正则表达式

1 动机

1.1 意义

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。

3.为了快速、方便的解决上述问题，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。本质是由一系列特殊字符和字符组成的子串，这个子串即是正则表达式。这个表达式描述了字符和字符的重复行为，可以匹配一类特征的字符串。

目标：

1.熟练掌握正则表达式符号

2.能够编写一定程度的正则表达式和理解较难的正则表达式

3.能够使用python操作正则表达式

1.2 特点

1.方便进行检索、修改的文本操作

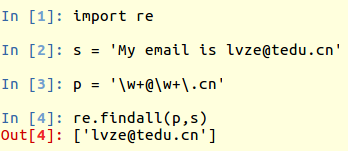
2.支持语言众多

3.使用灵活多样

1.3 python中的正则表达式

python ----》re模块 处理正则表达式

|  |  |
| --- | --- |
| re.findall（pattern,string） | |
| 功能 | 使用正则表达式处理匹配字符串 |
| 参数 | pattern正则表达式  string 目标字符串 |
| 返回值 | 返回一个列表，列表中为匹配到的内容 |



2 正则表达式的字符

元字符(即正则表达式中有特殊含义的符号)

2.1 普通字符

元字符：abc

匹配规则：匹配相应的字符



2.或 元字符 |

匹配|两边任意一个正则表达式的内容



重叠部分不能被匹配



\*|左右两端不要加空格

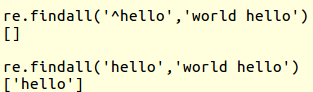
3.匹配单一字符 .

匹配除换行外的任意字符



4.匹配开始位置 ^

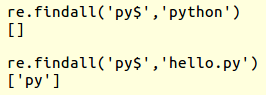
匹配字符串开头的位置



5.匹配结束位置 $

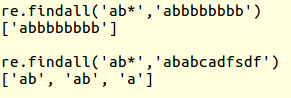
匹配目标字符串的结束位置

py$ --->hello.py



5.匹配重复 \*

匹配前面的正则表达式0个或者多个



a 就是b匹配0个

6.匹配重复 +

匹配前面的正则表达式，1个或者多个



7.匹配重复 ?

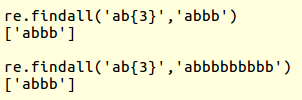
匹配前面的正则表达式0个或1个



8.匹配重复 {n}

匹配指定的重复次数

ab{3}---->abbb



9.匹配重复 {m,n}

匹配前面的正则表达式，m次到n次输出



10.匹配重复字符集合 [字符集]

匹配括号内的任意一个字符

[abc123d] a b c 1 2 3 d

[a-z]

[A-Z]

[0-9]

[123a-zA-Z]匹配123小写a到小写z大写A到大写Z





11.匹配重复字符集合 [^字符集]

匹配除了括号中字符集的任意一个字符

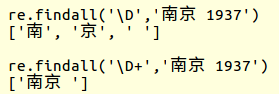


12.匹配任意（非）数字字符 \d \D

\d匹配任意数字字符 [0-9]

\D匹配任意的非数字字符 [^0-9]

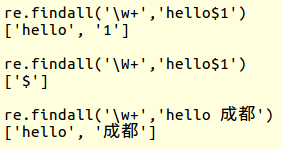




13.匹配（非）普通字符 （数字、字母、下划线）\w \W

\w匹配普通字符 [\_0-9a-zA-Z]

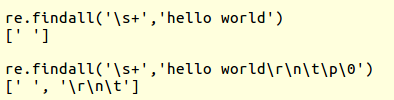
\W匹配非普通字符 [^\_0-9a-zA-Z]



