正则表达式：Regular Expression，在代码中常简写为regex、regexp或RE

正则表达式使用单个字符串来描述、匹配一系列匹配某个句法规则的字符串。在很多文本编辑器里，正则表达式通常被用来检索、替换那些匹配某个模式的文本。

re模块

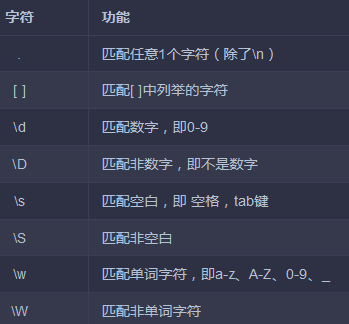
re.match() 能够匹配出以xxx开头的字符串 result = re.match("itcast","itcast.cn")

match()方法判断是否匹配，如果匹配成功，返回一个Match对象，否则返回None

直接给出字符，就是精确匹配

单字符匹配

ret = re.match("[hH]","hello Python")



匹配长字符



\d{3}\s+\d{3,8}

* 1. \d{3}表示匹配3个，数字，例如'010'；
  2. \s可以匹配一个空格（也包括Tab等空白符），所以\s+表示至少有一个空格，例如匹配' '，' '等；
  3. \d{3,8}表示3-8个数字，例如'1234567'。

上面的正则表达式可以匹配以任意个空格隔开的带区号的电话号码。

如果要匹配'010-12345'这样的号码，由于'-'是特殊字符，在正则表达式中，要用'\'转义，所以，上面的正则是\d{3}\-\d{3,8}。

但是，仍然无法匹配'010 - 12345'，因为带有空格。所以我们需要更复杂的匹配方式。

更精确的匹配：可以用[]表示范围

[a-zA-Z\\_][0-9a-zA-Z\\_]{0, 19}

可以匹配由字母或下划线开头，后接任意个由一个数字、字母或者下划线组成的字符串，限制了变量的长度是1-20个字符

A|B可以匹配A或B，所以(P|p)ython可以匹配'Python'或者'python'。

^表示行的开头，^\d表示必须以数字开头。

$表示行的结束，\d$表示必须以数字结束。

原始字符串

Python中字符串前面加上 r 表示原生字符串，正则表达式里使用"\"作为转义字符，假如你需要匹配文本中的字符"\"，那么使用编程语言表示的正则表达式里将需要4个反斜杠"\\"：前两个和后两个分别用于在编程语言里转义成反斜杠，转换成两个反斜杠后再在正则表达式里转义成一个反斜杠。有了原始字符串，不用担心是不是漏写了反斜杠，写出来的表达式也更直观。

切分字符串：用正则表达式来把不规范的输入转化成正确的数组

re.split(r'[\s\,\;]+', 'a,b;; c d') --- ['a', 'b', 'c', 'd']

分组：提取子串，用()表示的就是要提取的分组（Group）

c=re.match(r'(\d{3})\-(\d{3,8})$','010-666666')  
print(c.group(2))

贪婪匹配：正则匹配默认是贪婪匹配，匹配尽可能多的字符

\d+采用贪婪匹配，加个?让\d+采用非贪婪匹配（尽可能少匹配）

c=re.match(r'^(\d+?)(6\*)$','010662666')  
print(c.groups())

编译

Python中使用正则表达式时，re模块内部会干两件事情：

1. 编译正则表达式，如果正则表达式的字符串本身不合法，会报错；
2. 用编译后的正则表达式去匹配字符串。

如果正则表达式要重复使用，可以预编译该正则表达式

re\_num=re.compile(r'^(\d+?)(6\*)$')