python 正则表达式

作者：shuaiqijun
原文链接：https://ld246.com/article/1649514432628
来源网站：链滴

```
"""
正则表达式
"""
"""
re.match 函数
原形：match(pattern, string, flags=0)
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，值如下：
re.I    忽略大小写
re.L    做本地户识别
re.M    多行匹配，影响^和￥
re.S    是.匹配包括换行符在内的所有字符
re.U    根据Unicode字符集解析字符，影响\W   \w  \B  \b
re.X    使我们以更灵活的格式理解正则表达式
参数：
功能：尝试从字符串的起始位置匹配一个模式，如果不是起始位置匹配成功的话，返回None
"""
# www.baidu.com
import re

a = re.match('www', 'www.baidu.com')
print(re.match('www', 'www.baidu.com'))  # <re.Match object; span=(0, 3), match='www'>
print(re.match('www', 'ww.baidu.com'))  # None
print(re.match('www', 'baidu.wwwcom'))  # None
print(re.match('www', 'wwW.baidu.com'))  # None
print(re.match('www', 'wwW.baidu.com', flags=re.I))  # <re.Match object; span=(0, 3), match=
wwW'>
print(a.span())  # (0, 3)
print('------------------------------------------------------------------------------')

# 扫描整个字符串，返回从起始位置成功的匹配
"""
re.search()函数
原形：search(pattern, string, flags=0)
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，值如下：
功能：扫描整个字符串，并返回第一个成功的匹配
"""
print(re.search('sunck', 'good man is sunck!sunck is nice'))  # <re.Match object; span=(12, 17),
match='sunck'>

"""
re.findall()函数
原形：findall(pattern, string, flags=0)
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，值如下：
功能：扫描整个字符串，并返回第一个成功的匹配
"""
print(re.findall('sunck', 'good man is sunck!sunck is nice'))  # ['sunck', 'sunck']

print('---------------------------------匹配单个字符与数字---------------------------------')
```

```
"""
r
.               匹配除换行符以外的任意字符
[0123456789]    匹配单个数字，[]是字符集合，表示匹配方括号中所包含的任意一个字符
[sunck]         匹配's'  'u' 'n' 'c' 'k'中任意一个字符
[a-z]           匹配任意小写字母
[A-Z]           匹配任意大写字母
[0-9]           匹配任意数字
[0-9a-zA-Z]     匹配任意字母和数字
[0-9a-zA-Z_]    匹配任意字母和数字和下划线
[^sunck]        匹配除了's' 'u' 'n' 'c' 'k'这几个字母以外的所有字符，中括号的^称为脱字符，表示不匹配集合中的字符
[^0-9]          匹配所有的非数字字符
\d              匹配数字，效果通[0-9]
\D              匹配非数字字符，效果同[^0-9]
\w              匹配数字、字母和下划线，效果同[0-9a-zA-Z_]
\W              匹配非数字，字母和下划线，效果同[^0-9a-zA-Z_]
\s              匹配任意的空白字符（空格、回车、换页、制表符、）[ \f\n\t\r]
\S              匹配任意的非空白符[^ \f\n\t\r]

"""
print(re.search('.', 'sunck is a good man'))  # <re.Match object; span=(0, 1), match='s'>
print(re.search('[0123456789]', 'sunck is a good man 6'))  # <re.Match object; span=(20, 21), match='6'>
print(re.findall('.', '.abc.s/dedf$%Ew-ty#@!*()'))      # ['.', 'a', 'b', 'c', '.', 's', '/', 'd', 'e', 'd', 'f', '$', '%', 'E', 'w', '-', 't', 'y', '#', '@', '!', '*', '(', ')']

print('---------------------------------锚字符（边界字符）---------------------------------')
"""
^               行首匹配，和在[]里的^不是一个意思
$               行尾匹配
\A              匹配字符串开始，它和^的区别是，\A只匹配整个字符串的开头，即使在re.Mm模式下也会匹配其他行的行首
\Z              匹配字符串结束，它和$的区别是，\Z只匹配整个字符串的结束，即使在re.Mm模式下也会匹配其他行的行尾
\b              匹配一个单词的边界，也就是指单词和空格间的位置
\B              匹配非单词的边界
"""
b = re.search('boy$', 'sunck is a good boy')
print(re.search('^sunck', 'sunck is a good boy'))   # <re.Match object; span=(0, 5), match='sunk'>
print(re.search('^sunck$', 'sunck is a good boy'))  # None
print(re.search('boy$', 'sunck is a good boy'))     # <re.Match object; span=(16, 19), match='boy'>
print(b.span())                                     # (16, 19)
print(re.findall('^sunck', 'sunck is a good boy\nsunck is a bad man',re.M)) # ['sunck', 'sunck']
print(re.findall('\Asunck', 'sunck is a good boy\nsunck is a bad man'))     # ['sunck']

print(re.search(r'er\b','server is server '))
print('---------------------------------匹配多个字符---------------------------------')
"""
说明：下方的x,y,z均为假设的普通字符吗，不是正则表达式的元字符

(xyz)           匹配小括号内的xyz（作为一个整体去匹配）
```

```
x?          匹配0个或者1个x，非贪婪匹配（尽可能少的匹配）
x*          匹配0个或者多个x,贪婪匹配（尽可能多的匹配）
.*          匹配0个或者任意多个字符（换行符除外）
x+          匹配至少一个x,贪婪匹配
x{n}        匹配确定的n个x（n是一个非负整数）
x{n,}       匹配至少n个x
x{n,m}      匹配至少n个x,最多m个x,n<=m
x|y         匹配x或y,|表示或
"""
print(re.findall(r'(sunck)','sunckgood is a godo man,sunck is a boy')) # ['sunck', 'sunck']
print(re.findall(r'(.*)','sunckgood is a godo man,sunck is a boy'))
print(re.findall(r'(a.*)','aaaaaabaacaaaa'))
print(re.findall(r'(a*)','aaaaaabaacaaaa'))
print(re.findall(r'(a+)','aaaaaabaacaaaa'))
print(re.findall((r'a{2}'),'aaaaaabaacaaaa'))
print(re.findall((r'a{3,}'),'aaaaaabaacaaaa'))
print(re.findall((r'a{3,6}'),'aaaaaabaacaaaa'))
print(re.findall(r'((s|S)unck)','sunck is Sunck'))
print(re.findall(r'(^sunckgood is a godo man,sunck is a boy.*)$','sunckgood is a godo man,sunk is a boy'))
print('---------------------------------特殊---------------------------------')
"""
说明：下方的x,y,z均为假设的普通字符吗，不是正则表达式的元字符

x? +? *? 最小匹配，通常都是尽可能多的匹配，可以使用这种方式来解决贪婪匹配

"""
# /* part1 */ /* part2 */
print(re.findall(r'(//*.*/*/)','/* part1 */ /* part2 */'))      # ['/* part1 */ /* part2 */']
print(re.findall(r'(//*.*?/*/)','/* part1 */ /* part2 */'))     # ['/* part1 */', '/* part2 */']
```