## 正则表达式

1. 正则表达式的基本使用

re.match(r’xxx’, 匹配的字符串) 尝试从字符串的起始位置匹配一个模式匹配成功re.match方法返回一个匹配的对象，否则返回None。可以使用group(num) 或 groups() 匹配对象函数来获取匹配表达式，groups()返回的是一个元组。

1. 使用正则表达式匹配单个字符  
   . 匹配任意1个字符（除了\n） re.S则可以匹配换行

re.I 表示大小写不敏感

[a-c ] 匹配[ ]中列举的字符

\d 匹配数字，即0-9

\D 匹配非数字，即不是数字

\s 匹配空白，即 空格，tab键

\S 匹配非空白

\w 匹配单词字符，即a-z、A-Z、0-9、\_。默认是re.U，即Unicode编码，可以匹配中文，加上re.A即可指定ASCII编码。[a-zA-Z0-9\_]

\W 匹配非单词字符

1. 使用正则表达式匹配多个字符  
   \* 匹配前一个字符出现0次或者无限次，即可有可无

+ 匹配前一个字符出现1次或者无限次，即至少有1次

? 匹配前一个字符出现1次或者0次，即要么有1次，要么没有

{m} 匹配前一个字符出现m次

{m,n}匹配前一个字符出现从m到n次

{m,}匹配前一个字符出现m到无限次

1. 匹配开头结尾边界的使用  
   ^ 匹配字符串开头 [^a]表示的是只要不是a

$ 匹配字符串结尾

1. 匹配分组的使用  
   | 匹配左右任意一个表达式

(ab) 将括号中字符作为一个分组

\num 引用分组num匹配到的字符串 \1 \2

(?P<name>) 分组起别名

(?P=name) 引用别名为name分组匹配到的字符串

1. re高级函数 search、findall、sub等的使用  
   re.search 扫描整个字符串并返回第一个成功的匹配。匹配成功re.search方法返回一个匹配的对象，否则返回None。可以使用group(num) 或 groups() 匹配对象函数来获取匹配表达式。

re.findall

在字符串中找到正则表达式所匹配的所有子串，并返回一个列表，如果没有找到匹配的，则返回**空列表**。  
re.sub用于替换字符串中的匹配项

re.split 根据匹配进行切割字符串，并返回一个列表

1. re.match与re.search的区别

re.match只匹配字符串的开始，如果字符串开始不符合正则表达式，则匹配失败，函数返回None；而re.search匹配整个字符串，直到找到一个匹配。

1. 贪婪和非贪婪的特点以及使用方式  
   Python里正则匹配默认是贪婪的（在少数语言里也可能是默认非贪婪），总是尝试匹配尽可能多的字符；非贪婪则相反，总是尝试匹配尽可能少的字符。在**"\*","?","+","{m,n}"后面加上？**，使贪婪变成非贪婪
2. 字符串前加r的作用  
   Python中字符串前面加上 r 表示**原生字符串**