14.匹配（非）空字符 （空格、\r \t \n \0）\s \S

\s匹配空字符

\S任意的非空字符

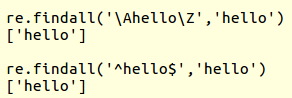




15.匹配起止位置 \A \Z

\A匹配开始位置 ^

\Z匹配结束位置 $







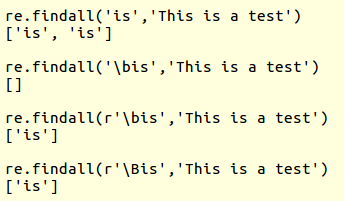
\*绝对匹配：使用开头和结尾位置的元字符将正则表达式放在其中，则目标字符串仅有正则匹配内容时，才能匹配上。

16.匹配（非）单词边界位置 \b \B

普通字符和非普通字符的交界处为单词边界

\b单词边界位置

\B非单词边界





2.2 元字符的总结

匹配单个字符：a . \d \D \w \W \s \S […] [^…]

匹配重复性：\* + ? {n} {m,n}

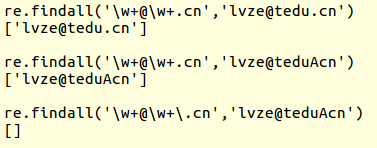
匹配位置：^ $ \A \Z \b \B

其他： | () \

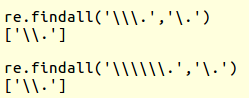
3 正则表达式的转义

正则表达式的特殊符号：

. \* ? $ [] {} () ^ \







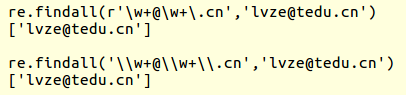
3.1 转义

raw字符串

原生字符串，字符串内容就是本身不进行任何转义

用法：

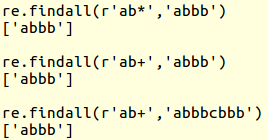
r’’hello’’



4 贪婪与非贪婪

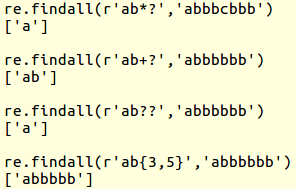
贪婪模式

正则表达式的重复匹配默认总是尽可能多的向后匹配内容



非贪婪模式

尽可能少的匹配内容



贪婪-----------》非贪婪

\*? +? ?? {m,n}?

