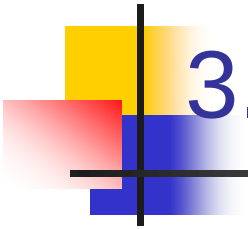


3.5 Python的正则表达式处理

3.5.1 正则表达式

3.5.2 Python中使用正则表达式

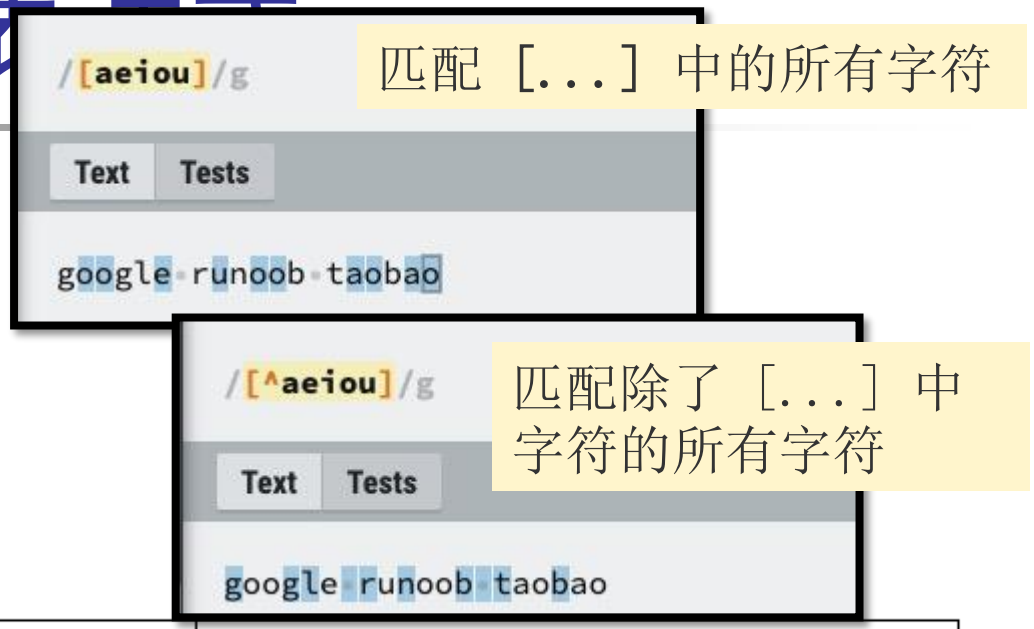
3.5.1 正则表达式

- 正则表达式（Regular Expression）原文链接：https://blog.csdn.net/cherrydreamsover/article/details/80941220
  - 是用于描述一组字符串特征的模式，用来匹配特定的字符串
  - 是一组由字母和符号组成的特殊文本, 可以用来从文本中找出满足你想要的格式的句子。
  - 目前被集成到了各种文本编辑器/文本处理工具当中
- 应用场景
  - （1）验证：表单提交时，进行用户名密码的验证。
  - （2）查找：从大量信息中快速提取指定内容，在一批url中，查找指定url。
  - （3）替换：将指定格式的文本进行正则匹配查找，找到之后进行特定替换。
  - … …

```
^[a-z0-9_-]{3,15}$
```

3.5.1 正则表达式

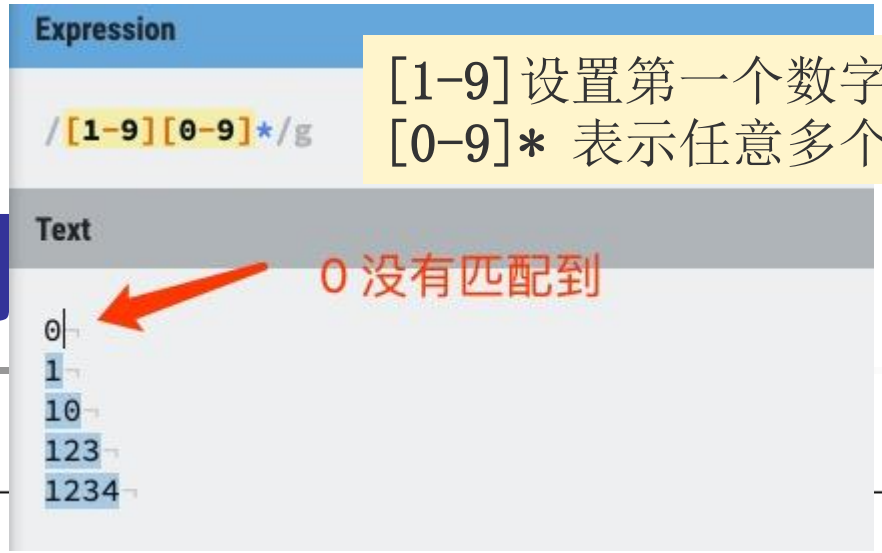
- 基本要素
  - （1）字符类
  - （2）数量限定符
  - （3）位置限定符
  - （4）特殊符号
- 1.字符类



