# 正则表达式30分钟入门

## 基本概念：

正则表达式：在编写处理字符串的程序或网页时，经常会有查找符合某些复杂规则的字符串的需要。正则表达式就是用于描述这些规则的工具。换句话说，正则表达式就是记录文本规则的代码。

\b是正则表达式规定的一个特殊代码（好吧，某些人叫它元字符，metacharacter），代表着单词的开头或结尾，也就是单词的分界处。\b并不匹配这些单词分隔字符中的任何一个，它只匹配一个位置。

\bhi\b.\*\bLucy\b \*hi\*\*\*\*\*\*\*\*\*Lucy

先是一个单词hi,然后是任意个任意字符(但不能是换行)，最后是Lucy这个单词。

0\d\d-\d\d\d\d\d\d\d\d 0\*\*-\*\*\*\*\*\*\*\* 0\d{2}-\d{8}。

\s匹配任意的空白符，包括空格，制表符(Tab)，换行符，中文全角空格等。

\w匹配字母或数字或下划线或汉字等。

\ba\w\*\b匹配以字母a开头的单词——先是某个单词开始处(\b)，然后是字母a,然后是任意数量的字母或数字(\w\*)，最后是单词结束处(\b)。

\d+匹配1个或更多连续的数字。这里的+是和\*类似的元字符，不同的是\*匹配重复任意次(可能是0次)，而+则匹配重复1次或更多次。

\b\w{6}\b 匹配刚好6个字符的单词。

| **表1.常用的元字符** | |
| --- | --- |
| **代码** | **说明** |
| . | 匹配除换行符以外的任意字符 |
| \w | 匹配字母或数字或下划线或汉字 |
| \s | 匹配任意的空白符 |
| \d | 匹配数字 |
| \b | 匹配单词的开始或结束 |
| ^ | 匹配字符串的开始 |
| $ | 匹配字符串的结束 |

元字符^（和数字6在同一个键位上的符号）和$都匹配一个位置，这和\b有点类似。^匹配你要用来查找的字符串的开头。

$匹配结尾。这两个代码在验证输入的内容时非常有用，比如一个网站如果要求你填写的QQ号必须为5位到12位数字时，可以使用：^\d{5,12}$。

字符转义需要用：反斜杠+需要转义的内容 eg： \\ \\* \.

| **表2.常用的限定符** | |
| --- | --- |
| **代码/语法** | **说明** |
| \* | 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 |

像[aeiou]就匹配任何一个英文元音字母，[.?!]匹配标点符号(.或?或!)。

指定一个字符范围，像[0-9]代表的含意与\d就是完全一致的：一位数字；同理[a-z0-9A-Z\_]也完全等同于\w

\(?0\d{2}[) -]?\d{8} 这个表达式可以匹配几种格式的电话号码，像(010)88886666，或022-22334455，或02912345678等。

分之条件：| eg 0\d{2}-\d{8}|0\d{3}-\d{7}这个表达式能匹配两种以连字号分隔的电话号码：一种是三位区号，8位本地号(如010-12345678)，一种是4位区号，7位本地号(0376-2233445)。

重：使用分枝条件时，要注意各个条件的顺序

你可以用小括号来指定子表达式(也叫做分组)，然后你就可以指定这个子表达式的重复次数了，你也可以对子表达式进行其它一些操作。

后向引用：使用小括号指定一个子表达式后，匹配这个子表达式的文本(也就是此分组捕获的内容)可以在表达式或其它程序中作进一步的处理。默认情况下，每个分组会自动拥有一个组号，规则是：从左向右，以分组的左括号为标志，第一个出现的分组的组号为1，第二个为2，以此类推。

后向引用用于重复搜索前面某个分组匹配的文本。例如，\1代表分组1匹配的文本。

| **表3.常用的反义代码** | |
| --- | --- |
| **代码/语法** | **说明** |
| \W | 匹配任意不是字母，数字，下划线，汉字的字符 |
| \S | 匹配任意不是空白符的字符 |
| \D | 匹配任意非数字的字符 |
| \B | 匹配不是单词开头或结束的位置 |
| [^x] | 匹配除了x以外的任意字符 |
| [^aeiou] | 匹配除了aeiou这几个字母以外的任意字符 |

|  |  |  |
| --- | --- | --- |
| **表4.常用分组语法** | | |
| **分类** | **代码/语法** | **说明** |
| **捕获** | (exp) | 匹配exp,并捕获文本到自动命名的组里 |
| (?<name>exp) | 匹配exp,并捕获文本到名称为name的组里，也可以写成(?'name'exp) |
| (?:exp) | 匹配exp,不捕获匹配的文本，也不给此分组分配组号 |
| **零宽断言** | (?=exp) | 匹配exp前面的位置 |
| (?<=exp) | 匹配exp后面的位置 |
| (?!exp) | 匹配后面跟的不是exp的位置 |
| (?<!exp) | 匹配前面不是exp的位置 |
| **注释** | (?#comment) | 这种类型的分组不对正则表达式的处理产生任何影响，用于提供注释让人阅读 |