正则表达式

动机

1. 文本处理已经成为计算机的常见工作之一
2. 对文本内容的搜索，定位，提取是逻辑比较复杂的工作
3. 为了快速解决上述问题，产生了正则表达式技术

定义：即文本的高级匹配模式，提供搜索，替代等功能。其本质是一系列由特殊符号组成的子串，这个字串即正则表达式。

匹配原理：由普通字符和特殊符号组成字符串，通过描述字符的重复和位置等行为，达到匹配某一类字符串的目的

目标:

1. 熟练掌握正则表达式符号
2. 能够实现基本的文本搜索，定位，提取，理解正则用法
3. 能够使用re模块操作正则表达式

特点

\*方便文本处理

\*支持语言众多

\*使用灵活多样

Python---》re模块

re.findall(pattern，string)

功能：使用正则表达式匹配目标字符串内容

参数：pattern正则表达式

string 目标字符串

返回值：；列表，列表中为匹配到的内容

元字符的使用

1. 普通字符

元字符：a b c

匹配规则：每个字符匹配对应的字符

In [2]: re.findall("年","年后")

Out[2]: ['年']

1. 或

元字符： |

匹配规则：匹配|两边的任意字符

In [3]: re.findall("as|gf","asdfgnmkasdfghjbvchgfd")

Out[3]: ['as', 'as', 'gf']

1. 匹配单个字符

元字符：.

匹配规则：匹配除换行外的任意字符

f.o-->foo fao

1. 匹配开始位置

元字符：^

匹配规则：匹配目标字符串的开头位置

In [9]: re.findall('^lal','lalislallalal')

Out[9]: ['lal']

1. 匹配结束位置

元字符：$

匹配规则：匹配字符串的结束位置

6．匹配重复

元字符：\*

匹配规则：匹配前面的字符出现0次或多次

fo\* 🡪 foooo f fo

1. 匹配重复

元字符：+

匹配规则：匹配前面的字符出现1次或者多次

fo+ --》fo fooo

In [31]: re.findall('fo+','ffooffofoaosodoof')

Out[31]: ['foo', 'fo', 'fo']

1. 匹配重复

元字符 ：？

匹配规则： 匹配前面的字符出现0次或1次

fo? ---> fo f

1. 匹配重复

元字符：｛n｝

匹配 规则：匹配指定的重复次数

fo{3} –》fooo

1. 匹配重复

元字符：｛m,n｝

匹配规则：匹配前面的正则表达式m-----n次

fo{2,4} --> foo fooo foooo

In [45]: re.findall("fo{2,5}","asdfofofofofpasdofofoofofofoofofoofooo")

Out[45]: ['foo', 'foo', 'foo', 'fooo']

1. 匹配字符集合

元字符：[字符集]

匹配规则：匹配任意一个字符集中的字符

[abc123] a b c 1 2 3

[a-z]

[A-Z]

[0-9]

1. 匹配字符集

元字符：[^…]

匹配规则：字符集取非，除列出的字符之外任意一个字符

[^abc]---》》除a,b,c之外的任意字符

1. 匹配任意（非）数字字符

元字符：\d \D

匹配规则： \d匹配任意数字字符

\D匹配任意非数字字符

1. 匹配任意（非）普通字符

元字符：\w \W

匹配规则：\w普通字符 [\_0-9A-Za-z]也能匹配汉字字符

\W非普通字符

1. 匹配任意（非）空字符

元字符： \s 匹配任意空字符[ \r\t\n\v\f]