字符	含义	举例
.	匹配任意一个字符	abc. 可以匹配abcd或abc6等
[ ]	匹配括号中的任意一个字符	[abc]d可以匹配ad、bd、cd
-	在[ ]内表示字符范围	[0-9a-fA-F]可以匹配一位十六进制数字
^	位于[ ]括号内的开头，匹配除括号中的字符之外的任意一个字符	[^xy]匹配xy之外的任意一个字符，因此[^xy]1可以匹配a1、b1但不匹配x1、y1
[[:xxx:]]	grep工具预定义的一些命名字符类	[[:alpha:]]匹配一个字母，[[:digit:]]匹配一个数字

3.5.1 正则

- 2.数量限定符



字符	含义	
?	紧跟在它前面的单元应匹配零次或一次	[0-9]?\.[0-9]匹配0.0、1.7、.6等，由于在正则表达式中是一个特殊字符，因此需要\转义取其字面值
+	紧跟在它前面的单元应匹配一次或多次	[a-zA-Z0-9_.-]+@[a-zA-Z0-9_.-]+\.[a-zA-Z0-9_.-]+匹配email地址
*	紧跟在它前面的单元应匹配零次或多次	[0-9][0-9]*匹配至少一位数字，等价于[0-9]+，[a-zA-Z_]+[a-zA-Z_0-9]*匹配C语言的标识符
{N}	紧跟在它前面的单元应精确匹配N次	[1-9][0-9]{2}匹配从100到999的整数
{N,}	紧跟在它前面的单元应匹配至少N次	[1-9][0-9]{2,}匹配大于等于三位数的整数
{,M}	紧跟在它前面的单元应匹配最多M次	[0-9]{,1}指最多匹配0-9之间的一个整数，相当于[0-9]?
{N,M}	紧跟在它前面的单元应匹配至少N次，最多M次	[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}匹配IP地址

3.5.1 正则表达式

- 3.位置限定符

字符	含义	举例
^	匹配行首的位置	^Content匹配位于一行开头的Content
$	匹配行末的位置	;$匹配位于一行结尾的;号，^$匹配空行
\<	匹配单词开头的位置	\<th匹配…this，但不匹配ethernet、tenth
\>	匹配单词结尾的位置	p\>匹配leap …，但不匹配parent、sleepy
\b	匹配单词开头或结尾的位置	\bat\b匹配…at…,但不匹配cat、atexit、batch
\B	匹配非单词开头或结尾的位置	\Bat\B匹配battery，但不匹配…attend、hat…

3.5.1 正则表达式

- 4.特殊符号

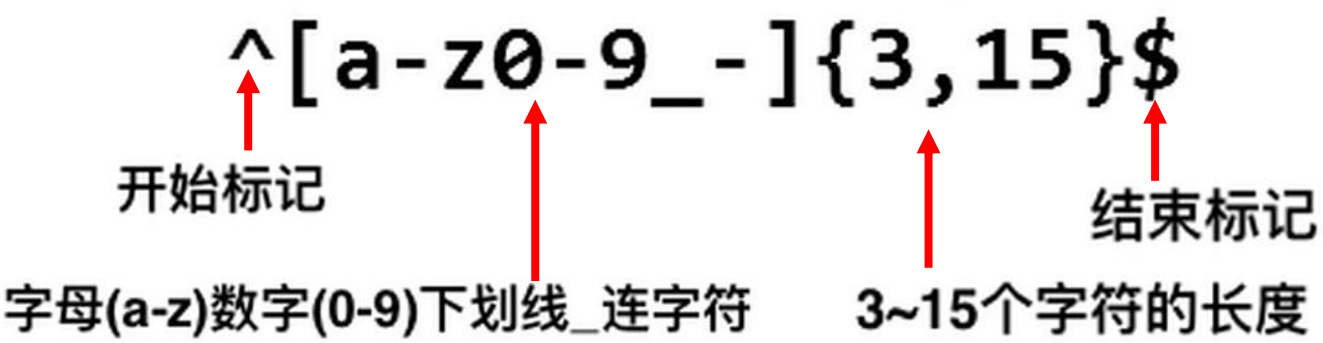
字符	含义	举例
\	转义字符，普通字符转义为特殊字符，特殊字符转义为普通字符	普通字符<写成\<表示单词开头位置，特殊字符.写成\.以及\写成\\就当作普通字符匹配
( )	将正则表达式的一部分括起来组成一个单元，可以对整个单元使用数量限定符	([0-9]{1,3}\.){3}[0-9]{1,3}匹配IP地址
|	连接两个子表达式，表示或的关系	n(o|either)匹配no或neither

