re模块用于Python中正则表达式功能。

compile函数根据一个模式字符串和可选的标志参数生成一个正则表达式对象。该对象拥有一系列方法用于正则表达式匹配和替换。

re 模块也提供了与这些方法功能完全一致的函数，这些函数使用一个模式字符串做为它们的第一个参数。

## re.compile方法

定义：生成一个正则表达式；

语法：re.compile(pattern, flags=0)

escape方法

定义：转义，写正则的时候有时需要将元字符加上反斜扛 \ 去匹配自身，使用该方法可以避免使用反斜扛 \

语法：re.escape(pattern)

## re.match方法

定义：从字符串的起始位置进行匹配，如果起始位置匹配成功，返回一个Match对象，并可以使用group(num) 或 groups()函数来获取相应的内容，否则返回None。

语法：re.match(pattern, string, flags=0)

函数参数说明：pattern, 匹配的正则表达式；string, 要匹配的字符串；【flags, 标志位，用于控制正则表达式的匹配方式，如：是否区分大小写，多行匹配等，参见正则表达式修饰符】

## re.search方法

定义：扫描整个字符串并返回第一个成功的Match对象。

语法：re.search(pattern, string, flags=0)

函数参数说明：pattern, 匹配的正则表达式；string, 要匹配的字符串；【flags, 标志位，用于控制正则表达式的匹配方式，如：是否区分大小写，多行匹配等，参见正则表达式修饰符】

## re.sub方法

定义：替换字符串中的匹配项

语法：re.sub(pattern, repl, string, count=0, flags=0)

参数：pattern : 匹配的正则表达式；repl : 替换的字符串，也可为一个函数。string : 要被查找替换的原始字符串；【count : 模式匹配后替换的最大次数，默认 0 表示替换所有的匹配；flags : 编译时用的匹配模式，数字形式】

## re.findall方法

定义：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。

语法：re.findall(pattern, string, flags=0) 或 pattern.findall(string[, pos[, endpos]])

参数：pattern，匹配的正则表达式；string，要被查找替换的原始字符串；【pos，指定字符串的起始位置，默认为 0；endpos，指定字符串的结束位置，默认为字符串的长度】

## re.finditer方法

定义：在字符串中找到正则表达式所匹配的所有子串，并把它们作为一个迭代器返回

语法：re.finditer(pattern, string, flags=0)

## re.split方法

定义：按照能够匹配的子串将字符串分割后返回列表

语法：re.split(pattern, string[, maxsplit=0, flags=0])

参数：pattern，匹配的正则表达式；string，要被查找替换的原始字符串；【maxsplit，分隔次数，maxsplit=1 分隔一次，默认为 0，不限制次数；flags, 标志位，用于控制正则表达式的匹配方式，如：是否区分大小写，多行匹配等，参见正则表达式修饰符】

re.RegexObject对象

## re.MatchObject对象

1. group([group1, …]) 方法，用于获得一个或多个分组匹配的字符串，当要获得整个匹配的子串时，可直接使用 group() 或 group(0)；

2. start([group]) 方法用于获取分组匹配的子串在整个字符串中的起始位置（子串第一个字符的索引），参数默认值为 0；

3. end([group]) 方法用于获取分组匹配的子串在整个字符串中的结束位置（子串最后一个字符的索引+1），参数默认值为 0；

4. span([group]) 方法返回 (start(group), end(group))。

## 正则表达式模式

. 匹配任意字符，除了换行符，当re.DOTALL标记被指定时，则可以匹配包括换行符的任意字符；

^ 匹配开始位置，多行模式下匹配每一行的开始；

$ 匹配结束位置，多行模式下匹配每一行的结束；

\* 匹配前一个元字符0到多次；

+ 匹配前一个元字符1到多次；

? 匹配前一个元字符0到1次；

{n} 匹配n个前面表达式。例如，"o{2}"不能匹配"Bob"中的"o"，但是能匹配"food"中的两个o。

{n,} 精确匹配n个前面表达式。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} 匹配 n 到 m 次由前面的正则表达式定义的片段，贪婪方式。

[...] 字符集，一个字符的集合，可匹配其中任意一个字符。例如，[amk] 匹配 'a'，'m'或'k'；

[^...] 不在[]中的字符。例如，[^abc] 匹配除了a,b,c之外的字符。

a| b 逻辑表达式”或”，例如，a|b 代表可匹配 a 或者 b匹配a或b

(...) 分组，默认为捕获，即被分组的内容可以被单独取出，默认每个分组有个索引，从 1 开始，按照"("的顺序决定索引值

(?imx) 正则表达式包含三种可选标志：i, m, 或 x 。只影响括号中的区域。

(?-imx) 正则表达式关闭 i, m, 或 x 可选标志。只影响括号中的区域。

(?: re) 类似 (...), 但是不表示一个组

(?imx: re) 在括号中使用i, m, 或 x 可选标志

(?-imx: re) 在括号中不使用i, m, 或 x 可选标志

(?#...) 注释.

(?= re) 前向肯定界定符。如果所含正则表达式，以 ... 表示，在当前位置成功匹配时成功，否则失败。但一旦所含表达式已经尝试，匹配引擎根本没有提高；模式的剩余部分还要尝试界定符的右边。

(?! re) 前向否定界定符。与肯定界定符相反；当所含表达式不能在字符串当前位置匹配时成功。

(?> re) 匹配的独立模式，省去回溯。

\w 匹配数字字母下划线

\W 匹配非数字字母下划线

\s 匹配任意空白字符，等价于 [\t\n\r\f]。

\S 匹配任意非空字符

\d 匹配任意数字，等价于 [0-9]。

\D 匹配任意非数字

\A 匹配字符串开始

\Z 匹配字符串结束，如果是存在换行，只匹配到换行前的结束字符串。

\z 匹配字符串结束

\G 匹配最后匹配完成的位置。

\b 匹配一个单词边界，也就是指单词和空格间的位置。例如， 'er\b' 可以匹配"never" 中的 'er'，但不能匹配 "verb" 中的 'er'。

\B 匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'，但不能匹配 "never" 中的 'er'。

\n, \t, 等。 匹配一个换行符。匹配一个制表符, 等

\1...\9 匹配第n个分组的内容。

\10 匹配第n个分组的内容，如果它经匹配。否则指的是八进制字符码的表达式。

#--------------------------------------------------------------------------------------------------------------------------------------------------------------#

## 正则表达式修饰符

re.I (IGNORECASE) 使匹配对大小写不敏感；

re.L (LOCALE) 字符集本地化。这个功能是为了支持多语言版本的字符集使用环境的，比如在转义符\w，在英文环境下，它代表[a-zA-Z0-9\_]，即所以英文字符和数字。如果在一个法语环境下使用，缺省设置下，不能匹配"é" 或 "ç"。加上这L选项和就可以匹配了。不过这个对于中文环境似乎没有什么用，它仍然不能匹配中文字符。re.M (MULTILINE) 多行模式，改变^ 和 $的行为；

re.S (DOTALL) 使 . 匹配包括换行在内的所有字符；

re.U (UNICODE) 根据Unicode字符集解析字符。这个标志影响 \w, \W, \b, \B.；

re.X (VERBOSE) 冗余模式, 此模式忽略正则表达式中的空白和#号的注释，使你以更灵活的格式编写正则表达式，以便你更易于理解；