\S匹配任意非空字符

1. 匹配字符串位置

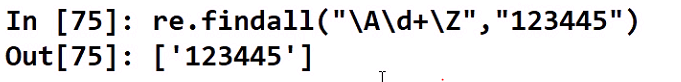
元字符： \A \Z

匹配规则：\A匹配字符串的开头位置 ^

\Z匹配字符串结尾的位置 $

绝对匹配：正则表达式要完全匹配目标字符串内容

在正则表达式开始和结束位置加上^ $(或者\A \Z).这样正则表达式必须匹配整个目标字符串才会有结果。



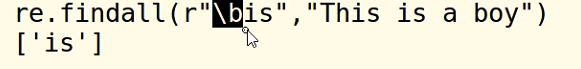
1. 匹配（非）单词边界

元字符：\b \B

匹配规则： \b匹配单词边界位置

普通字符和非普通字符交界认为是单词边界

\B匹配非单词边界位置



元字符总结：

匹配单个字符：a . \d \D \w \W \s \S […] [^…]

匹配重复：\* + ？ ｛n｝{m,n}

匹配位置：^ $ \A \Z \b \B

其他：| （） \

正则表达式转义

正则中的特殊符号：：

. \* + ? ^ $ [] ｛｝() | \

正则表达式如果匹配特殊字符需要加\表达转义

e.g. \$\d+ ---> $10

pattern string

python “\\$\\d+” ‘$10’

raw r“\$\d+” ‘$10’

raw字串： 原始字符串对内容不解释转义，就表达内容原本意义

贪婪与非贪婪

贪婪模式：正则表达式的重复匹配总是尽可能多的向后匹配更多的内容

\* + ？ ｛m,n｝

非贪婪：（懒惰模式）需要加思考（？）尽少 的匹配少的内容

贪婪----》非贪婪 \*? +? ?? {m,n}?

正则表达式的子组

可以使用（）为正则表达式建立子组，子组可以看作是正则表达式内部操作的一个整体

\*子组是在正则表达式整体匹配到内容的前提下才会发挥作用，他不影响正则表达式整体去匹配目标内容这一原则。

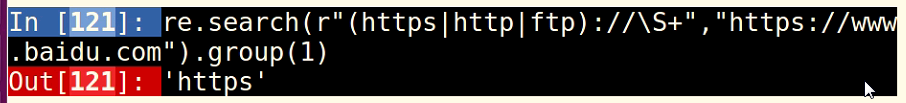
子组作用

1.作为内部整体可以改变某些元字符的行为

In [127]: re.search(r'(ab)+\d+','ababababab456463').group()

Out[127]: 'ababababab456463'

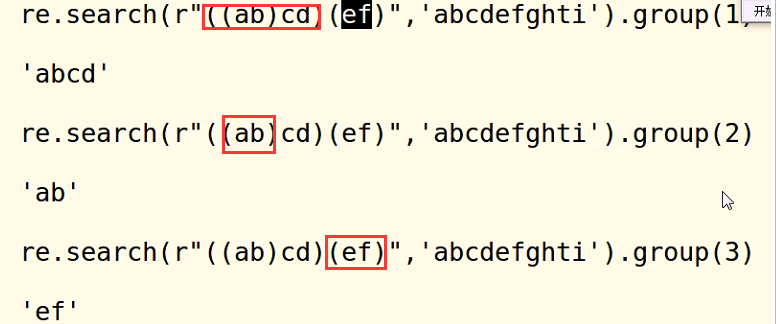
2.子组在某些操作中可以单独提取出匹配内容



子组使用注意事项

\*一个正则表达式中可以有多个子组

\*子组一般由外到内，由左到右称之为第一，第二，第三。。。子组



\*子组不能重叠，嵌套也不宜太多

捕获组和非捕获组

格式：（？P<name> pattern）

In [135]: re.search(r'(?P<aa>cfes)','xdcfesdfg').group('aa')

Out[135]: 'cfes'

作用：可以通过组名更方便获取某组内容

正则表达式设计原则

1. 正确性，能正确匹配到目标内容
2. 排他性，除了要匹配的内容，尽可能不会匹配与到其他内容
3. 全面性，需要对目标的各种情况进行考虑，做到不遗漏

re模块

regex = compile(pattern,flags = 0)

功能：生成正则表达式对象

参数：pattern正则表达式

flags 功能标志位，丰富正则表达式的匹配功能

返回值：返回正则表达式对象

regex.findall(string，pos,endpos)

