python核心-正则

2019年4月14日 9:11

表 示 法	描 述	正则表达式示例
符号		
literal	匹配文本字符串的字面值 *literal*	foo
*re*1|*re*2	匹配正则表达式 *re*1 或者 *re*2	foo|bar
.	匹配任何字符（除了\n 之外）	b.b
^	匹配字符串起始部分	^Dear
$	匹配字符串终止部分	/bin/*sh$
*	匹配 0 次或者多次前面出现的正则表达式	[A-Za-z0-9]*
+	匹配 1 次或者多次前面出现的正则表达式	[a-z]+\.com
?	匹配 0 次或者 1 次前面出现的正则表达式	goo?
{*N*}	匹配 *N* 次前面出现的正则表达式	[0-9]{3}
{*M*,*N*}	匹配 *M*～*N* 次前面出现的正则表达式	[0-9]{5,9}
[…]	匹配来自字符集的任意单一字符	[aeiou]
[..*x*-*y*..]	匹配 *x* ～*y* 范围中的任意单一字符	[0-9], [A-Za-z]
[^…]	不匹配此字符集中出现的任何一个字符，包括某一范围的字符（如果在此字符集中出现）	[^aeiou], [^A-Za-z0-9]
(*|+|?|{})?	用于匹配上面频繁出现/重复出现符号的非贪婪版本（*、+、?、{}）	.*?[a-z]
(…)	匹配封闭的正则表达式，然后另存为子组	([0-9]{3})?,f(oo|u)bar
特殊字符		
\d	匹配任何十进制数字，与[0-9]一致（\D 与\d 相反，不匹配任何非数值型的数字）	data\d+.txt
\w	匹配任何字母数字字符，与[A-Za-z0-9_]相同（\W 与之相反）	[A-Za-z_]\w+
\s	匹配任何空格字符，与[\n\t\r\v\f]相同（\S 与之相反）	of\sthe
\b	匹配任何单词边界（\B 与之相反）	\bThe\b
N	匹配已保存的子组 *N*（参见上面的(…)）	price: \16
c	逐字匹配任何特殊字符 *c*（即，仅按照字面意义匹配，不匹配特殊含义）	\., \\, *
\A(\Z)	匹配字符串的起始（结束）（另见上面介绍的^和$）	\ADear
扩展表示法		
(?iLmsux)	在正则表达式中嵌入一个或者多个特殊“标记”参数（或者通过函数/方法）	（?x），ℓ im）
(?:…)	表示一个匹配不用保存的分组	(?:\w+\.)*
(?P<*name*>…)	像一个仅由 name 标识而不是数字 ID 标识的正则分组匹配	(?P<data>)
(?P=*name*)	在同一字符串中匹配由(?P<name)分组的之前文本	(?P=data)
(?#…)	表示注释，所有内容都被忽略	(?#comment)
(?=…)	匹配条件是如果…出现在之后的位置，而不使用输入字符串；称作正向前视断言	(?=.com)
(?!…)	匹配条件是如果…不出现在之后的位置，而不使用输入字符串；称作负向前视断言	(?!.net)
(?<=…)	匹配条件是如果…出现在之前的位置，而不使用输入字符串；称作正向后视断言	(?<=800-)
(?<!…)	匹配条件是如果…不出现在之前的位置，而不使用输入字符串；称作负向后视断言	(?<!192\.168\.)
(?(*id/name*)*Y*|*N*)	如果分组所提供的 *id* 或者 *name*（名称）存在，就返回正则表达式的条件匹配 *Y*，如果不存在，就返回 *N*；|*N* 是可选项	(?(1)y|x)

正则表达式模式	匹配的字符串
at | home	at、home
r2d2 | c3po	r2d2、c3po
bat | bet | bit	bat、bet、bit

正则表达式模式	匹配的字符串
f.o	匹配在字母“f”和“o”之间的任意一个字符；例如 fao、f9o、f#o 等
..	任意两个字符
.end	匹配在字符串 end 之前的任意一个字符

正则表达式模式	匹配的字符串
^From	任何以 From 作为起始的字符串
/bin/tcsh$	任何以/bin/tcsh 作为结尾的字符串
^Subject: hi$	任何由单独的字符串 Subject: hi 构成的字符串

正则表达式模式	匹配的字符串
the	任何包含 the 的字符串
\bthe	任何以 the 开始的字符串
\bthe\b	仅仅匹配单词 the
\Bthe	任何包含但并不以 the 作为起始的字符串

正则表达式模式	匹配的字符串
b[aeiu]t	bat、bet、bit、but
[cr][23][dp][o2]	一个包含四个字符的字符串，第一个字符是“c”或“r”，然后是“2”或“3”，后面是“d”或“p”，最后要么是“o”要么是“2”。例如，c2do、r3p2、r2d2、c3po 等

正则表达式模式	匹配的字符串
z.[0-9]	字母“z”后面跟着任何一个字符，然后跟着一个数字
[r-u][env-y][us]	字母“r”、“s”、“t”或者“u”后面跟着“e”、“n”、“v”、“w”、“x”或者“y”，然后跟着“u”或者“s”
[^aeiou]	一个非元音字符（练习：为什么我们说“非元音”而不是“辅音”？）
[^\t\n]	不匹配制表符或者\n
[“-a]	在一个 ASCII 系统中，所有字符都位于“”和“a”之间，即 34~97 之间