- 5.其他普通字符集及其替换

符号	替换正则	匹配
\d	[0-9]	数字字符
\D	[^0-9]	非数字字符
\w	[a-zA-Z0-9]	数字字母下划线
\W	[^\w]	非数字字母下划线
\s	[\r\t\n\f]	表格，换行等空白区域
\S	[^\s]	非空白区域

3.5.1 正则表达式

- 例. 网站中对用户名规则做出了如下限制：
  - 只能包含小写字母、数字、下划线和连字符
  - 并且限制用户名长度在3~15个字符之间
- 使用以下正则表达式限制：



常用的正则表达式http://www.weixuecn.cn/article/12216.html

1. 校验基本日期格式

```
var  reg1 = /^\d{4}(\-|\/|\.)\d{1,2}\1\d{1,2}$/;
var  reg2 = /^(^(\d{4}|\d{2})(\-|\/|\.)\d{1,2}\3\d{1,2}$)|(^\d{4}年\d{1,2}月\d{1,
2}日$)$/;
```

2. 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

```
var  reg = /^(?=.*\\d)(?=.*[a-z])(?=.*[A-Z]).{8,10}$/;
```

3. 校验中文字符串仅能是中文。

```
var  reg = /^[\\u4e00-\\u9fa5]{0,}$/;
```

5. 校验E-Mail 地址　同密码一样，下面是E-mail地址合规性的正则检查语句。

```
var  reg =
/[\\w!#$%&'*+/=?^_`{|}~-]+(?:\\.[\\w!#$%&'*+/=?^_`{|}~-]+)*@(?:[\\w](?:[\\w-]*
[\\w])?\\.)+[\\w](?:[\\w-]*[\\w])?/;
```

9. 校验手机号

下面是国内 13、15、18开头的手机号正

码)

```
var  reg  =  /^(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}
$/;
```

11. 校验IP-v4地址

```
var  reg =
/\\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[0
1]?[0-9][0-9]?)\\b/;
```

3.5.1 正则表达式

- 例. 在字符串中查找电话号码
- 号码模式：
  - 3 个数字，一个短横线，3个数字，一个短横线，再是4 个数字
  - 如：415-555-4242

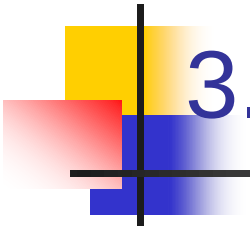
```
def isPhoneNumber(text):
❶    if len(text) != 12:
         return False
     for i in range(0, 3):
❷        if not text[i].isdecimal():
             return False
❸    if text[3] != '-':
         return False
     for i in range(4, 7):
❹        if not text[i].isdecimal():
             return False
❺    if text[7] != '-':
         return False
     for i in range(8, 12):
❻        if not text[i].isdecimal():
             return False
❼    return True
```

正则表达式

```
'\d\d\d-\d\d\d-\d\d\d\d'
```

```
print('415-555-4242 is a phone number:')
print(isPhoneNumber('415-555-4242'))
print('Moshi moshi is a phone number:')
print(isPhoneNumber('Moshi moshi'))
```

```
415-555-4242 is a phone number:
True
Moshi moshi is a phone number:
False
```



3.5 Python的正则表达式处理

3.5.1 正则表达式

3.5.2 Python中使用正则表达式

3.5.2 Python中使用正则表达式

- Python通过re模块提供对正则表达式的支持
- 正则表达式使用：
  - 首先需要定义一个用于匹配的模式（pattern）字符串
  - 之后定义一个匹配的对象：源（source）字符串
- 简例：
- Result = re.match('You' ,Young France')