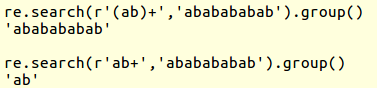
5 正则表达式分组

使用（）可以为一个正则表达式建立子组，子组并不会影响正则表达式的整体匹配内容，子组可以看作是一个内部的整体部分。

5.1 子组的作用

1.子组可以改变正则表达式的重复行为

2.子组在某些操作中可以被单独提取出来





子组使用的注意事项

1.一个正则表达式中，可以有多个子组，从外到内，从左到右分别为第一第二第三……子组

2.子组不存在交叉的情况（ab（c）de）f

捕获组和非捕获组（命名组合非命名组）

格式：

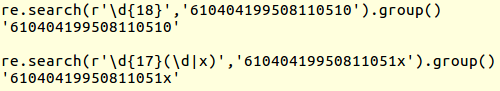
（？P<name>pattern）

1.很多编程接口可以直接通过名字获取子组匹配内容

2.命名组可以被重复调用（?P=name）







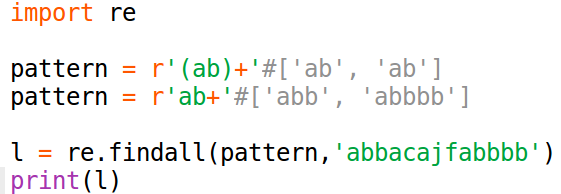
6 正则表达式使用要求

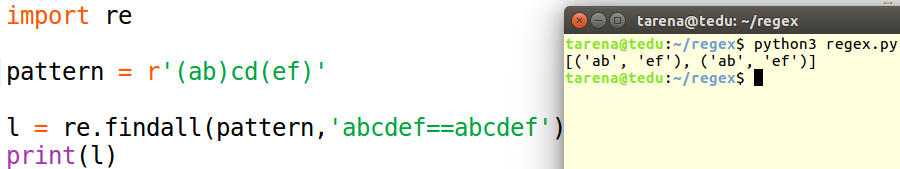
1.唯一性：正则表达式只能匹配目标类别字符串，而不能匹配其他内容

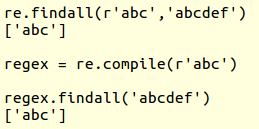
2.准确性：尽可能全面的考虑目标类别的字符串特征，做到不遗漏

7 re模块

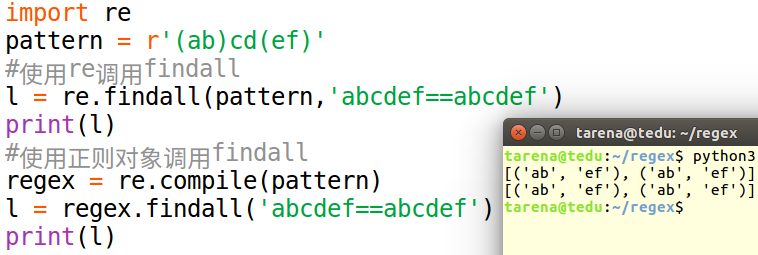
|  |  |
| --- | --- |
| re.compile(pattern,flags=0) | |
| 功能 | 生成正则表达式对象 |
| 参数 | pattern正则表达式  flags功能标志位，提供更丰富的筛选功能 |
| 返回值 | 正则表达式对象 |
| re.findall(pattern,string,flags) | |
| 功能 | 查找正则表达式匹配内容 |
| 参数 | pattern正则表达式  string目标字符串  flags功能标志位，提供更丰富的筛选功能 |
| 返回值 | 将匹配到的内容放入一个列表返回  如果有子组，只能返回子组匹配到的内容 |





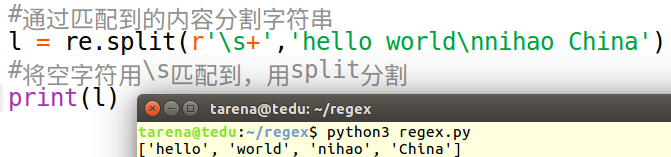


|  |  |
| --- | --- |
| regex.findall(string=None,pos=0,endpos=999) | |
| 功能 | 查找正则表达式匹配内容 |
| 参数 | string目标字符串  pos匹配目标字符串的起始位置  endpos匹配目标字符串的结束位置 |
| 返回值 | 将匹配到的内容放入一个列表返回  如果有子组，只能返回子组匹配到的内容 |

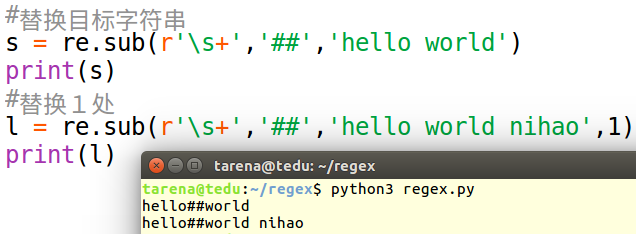




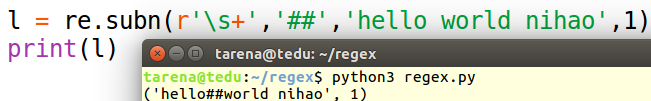
|  |  |
| --- | --- |
| re.split(pattern,string,flags=0) | |
| 功能 | 通过正则表达式分割目标字符串 |
| 参数 | pattern正则表达式  string目标字符串 |
| 返回值 | 返回分割后的字符串列表 |



