正则表达式

python django

基本语法

符号	匹配
.(dot)	任意单一字符
\d	任意一位数字
[A-Z]	A到Z中任意一个字符(大写)
[a-z]	a到z中任意一个字符(小写)
[A-Za-z]	a到z中任意一个字符(不区分大小写)
+	匹配一个或更多(例如,\d+匹配一个或多个数字字符)
[^/]+	一个或多个不为'/'的字符
?	零个或一个(例如,\d?匹配零个或一个数字)
*	0个或更多(例如,\d*匹配0个或更多数字字符)
{1,3}	介于一个和三个（包含）之前的表达式(例如:/d{1,3}匹配一个或两个或三个数字,也就是最少匹配1次,最多匹配3次)

具体分析

- 先分析^和这两个是分别用来匹配字符串的开始和结束. 举例分析 :"$^t\ he$ ": 匹配开头一定要有 " the " 字符串. " the":匹配结束一定要有"the"的字符串.
  "^the$":匹配开头是the,结束也是the的字符串.

- 再分析"*","+","?"
  举例分析如下:
  1."ab *":匹配以a开头的,后面接0个或者更多b的字符串,(abbb,a,abb).

  2."ab+":匹配以a开头的,后面接1个或者更多b的字符串,(ab,abb,abbbbbb).

  3ab?":匹配以a开头的,后面接0个或者1个b的字符串,(a,ab).
  要点
  "*","+","?"只管它前面那个字符.
  要点

- 再分析{num},其中num是一个正数,这个业可以限制字符出现的个数.
  举例分析如下:
  1."ab{2}":要求a后面一定要跟两个b,一个也不能少.(abb)
  2."ab{2,}":要求a后面一定要跟两个或者两个以上b(abb,abbbb)
  3."ab{3,5}":要求a后面一定要跟3到5个b(abbb,abbbb,abbbbb)

- 分析把一定几个字符放到小括号里.
  举例分析如下:
  1."a(ab)*":匹配a后面跟0个或多个ab.
  2."a(bc){3,5}":匹配a后面跟3到5个bc.

- 分析 '|'字符
  这个就是相当于or操作.
  举例如下:
  1."hi|hello":匹配含有"hi"和"hello"的字符串.
  2."(ab|cb)f":匹配含有abf或者是cbf的字符串.
  3."(a|b)*f":匹配含有这样0个或者更多个a或者b,并且后面跟着一个f的字符串.

- 分析'.'
  一个点时可以代表所有的单一字符,不包括'\n',如果要包括'\n'在内的所有单个字符,用[\n.]这种模式.
  举例如下:
  1."a.[0-9]":一个a加一个字符再加一个0到9的字符.
  2."^.{3}$":匹配任意三个字符.

- 中括号[],括住的内容只匹配一个单一的字符.
  举例如下:
  1."[ab]":匹配单个的a或者b.和a|b一样.
  2."[a-d]":匹配a到d之间的一个单一字符.
  3."[0-9]%":匹配含有形如x%的字符串,其中x是0到9任意的一个数字.
  4."^[a-zA-Z]":匹配开头以大小写字母开头的字符串.

- 可以把不想要得到的字符串列在中括号里面,然后在中括号里面加上^作为开头.
  举例如下:
  1."%[^a-zA-Z]%":匹配含有两个%的里面不是字母的字符串.
  要点
  ^用在中括号开头的时候,就表示排除括号里的字符

要点

- 分析\b.
  用它来匹配一个单词的边界.
  举例如下:
  1."ve\b":可以匹配love里的ve,而不是very里的ve.

- 分析\B.
  举例如下:
  1."ve\B":可以匹配very里的ve,而不是love里的ve.

小试牛刀

- 如何构建一个模式来匹配货币数量的输入,构建一个匹配模式去检查输入的信息是否为一个表示money的数字.

  思路:一个表示money的数量有四种方式:1."10000.00",2."10,000.00",或者没有小数部分,"10000"和"10,000".

现在开始构建匹配模式:
首先会想到这个: "^[1-9][0-9]*$":*这个是以非0的数字开头的money*.
这样显然不满足,因为单一的0就不能通过测试,所以改造如下: "^(0|[1-9][0-9])$":*这样就可以表示单一的0和一个以非0的数字开头的money.
但是我们抛开money的实际意义.如何表示负数,正数,0呢?
改造如下: "^(0|-?[1-9][0-9]*)$":这样玖表示了单个*0*,还有以非*0*开头的数字的正负性.继续回到这个问题,不需要负号.然后我们开始号表示小数部分.如下所示:$"^(0|-?[1-9][0-9]*(.[0-9]{2})?$"$ *这样就表示必须最少以一个数字开头.**但是要注意,"10."是不匹配的,"10"和"10.2"才能匹配**.现在我们可以指定小数点后面几位.如下所示:"^[0-9]+(.[0-9]{2})?$"* 同时我们也可以改成 :$"^(0|-?[1-9][0-9]*(.[0-9]{1,2})?$"$.*现在我们开始加上用来增加可读性的逗号(每隔3位),我们可以这样表示如下:"^[0-9]{1,3}(,[0-9]{1,3})*(.[0-9]{1,2})?$".*这就是最终答案了.

- 构造检查email的正则表达式.

  思路:在一个完整的email地址中有三部分.1.用户名(在@左边) 2.@ 3.服务器名字(剩下的那部分).

首先用户名可以含有大小写字母,阿拉伯数字,句点(.),减号,下划线.服务器名字也是这个规则,但是下划线除外. 而且现在用户名和服务器的名字的开始和结束不能是句点.还有你不能有两个连续的句点,它们之间至少存在一个字符

现在为用户名写一个匹配模式:
"^[_a-zA-Z0-9-]+$":这个是句点不存在的情况,现在加上后:
"^[_a-zA-Z0-9-]+(.[*a-zA-Z0-9-]+)*$",*然而服务器的名字也是这样,但是要去掉下划线*"",所以服务器的是:"^[a-zA-Z0-9-]+(.[a-zA-Z0-9-)+]*$".*这样完整的email认证匹配模式是:"^[_a-zA-Z0-9-]+(.[_a-zA-Z0-9]+)*@*[a-zA-Z0-9-]+(.[a-zA-Z0-9-]+)*$".

** 通过这篇文章,应该可以理解正则表达式了吧 **