正则表达式是一种文本模式，包括普通字符（例如，a 到 z 之间的字母）和特殊字符（称为“元字符”）。模式描述在搜索文本时要匹配的一个或多个字符串。  
如：/^\d{2}-\d{5}$/ 验证由两位数字、一个连字符再加 5 位数字组成的 ID 号。

    下列包含了常用元字符的列表以及它们在正则表达式上下文中的行为：

1. \ 将下一字符标记为特殊字符、文本、反向引用或八进制转义符。如：“n”匹配字符“n”。“\n”匹配换行符。

2. ^ 匹配输入字符串开始的位置。

3. $ 匹配输入字符串结束的位置。

4. \* 零次或多次匹配前面的字符或子表达式。如：“han\*” 匹配 “ha”, “han”, "hann"...以此类推。\*等效于{0,}

5. + 一次或多次匹配前面的字符或子表达式。如：“han+” 匹配 “han”, "hann"...以此类推。+等效于{1,}

6. ? 零次或一次匹配前面的字符或子表达式。如：“h(an)?” 匹配 “h”或“han”。?等效于{0,1}

7. {n} n 是非负整数。正好匹配 n 次。如：“o{2}” 不匹配 “boy”, 匹配 “food”

8. {n,} n 是非负整数。至少匹配 n 次。例如，“o{2,}” 不匹配 “boy”中的“o”, 匹配 “food”, "foood","fooood"...以此类推。“o{1,}”等效于“o+”。“o{0,}”等效于“o\*”。

9. {n,m} n,m都是非负整数。其中 n <= m。匹配至少 n 次，至多 m 次。

10. ? 当此字符紧随任何其他限定符（\*、+、?、{n}、{n,}、{n,m}）之后时，匹配模式是“非贪心的”。“非贪心的”模式匹配搜索到的、尽可能短的字符串，而默认的“贪心的”模式匹配搜索到的、尽可能长的字符串。例如，在字符 串“oooo”中，“o+?”只匹配单个“o”，而“o+”匹配所有“o”。

11. (pattern) 匹配 pattern 并捕获该匹配的子表达式。可以使用 $0…$9 属性从结果“匹配”集合中检索捕获的匹配。若要匹配括号字符 ( )，请使用“\(”或者“\)”

12. (?:pattern) 匹配 pattern 但不捕获该匹配的子表达式，即它是一个非捕获匹配，不存储供以后使用的匹配。这对于用“or”字符 (|) 组合模式部件的情况很有用。例如，'industr(?:y|ies) 是比 'industry|industries' 更经济的表达式。

13. (?=pattern) 例如，'Windows (?=95|98|NT|2000)' 匹配“Windows 2000”中的“Windows”，但不匹配“Windows 3.1”中的“Windows”。

14. (?!pattern) 例如，'Windows (?!95|98|NT|2000)' 匹配“Windows 3.1”中的 “Windows”，但不匹配“Windows 2000”中的“Windows”。

15. x|y 例如，匹配 x 或 y。例如，'z|food' 匹配“z”或“food”。'(z|f)ood' 匹配“zood”或“food”。

16. [xyz] 字符集。匹配包含的任一字符。例如，“[abc]” 匹配 “han”中的“a”。

17. [^xyz] 反向字符集。匹配未包含的任何字符。例如，“[^abc]” 匹配 “han”中的“h”。

18. [a-z] 匹配小写字母。匹配指定范围内的任何字符。

19. [A-Z] 匹配大写字母。匹配指定范围内的任何字符。

20. [0-9] 匹配数字。匹配指定范围内的任何字符。

21. \b 匹配一个字边界，即字与空格间的位置。例如，“er\b”匹配“never”中的“er”，但不匹配“verb”中的“er”。

22. \B 非字边界匹配。“er\B”匹配“verb”中的“er”，但不匹配“never”中的“er”。

(一下的匹配符中小写字母为匹配，大写字母为与小写字母相反的匹配法则)

23. \d 数字字符匹配。等效于 [0-9]。

24. \w 匹配任何字类字符，包括下划线。与“[A-Za-z0-9\_]”等效。

**常用正则表达式：**

邮箱的判断：

/^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$/

我们大体上从左到右看看它表示什么：

^ 开始

\w 单个基本字符，可以是 a-z，A-Z，0-9，\_

+ 表示前面的项（\w）有 1 个或多个

\. 就表示点 .

- 就是减号 -

[\.-] 表示点或减号任取其一

? 表示前面的项（[\.-]）有 0 个或 1 个

\w+ 表示有 1 个或多个基本字符

([\.-]?\w+) 用括号表示一组东西： 0 个或 1 个点或减号接着 1 个或多个基本字符

\* 表示有 0 个或多个前面的项（([\.-]?\w+)）

@ 就是字符 @

\w+([\.-]?\w+)\* 表示 1 个或多个基本字符，接着 0 个或多个（0 个或 1 个点或减号接着 1 个或多个基本字符）

\. 就表示点 .

\w{2,3} 2 个或 3 个基本字符，例如 cn，com

+ 表示前面的项（(\.\w{2,3})）有 1 个或多个，例如 .com，.cn，.com.cn

$ 结束

/ 前后两个 / 表示中间是正则表达式

在 JaveScript，对于电邮地址字符串 strEmail，可以用下面句子表示 strEmail是否是 ：

def check\_email(subject):

regex=ur"^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$"

match = re.search(regex, subject)

if not match:

return False

else:

return True

中文判断：

Unicode下：

/^.\*[\u4e00-\u9fa5]+.\*$/

def check\_chinese\_exists\_not\_none(subject):

regex=ur"^.\*[\u4e00-\u9fa5]+.\*$"

match = re.search(regex, subject)

if not match:

return False

else:#有中文

return True

utf8下：

def is\_chinese\_or\_not(subject):

regex=ur"^[\x80-\xff]+$"

match = re.search(regex, subject)

if not match:

return False

else:

return True #只能全是中文

HH:MM:SS格式的时间的测试：

/^(([0-1][0-9])|([1-2][0-3])):[0-5][0-9]:[0-5][0-9]$/

def check\_time(subject):

regex=ur"^(([0-1][0-9])|([1-2][0-3])):[0-5][0-9]:[0-5][0-9]$"

match = re.search(regex, subject)

if not match:

return False

else:#时间格式匹配（如：21:48:59）

return True

import re  
s = "123/456/789135?68?579"  
a = re.split(r'/|\?', s)  
print a