正则表达式模式

模式字符串使用特殊的语法来表示一个正则表达式：

1. 字母和数字表示他们自身，一个正则表达式模式中的字母和数字匹配同样的字符串。

```
re.search(r'H','Hello')  # 这里的 H 表示的就是字母 H 自身，代表有特殊含义
```

2. 多数字母和数字前加一个反斜杠时会拥有不同的含义。

```
ret = re.search(r'\d','he12ms90') # 这里的 \d 表示的是匹配数字
```

3. 标点符号只有被转义时才匹配自身，否则它们表示特殊的含义。

```
ret = re.search(r'.','hello') # 这里的 . 表示的是匹配任意字符
ret = re.search(r'\.','he.llo')  # 这里的 \. 进行了转义，才表示标点符号自身。
```

4. 反斜杠本身需要使用反斜杠转义。由于正则表达式通常都包含反斜杠，所以你最好使用原始字符串来表示它们。模式元素(如 **r'\t'**，等价于 `\\t`)匹配相应的特殊字符。

下表列出了正则表达式模式语法中的特殊元素，如果你使用模式的同时提供了可选的标志参数，某些模式元素的含义会改变。

非打印字符

非打印字符也可以是正则表达式的组成部分。下表列出了表示非打印字符的转义序列：

字符	描述
\cx	匹配由x指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符。
\f	匹配一个换页符。等价于 \x0c 和 \cL。
\n	匹配一个换行符。等价于 \x0a 和 \cJ。
\r	匹配一个回车符。等价于 \x0d 和 \cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。注意 Unicode 正则表达式会匹配全角空格符。
\S	匹配任何非空白字符。等价于 `[^ \f\n\r\t\v]` 。
\t	匹配一个制表符。等价于 \x09 和 \cI。
\v	匹配一个垂直制表符。等价于 \x0b 和 \cK。

特殊字符

所谓特殊字符，就是一些有特殊含义的字符。若要匹配这些特殊字符，必须首先使字符"转义"，即，将反斜杠字符 `\` 放在它们前面。下表列出了正则表达式中的特殊字符：

特殊字符	描述
( )	标记一个子表达式的开始和结束位置。子表达式可以获取供以后使用。要匹配这些字符，请使用 `\(` 和 `\)`。
.	匹配除换行符 `\n` 之外的任何单字符。要匹配 . ，请使用 `\.` 。
[	标记一个中括号表达式的开始。要匹配 [，请使用 `\[`。
\	将下一个字符标记为或特殊字符、或原义字符、或向后引用、或八进制转义符。例如， 'n' 匹配字符 'n'。'\n' 匹配换行符， `\\` 匹配 `\` ，而 `\(` 则匹配 (。
{	标记限定符表达式的开始。要匹配 {，请使用 `\{` 。
|	指明两项之间的一个选择。要匹配 |，请使用 |。
\d	匹配一个数字字符。等价于 [0-9]。
[0-9]	匹配任何数字。等价于 `\d`
\D	匹配一个非数字字符。等价于 `[^0-9]` 。
[a-z]	匹配任何小写字母
[A-Z]	匹配任何大写字母
[a-zA-Z0-9]	匹配任何字母及数字。等价于 `\w`
\w	匹配包括下划线的任何单词字符。等价于 `[A-Za-z0-9_]` 。
\W	匹配任何非单词字符。等价于 `[^A-Za-z0-9_]` 。
[\u4e00-\u9fa5]	匹配纯中文

定位符

定位符使您能够将正则表达式固定到行首或行尾。它们还使您能够创建这样的正则表达式，这些正则表达式出现在一个单词内、在一个单词的开头或者一个单词的结尾。

定位符用来描述字符串或单词的边界，**^** 和 **$** 分别指字符串的开始与结束，**\b** 描述单词的前或后边界，**\B** 表示非单词边界。

正则表达式的定位符有：

特殊字符	描述
^	匹配输入字符串的开始位置，例如：^h匹配以h开头；在方括号表达式中时，它表示不接受该字符集合，例如 `[^0-9]` 匹配除了数字以外的数据。要匹配 ^ 字符本身，请使用 ^。
$	匹配输入字符串的结尾位置。要匹配 `$` 字符本身，请使用 `\$` 。
\b	匹配一个单词边界，即字与空格间的位置。
\B	非单词边界匹配。

限定符

限定符用来指定正则表达式的一个给定组件必须要出现多少次才能满足匹配。有 ***** 或 **+** 或 **?** 或 **{n}** 或 **{n,}** 或 **{n,m}** 共6种。

正则表达式的限定符有：

字符	描述
*	匹配前面的子表达式零次或多次。例如，zo* 能匹配 "z" 以及 "zoo"。* 等价于{0,}。
+	匹配前面的子表达式一次或多次。例如，'zo+' 能匹配 "zo" 以及 "zoo"，但不能匹配 "z"。+ 等价于 {1,}。
?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 、 "does" 中的 "does" 、 "doxy" 中的 "do" 。? 等价于 {0,1}。
{n}	n 是一个非负整数。匹配确定的 n 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{n,}	n 是一个非负整数。至少匹配n 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{n,m}	m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。

示例：

```
re.search(r'\s','大家好 我是 代码')  # 匹配所有的空字符
re.search(r'\S','大家') # 匹配所有的非空字符
re.search(r'\n','大家好\n我是代码') # 匹配换行
re.search(r'n$','hello python') # 匹配以 n 结尾
re.search(r'^h.+n$','hello python') # 匹配以 h 开头，中间出现一次或多次任意字符，并且以n结尾
re.search(r'^ha*','h')  # 匹配以 h 开头，a出现0次或者一次
```

练习:

1. 用户名匹配:由数字、大小写字母、下划线 `_` 和中横线 `-` 组成，长度为4到14位，并且不能以数字开头。

   ```
   r'^\D[a-z0-9A-Z_\-]{3,13}', 'sH_8'
   ```

2. 匹配邮箱

   ```
   r'^([A-Za-z0-9_\-\.])+@([A-Za-z0-9_\-\.])+\.([A-Za-z]{2,4})$
   ```

3. 匹配手机号

   ```
   r'^((13[0-9])|(14[5|7])|(15([0-3]|[5-9]))|(18[0,5-9]))\d{8}$'
   ```

4. 匹配身份证号。

   ```
   r'^[1-9]\d{5}(18|19|20|)\d{2}((0[1-9])|(10|11|12))(([0-2][1-9])|10|20|30|31)\d{3}[0-9Xx]$'
   ```

5. 匹配URL地址

   ```
   r'((ht|f)tps?):\/\/([\w\-]+(\.[\w\-]+)*\/)*[\w\-]+(\.[\w\-]+)*\/?(\?([\w\-\.,@?^=%&:\/~\+#]*)+)?'
   ```

6. 匹配QQ号

   ```
   r'^[1-9][0-9]{4,10}$'
   ```

7. 匹配微信号

   ```
   r'^[a-zA-Z]([-_a-zA-Z0-9]{5,19})+$'
   ```

8. 匹配车牌号

```
r'^[京津沪渝冀豫云辽黑湘皖鲁新苏浙赣鄂桂甘晋蒙陕吉闽贵粤青藏川宁琼使领A-Z]{1}[A-Z]{1}[A-Z0-9]{4}[A-Z0-9挂学警港澳]{1}$'
```