功能：从目标字符串查找正则匹配内容

参数：string目标字符串

pos匹配目标的起始位置

endpos匹配目标的终止位置

返回值：返回匹配到的内容

如果正则有子组则只返回子组对应的内容

re.findall(pattern，string，flags)

功能：从目标字符串查找正则匹配内容

参数：pattern正则表达式

string目标字符串

flags标志位

返回值：返回匹配到的内容

如果正则有子组则只返回子组对应的内容

re.split(pattern,atring,flags = 0)

功能：根据正则匹配内容切割字符串

参数：pattern string flags

返回值：返回列表，列表中为切割的内容

re.sub(pattern,replaceStr,string,max,flags)

功能：替换正则匹配到的目标子串部分

参数：pattern

repllaceStr ：要替换的内容

strring

max最多替换几次，默认全部替换

flags

返回值：返回替换后的字符串

re.subn(pattern,replaceStr,string,max,flags)

功能：替换正则匹配到的目标子串部分

参数：pattern

repllaceStr ：要替换的内容

strring

max最多替换几次，默认全部替换

flags

返回值：返回一个元组，为实际替换了几处和替换后的字符串

re.finditer(pattern,string,flags)

功能：使用正则表达式匹配目标字符串

参数：pattern string flags

返回值：返回一个迭代对象，迭代到的内容是一个match对象

re.fullmatch(pattern,string,flags)

功能：完全匹配目标字符串

参数：pattern string flags

返回值：返回匹配到的match对象

如果没有匹配成功返回none

re. match(pattern,string,flags)

功能：从开头位置匹配目标字符串

参数：pattern string flags

返回值：返回匹配到的match对象

如果没有匹配成功返回none

search(pattern,string,flags)

功能：正则表达式匹配目标字符串，只匹配第一处

参数：pattern string flags

返回值：返回匹配到的match对象

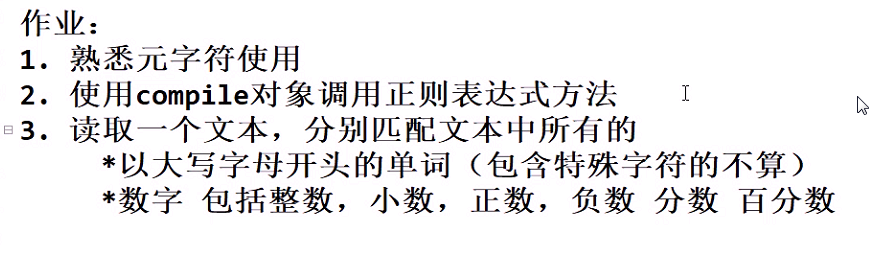
如果没有匹配成功返回none

flags标志位

pattern：正则表达式

groups：有多少子组

groupindex：捕获组形成组名和序列号的字典组名为键，第几组为值



-?\d+.?/?\d\*%?

match对象属性

属性变量

pos )#匹配目标字符串的起始位置

endpos )# 匹配目标字符串的结束位置

re )#正则表达式

string )#目标字符串

lastgroup )#最后一组的组名

lastindex )#最后一组是第几组

属性方法：

span()#匹配内容的起止位置

start()#匹配内容的开始位置

end()#匹配内容的结束位置

group()

功能：获取match对象对应的内容

参数：默认为0表示获取整个正则匹配的内容

如果为序列号或者子组名则为获取某个子组匹配的对应内容

groupdict()#获取捕获组的字典（组名为键，内容为值）

groups()#获取每个子组匹配内容

flags参数使用

re.compile re.findall re.search re.match re.finditer re.fullmatch re.sub re.subn re.split

作用：辅助正则表达式，丰富匹配结果

I == IGNORECASE匹配时忽略字母的大小写

S == DOTALL 作用于元字符.使其可以匹配换行

M == MULTLINE作用于^$使其可以匹配每一行的开头结尾位置

X == VERBOSE 可以给正则加注释

使用多个标志位使用按位或连接

e.g.

flags= re.X|re.I