# 正则表达式

## 限定符

```
x*      # x出现0次或多次
x+      # x出现1次或多次
x?      # x出现0次或1次
x{5}    # x出现6次
x{2,5}  # x出现2-6次
x{2,}   # x出现2次以上
```

## 或运算符

```
(x|y)       # 匹配x或b
(xy)|(gy)   # 匹配xy或gy
```

## 字符类

```
[abc]       # 匹配a或b或c
[a-c]       # 匹配a或b或c
[a-zA-Z0-9] # 匹配小写字母或者大写字母或者数字
[^0-9]      # 匹配非数字字符
```

## 元字符

```
\d  # 匹配数字字符
\D  # 匹配非数字字符
\w  # 匹配单词字符（字母、数字、下划线）
\W  # 匹配非单词字符
\s  # 匹配空白字符（包括空格、换行符、制表符）
\S  # 匹配非空白字符
.   # 匹配任意字符（换行符除外）
\b  # 标注字符的边界
^   # 匹配行首
$   # 匹配行尾
```

## 贪婪匹配、懒惰匹配

```
.+  # 贪婪匹配任意字符（换行符除外）
.+? # 懒惰匹配任意字符（换行符除外）
```

注：如果需要匹配正则里面的特殊字符，可以在符号前面加\，和Python里面字符串防止转义一样

## 断言

```
# 也有人叫环视，或者预搜索，或者前瞻后顾

'''
Happiness is not about being immortal nor having food or rights in one's hand.
It’s about having each tiny wish come true, or having something to eat when you
are hungry or having someone's love when you need love.
'''
```

1、正向先行断言：(?=表达式)，从左往右看，所在位置必须符合表达式

```
# 匹配some，而且some后面必须跟着one，可用如下正则：
some(?=one)
>>>'some'    # someone中的some
```

2、正向后行断言：(?<=表达式)，从右往左看，所在位置必须符合表达式

```
# 匹配one，而且one前面必须是some，可用如下正则：
(?<=some)one
>>>'one'     # someone中的one
```

3、反向先行断言：(?!表达式)，从左往右看，所在位置不能符合表达式

```
# 匹配some，而且some后面不能跟one，可用如下正则：
some(?!one)
>>>'some'  # something中的some
```

4、反向后行断言：(?<!表达式)，从右往左看，所在位置不能符合表达式

```
# 匹配one，而且one前面不能是some，可用如下正则：
(?<!some)one
>>>'one'     # rights in one's hand中的one
```

注：如果断言的括号前后没有写内容，则匹配的是空，这个空的位置是符合断言表达式的位置

# re模块

```
'''
在编程语言中使用正则，如果正则表达式中出现了小括号，编程语言会把小括号视为匹配边界，
也就是说它会把小括号里面的内容视为一个group，这个group才是编程语言眼里的正则表达式，
在Python里面的解决方法是在小括号内的最前面加上?:，这样可以申明这个小括号不是一个group
'''

import re

re.findall('.+','字符串', flags=re.S)   # 匹配所有符合条件的值，返回列表
re.search()      # 查找符合规则的字符，只返回第一个，且返回Match对象，匹配失败返回None
re.finditer()    # 返回一个迭代器，迭代器里面是所有符合规则的Match对象
re.match()       # 和search一样，返回Match对象，但要求必须从字符串开头匹配，匹配失败返回
None
re.fullmatch()  # 从头匹配到尾，进行匹配的字符串整体需要符合正则表达式，匹配成功返回Match对
象，匹配失败返回None

re.sub()     # 替换匹配的字符串，返回替换完成的文本
re.subn()    # 替换匹配的字符串，返回替换完成的文本和替换的次数
re.split()  # 用正则表达式的字符串做分隔符，分割原字符串，返回列表
re.compile()     # 创建正则表达式对象，方便后面使用


'''
flags参数:
re.I     不区分大小写
re.M     让^匹配每一行的开头
re.S     #让.匹配所有字符（包括换行符）

注：re下的所有方法，都可以传flags参数,如果想要同时使用多个flags参数，可以使用|进行分割，如：
flags=re.I | re.M
'''
```