正则表达式

并不是Python独有的--是一套引擎，用于做字符串的检索
1. 网络爬虫爬去数据应该用到正则表达式
2. 正则表达式：起源于Unix

- 正则的基本概念

是一种小型的，高度专业化的编程语言
它内嵌在了Python的re模块下
1. 可以为想要的匹配的字符制定查找
2. 可以通过re模块多种方式修改和查找字符串
运行原理：正则表达式被编译成字节码，由c编写的匹配引擎执行

- re模块

1. findall(正则表达式，目标字符串)
        返回一个列表，列表中是符合要求字符串

- 字符串的匹配

1. 普通字符
        多数的字母和字符都可以进行自身匹配
2. 元字符
        . ^ $? + {} () [] \ |

- 元字符

1. []
    a.常用来指定一个字符集
    b.可以表示一定的范围
    c.[]内的元字符不起作用，指标是字符
        如果把^放在了[]的第一位表示处了..以外的字符都进行匹配
        print(re.findall('[^a]','a^acadcabc'))#除了a以外的字符都进行匹配
2. ^
        通常来匹配行首
3. $
        通常来匹配行尾
4. \
        反斜杠后面可以加不同的字符，表示不同的含义
    \d:匹配任何十进制的数：等价于[0-9]
    \D:匹配任何非十进制的数[^0-9]

    \:可以转义：用于取消元字符，转成普通的字符
        print(re.findall('\^abc','^abc^abc^abc'))

```
    \s:匹配任何的空白字符：\t \n \r   等价于[\t\n\r]
    \S:同上取反，等价于[^\t\n\r]

    \w:匹配任何的字母，数字和下划线 等价于：[A-Za-z0-9_]
    \W:同上取反
5. {n}
    重复
    n：重复的次数
6. *
        指定一个字符可以匹配0次或多次(尽可能多的取值)
7. +
        表示匹配一次或者多次（尽可能多的取值）
8. .
        表示出了换行符以外的任何字符
        除了\n
9.  ?
        指定匹配0次或1次
        +，*：尽可能多的匹配数据---贪婪模式
        +?,*?：尽可能少的获取数据---非贪婪模式

10. {m,n}
        表示最少重复m次，最多是n次（尽可能多的获取）
        print(re.findall('ab{2,5}?','abbbbbbbb')) #{m,n}?尽可能少的获取
        {0,} : *
        {1,} : +
        {0,1} : ?
```

- 正则表达式的使用

```
1. re模块提供了一个正则表达式的引擎接口，把正则表达式编译成对象来进行匹配

2. re.compole(正则表达式)

s='010-123456789'
rule='010-\d*'
rule_compile=re.compile(rule)
print(rule_compile.findall(s))
```

- 正则对象的方法

方法	作用
match	决定re是否在字符串刚开始的位置匹配（匹配行首）
search	扫描字符串，找到re匹配的位置（首次查到的）
findall	找到re匹配到的字符串，返回一个列表
finditer	找到re匹配的所有字符串，并返回一个可迭代对象，返回的match对象

如果匹配到了字符串返回Match对象，否则返回None

- Match对象方法

```
1. group()
        返回re匹配的字符串
2. start()
        返回匹配开始的位置
3. end ( )
        返回匹配结束的位置
4. span ( )
        返回一个元祖（开始，结束）的位置
```

- re模块的函数

```
1. findall()
        根据正则表达式找到匹配所有的字符串，返回一个列表
2. sub(正则，新字符串，原字符串)
        替换字符串
3. subn（正则，新字符串，原字符串）
        替换字符串，返回替换的次数
4. split ( )
        分割字符串
        sub：分割的字符（正则）
        s：要分割的字符串
        s='123*456+789&110)100^1230'
        print(re.split('[*,+,&,),^]',s))
```

- 编译标示--flags

标识	含义
DOTALL，S	使 . 匹配包含换行符在内的所有字符
IGNORECASE，I	使匹配对大小写不敏感
MULTILINE，M	多行匹配，影响^和$
VERBOSE,X	能够把re详细化，使re被组织的清晰易懂

```
import re
#S
# print(re.findall('baizhi.com','baizhi\ncom',re.S))

#I:大小写不敏感
print(re.findall('baizhi','BaiZhibaizhi',re.I))

# M:多行匹配
s="""a
tiger
monkey
panda
```

```
elephant
python
pig
"""
print(re.findall('^monkey',s,re.M))

# X使re清晰易懂
rule="""010
-?
\d*
"""

print(re.findall(rule,'010-12345678',re.X))
```

- 分组：()

```
1. 只显示分组后的内容
2.  | :表示或

import re
s1='123@qq.com'
s2='123@163.com'
s3='123@zpark.cn'
s4='123@hehe.org'
rule='\w+@\w+\.com|\w+@\w+\.cn'

print(re.findall(rule,s1))
print(re.findall(rule,s2))
print(re.findall(rule,s3))
---------------------------------
import re
s="""
Python hehe
a=hehe
a=Python  a=monkey
baizhi
"""
print(re.findall('a=(\w*)',s))
```