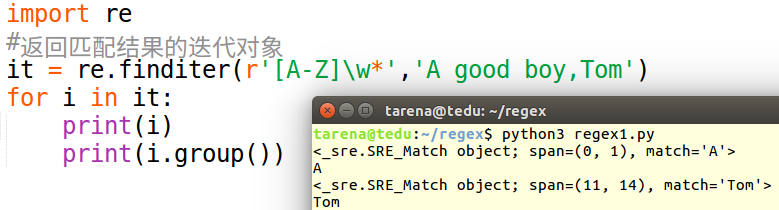
|  |  |
| --- | --- |
| re.sub(pattern,replaceStr,string,max,flags) | |
| 功能 | 替换正则表达式匹配到的内容 |
| 参数 | pattern正则表达式  replaceStr要替换的内容  string目标字符串  max最多替换几处 |
| 返回值 | 返回替换后的字符串 |



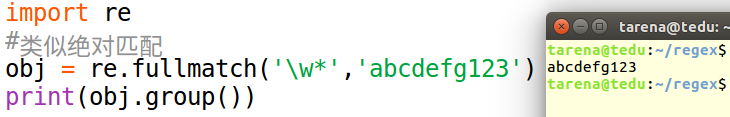
|  |  |
| --- | --- |
| re.subn(pattern,replaceStr,string,max,flags) | |
| 功能 | 替换正则表达式匹配到的内容 |
| 参数 | pattern正则表达式  replaceStr要替换的内容  string目标字符串  max最多替换几处 |
| 返回值 | 返回替换后的字符串和实际替换的个数 |



|  |  |
| --- | --- |
| re.finditer(pattern,string,flags) | |
| 功能 | 使用正则表达式匹配内容 |
| 参数 | pattern正则表达式  string目标字符串 |
| 返回值 | 返回匹配到的内容的迭代器对象 |



|  |  |
| --- | --- |
| re.fullmatch(pattern,string,flags) | |
| 功能 | 完全匹配目标字符串（从头到尾不能有别的） |
| 参数 | pattern正则表达式  string目标字符串 |
| 返回值 | match对象，匹配不到返回None |



|  |  |
| --- | --- |
| re.match(pattern,string,flags) | |
| 功能 | 从开始部分匹配向后匹配（开头必须是目标） |
| 参数 | pattern正则表达式  string目标字符串 |
| 返回值 | match对象，匹配不到返回None |

|  |  |
| --- | --- |
| re.search(pattern,string,flags) | |
| 功能 | 匹配目标字符串（任意位置搜索第一次出现的情况） |
| 参数 | pattern正则表达式  string目标字符串 |
| 返回值 | search对象，匹配不到返回None |