- 这里的 “You”是模式， “Young France”是源——你想检查的字符串

3.5.2 Python中使用正则表达式

- re库的核心函数
  https://blog.csdn.net/bingocoder/article/details/103746826

方法	说明
compile	将正则表达式的字符串转化为Pattern匹配对象
match	将输入的字符串从头开始对输入的正则表达式进行匹配，一直向后直至遇到无法匹配的字符或到达字符串末尾，将立即返回None，否则获取匹配结果
search	将输入的字符串整个扫描，对输入的正则表达式进行匹配，获取匹配结果，否则输出None
split	按照能够匹配的字符串作为分隔符，将字符串分割后返回一个列表
findall	搜索整个字符串，返回一个列表包含全部能匹配的子串
finditer	与findall方法作用类似，以迭代器的形式返回结果
sub	使用指定内容替换字符串中匹配的每一个子串内容

3.5.2 Python中使用正则表达式

- Python正则表达式使用的四个基本步骤

  1、用import re导入正则表达式模块

  2、用re.complile()函数创建一个Regex对象

  3、使用Regex对象的search()方法传入想查找的字符串，它返回一个Match对象

  4、调用Match对象的group()方法，返回实际匹配的文本字符串

3.5.2 Python中使用正则表达式

- 例.电话号码和E-mail 地址提取程序
- 需要做下面的事情：
  - 创建两个正则表达式，一个匹配电话号码，另一个匹配E-mail地址。
  - 对两个正则表达式，找到所有的匹配，而不只是第一次匹配
  - 将匹配的字符串整理好格式，放在一个字符串中
  - 如果文本中没有找到匹配，显示某种消息。

3.5.2 Python中使用正则表达式

- 第1 步：为电话号码创建一个正则表达式（美国电话）

```
#! python3
# phoneAndEmail.py - Finds phone numbers and email addresses on the clipboard.

import pyperclip, re

phoneRegex = re.compile(r'''(
    (\d{3}|\(\d{3}\))?              # area code
    (\s|-|\.)?                      # separator
    (\d{3})                         # first 3 digits
    (\s|-|\.)                       # separator
    (\d{4})                         # last 4 digits
    (\s*(ext|x|ext.)\s*(\d{2,5}))?  # extension
    )''', re.VERBOSE)

# TODO: Create email regex.

# TODO: Find matches in clipboard text.

# TODO: Copy results to the clipboard.
```

3.5.2 Python中使用正则表达式

- 第2 步：为E-mail 地址创建一个正则表达式

```
#! python3
# phoneAndEmail.py - Finds phone numbers and email addresses on the clipboard.
import pyperclip, re

phoneRegex = re.compile(r'''(
--snip--

# Create email regex.
emailRegex = re.compile(r'''(
❶     [a-zA-Z0-9._%+-]+      # username
❷     @                      # @ symbol
❸     [a-zA-Z0-9.-]+         # domain name
      (\.[a-zA-Z]{2,4})      # dot-something
      )''', re.VERBOSE)

# TODO: Find matches in clipboard text.

# TODO: Copy results to the clipboard.
```

3.5.2 Python中使用正则表达式

- 第3 步：在剪贴板文本中找到所有匹配

```
#! python3
# phoneAndEmail.py - Finds phone numbers and email addresses on the clipboard.

import pyperclip, re

phoneRegex = re.compile(r'''(
--snip--

# Find matches in clipboard text.
text = str(pyperclip.paste())
❶ matches = []
❷ for groups in phoneRegex.findall(text):
    phoneNum = '-'.join([groups[1], groups[3], groups[5]])
    if groups[8] != '':
        phoneNum += ' x' + groups[8]
    matches.append(phoneNum)
❸ for groups in emailRegex.findall(text):
    matches.append(groups[0])

# TODO: Copy results to the clipboard.
```

3.5.2 Python中使用正则表达式

- 第4 步：所有匹配连接成一个字符串，复制到剪贴板

```
#! python3
# phoneAndEmail.py - Finds phone numbers and email addresses on the clipboard.

--snip--
for groups in emailRegex.findall(text):
    matches.append(groups[0])

# Copy results to the clipboard.
if len(matches) > 0:
    pyperclip.copy('\n'.join(matches))
    print('Copied to clipboard:')
    print('\n'.join(matches))
else:
    print('No phone numbers or email addresses found.')
```