```

```
print(theMatch)
```

```
None
<_sre.SRE_Match object; span=(22, 25), match='abc'>
['abc', 'abc', 'abc']
wertyuio123pdfghjklasd==ABC==345fghjk==ABC==rtyuiop==ABC==gh345657-jkl
['wertyuio123pdfghjklasd', '345fghjk', 'rtyuiop', 'gh345657-jkl']
```

In [34]:
```
import re

p = re.compile('abc')
srcStr = 'wertyuio123pdfghjklasdabc345fghjkabcrtyuiopabcgh345657-jkl'

m = p.search(srcStr)
m.group()
```

Out[34]:
```
'abc'
```

In [48]:
```
"""运用正则表达式，匹配页面中所有ip地址"""
import urllib.request
import re
#106.75.226.36
p = re.compile('\d+\.\d+\.\d+\.\d+')

url = 'http://ip.yqie.com/ipproxy.htm'
headers = {'Accept':'text/html,application/xhtml+xml,application/xml;q=0
.9,image/webp,image/apng,*/*;q=0.8',
           'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKi
t/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36',
          }
request = urllib.request.Request(url,headers=headers)
with urllib.request.urlopen(request) as response:
    html = response.read().decode('utf-8')
    match = p.findall(html)

    print(match)
```

```
['106.75.226.36', '118.190.95.35', '61.135.217.7', '118.178.227.171', '1
17.40.159.249', '117.114.149.10', '113.121.240.117', '123.207.117.124',
'222.168.159.189', '125.77.80.105', '60.190.137.194', '125.94.218.149',
'124.235.135.74', '183.62.207.242', '121.40.183.166', '106.75.169.71', '
112.65.144.102', '222.92.145.10', '112.81.16.35', '210.26.124.143', '110
.40.13.5', '183.129.207.73', '183.129.207.70', '59.37.18.243', '112.115.
57.20', '61.128.208.94', '202.112.237.102', '183.129.207.80', '114.116.5
5.108', '211.101.136.86', '221.229.252.98', '221.6.32.214', '120.27.13.1
45', '111.202.37.195', '119.31.210.170', '14.29.32.106', '114.242.143.21
', '121.59.4.237', '106.75.164.15', '210.26.124.143', '106.75.226.36', '
183.129.207.73', '183.129.207.70', '117.40.159.249', '59.37.18.243', '11
3.121.240.117', '61.128.208.94', '202.112.237.102', '183.129.207.80', '1
14.116.55.108', '211.101.136.86', '221.229.252.98', '123.207.117.124', '
222.168.159.189', '125.77.80.105', '60.190.137.194', '125.94.218.149', '
61.135.217.7', '118.178.227.171', '110.40.13.5', '117.114.149.10', '112.
115.57.20', '221.6.32.214', '183.62.207.242', '112.65.144.102', '222.92.
145.10', '157.0.210.242', '101.81.89.188', '58.215.207.69', '119.254.94.
108', '110.188.0.86', '112.84.253.230', '221.13.137.202', '58.251.49.4',
'61.157.206.186', '221.226.68.194']
```