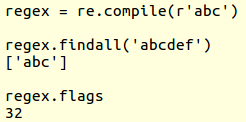


\*由于fullmatch match search函数匹配不到会返回None，而None没有match对象属性，所以往往需要用异常判断处理。

8 正则对象其他属性

flags：标志位

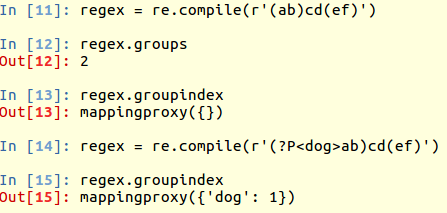
pattern：正则表达式



group：有多少个子组

groupindex：捕获组形成的字典

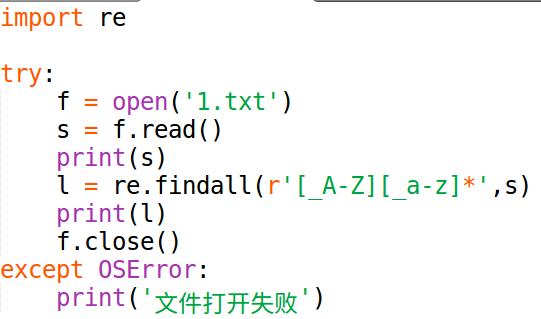
组名为键，第几组为值



作业：

1.读取一个文本，将文本中所有以大写字母开头的单词匹配出来。

2.熟练元字符的使用

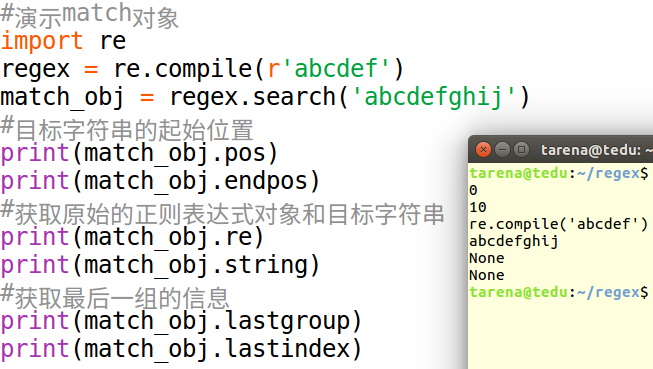


正则表达式2

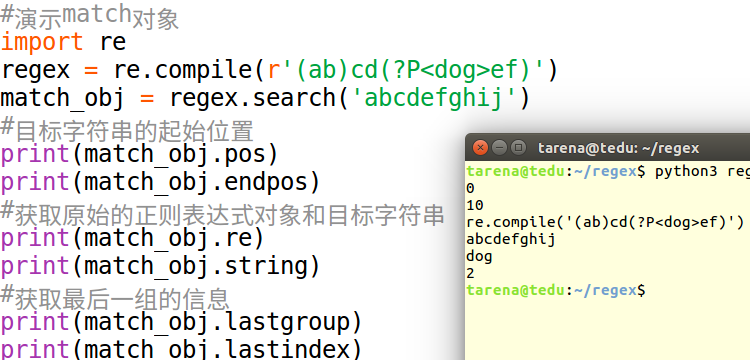
1 match对象属性变量

1.1 意义

|  |  |
| --- | --- |
| pos | 目标字符串的开头位置 |
| endpos | 目标字符串的结束位置 |
| re | 正则表达式对象 |
| string | 目标字符串 |
| lastgroup | 最后一组名字 |
| lastindex | 最后一组是第几组 |

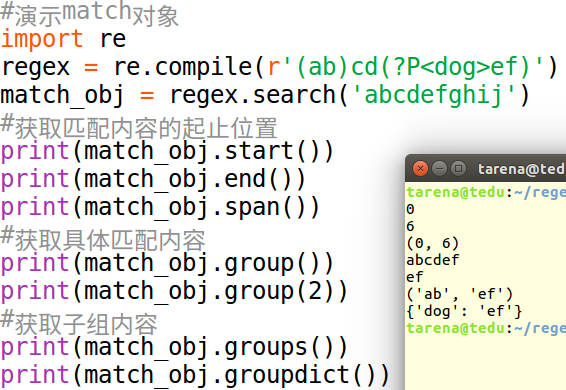


起名后lastgroup显示名字



2 属性方法

|  |  |
| --- | --- |
| start() | 获取匹配到的内容开始位置 |
| end() | 获取匹配到的内容结束位置 |
| span() | 获取匹配到的内容开始、结束位置 |
| group(n) | 功能:获取match对象对应匹配到的内容  参数:默认为0表示获取正则表达式整体的匹配内容。如果赋值1、2、3…则表示获取某个子组的匹配内容。  返回值:返回匹配字符串 |
| groups() | 获取所有子组匹配内容 |
| groupdict() | 将所有补获组内容形成一个字典 |



3 flags参数

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

|  |  |
| --- | --- |
| A==ASCII |  |
| S==DOTALL | 元字符 . 可以匹配换行\n |
| I==IGNORECASE | 不区分大小写 |
| L==LOCALE |  |
| M==MULTILINE | 元字符^$可以匹配每一行的开头、结尾的位置 |
| T==TEMPLATE |  |
| U==UNICODE |  |
| X==VERBOSE | 可以给正则添加注释 |

\*同时使用多个flags用 | 隔开

re.I | re.S