正则表达式模式	匹配的字符串
[dn]ot?	字母“d”或者“n”，后面跟着一个“o”，然后是最多一个“t”，例如，do、no、dot、not
0?[1-9]	任何数值数字，它可能前置一个“0”，例如，匹配一系列数（表示从 1～9 月的数值），不管是一个还是两个数字
[0-9]{15,16}	匹配 15 或者 16 个数字（例如信用卡号码）
</?[^>]+>	匹配全部有效的（和无效的）HTML 标签
[KQRBNP][a-h][1-8]-[a-h][1-8]	在“长代数”标记法中，表示国际象棋合法的棋盘移动（仅移动，不包括吃子和将军）。即“K”、“Q”、“R”、“B”、“N”或“P”等字母后面加上“a1”～“h8”之间的棋盘坐标。前面的坐标表示从哪里开始走棋，后面的坐标代表走到哪个位置（棋格）上

正则表达式模式	匹配的字符串
\w+-\d+	一个由字母数字组成的字符串和一串由一个连字符分隔的数字
[A-Za-z]\w*	第一个字符是字母；其余字符（如果存在）可以是字母或者数字（几乎等价于 Python 中的有效标识符［参见练习］）
\d{3}-\d{3}-\d{4}	美国电话号码的格式，前面是区号前缀，例如 800-555-1212
\w+@\w+\.com	以 *XXX@YYY*.com 格式表示的简单电子邮件地址

正则表达式模式	匹配的字符串
\d+(\.\d*)?	表示简单浮点数的字符串；也就是说，任何十进制数字，后面可以接一个小数点和零个或者多个十进制数字，例如“0.004”、“2”、“75.”等
(Mr?s?\.)?[A-Z][a-z]*[A-Za-z-]+	名字和姓氏，以及对名字的限制（如果有，首字母必须大写，后续字母小写），全名前可以有可选的“Mr.”、“Mrs.”、“Ms.”或者“M.”作为称谓，以及灵活可选的姓氏，可以有多个单词、横线以及大写字母

正则表达式模式	匹配的字符串
(?:\w+\.)*	以句点作为结尾的字符串，例如“google.”、“twitter.”、“facebook.”，但是这些匹配不会保存下来供后续的使用和数据检索
(?#comment)	此处并不做匹配，只是作为注释
(?=.com)	如果一个字符串后面跟着“.com”才做匹配操作，并不使用任何目标字符串
(?!.net)	如果一个字符串后面不是跟着“.net”才做匹配操作
(?<=800-)	如果字符串之前为“800-”才做匹配，假定为电话号码，同样，并不使用任何输入字符串
(?<!192\.168\.)	如果一个字符串之前不是“192.168.”才做匹配操作，假定用于过滤掉一组 C 类 IP 地址
(?(1)y|x)	如果一个匹配组 1（\1）存在，就与 y 匹配；否则，就与 x 匹配

函数/方法	描　　述
仅仅是 re 模块函数	
compile(*pattern*，*flags* = 0)	使用任何可选的标记来编译正则表达式的模式，然后返回一个正则表达式对象
re 模块函数和正则表达式对象的方法	
match(*pattern*，*string*，*flags*=0)	尝试使用带有可选的标记的正则表达式的模式来匹配字符串。如果匹配成功，就返回匹配对象；如果失败，就返回 None
search(*pattern*，*string*，*flags*=0)	使用可选标记搜索字符串中第一次出现的正则表达式模式。如果匹配成功，则返回匹配对象；如果失败，则返回 None
findall(*pattern*，*string* [, *flags*])①	查找字符串中所有（非重复）出现的正则表达式模式，并返回一个匹配列表
finditer(*pattern*，*string* [, *flags*])②	与 findall()函数相同，但返回的不是一个列表，而是一个迭代器。对于每一次匹配，迭代器都返回一个匹配对象
split(*pattern*，*string*，*max*=0)③	根据正则表达式的模式分隔符，split 函数将字符串分割为列表，然后返回成功匹配的列表，分隔最多操作 *max* 次（默认分割所有匹配成功的位置）
re 模块函数和正则表达式对象方法	
sub(*pattern*，*repl*，*string*，*count*=0) ③	使用 *repl* 替换所有正则表达式的模式在字符串中出现的位置，除非定义 *count*，否则就将替换所有出现的位置（另见 subn()函数，该函数返回替换操作的数目）
purge()	清除隐式编译的正则表达式模式
常用的匹配对象方法（查看文档以获取更多信息）	
group(*num*=0)	返回整个匹配对象，或者编号为 *num* 的特定子组
groups(*default*=None)	返回一个包含所有匹配子组的元组（如果没有成功匹配，则返回一个空元组）
groupdict(*default*=None)	返回一个包含所有匹配的命名子组的字典，所有的子组名称作为字典的键（如果没有成功匹配，则返回一个空字典）
常用的模块属性（用于大多数正则表达式函数的标记）	
re.I、re.IGNORECASE	不区分大小写的匹配
re.L、re.LOCALE	根据所使用的本地语言环境通过\w、\W、\b、\B、\s、\S 实现匹配
re.M、re.MULTILINE	^和$分别匹配目标字符串中行的起始和结尾，而不是严格匹配整个字符串本身的起始和结尾
re.S、rer.DOTALL	“.”（点号）通常匹配除了\n（换行符）之外的所有单个字符；该标记表示“.”（点号）能够匹配全部字符
re.X、re.VERBOSE	通过反斜线转义，否则所有空格加上#（以及在该行中所有后续文字）都被忽略，除非在一个字符类中或者允许注释并且提高可读性