**1.2.1字符串操作函数**

字符串操作（string替换、删除、截取、复制、连接、比较、查找、包含、大小写转换、分割等）

python print 不换行（在后面加上,end=''），print(string,end='')

Python split()通过指定分隔符对字符串进行切片，如果参数num 有指定值，则仅分隔 num 个子字符串

split()方法语法：str.split(str="", num=string.count(str)).

参数

str -- 分隔符，默认为空格。

num -- 分割次数。

返回值

返回分割后的字符串列表。

去空格及特殊符号

s.strip().lstrip().rstrip(',')

复制字符串

#strcpy(sStr1,sStr2)

sStr1 = 'strcpy'

sStr2 = sStr1

sStr1 = 'strcpy2'

print sStr2

连接字符串

#strcat(sStr1,sStr2)

sStr1 = 'strcat'

sStr2 = 'append'

sStr1 += sStr2

print sStr1

查找字符

#strchr(sStr1,sStr2)

# < 0 为未找到

sStr1 = 'strchr'

sStr2 = 's'

nPos = sStr1.index(sStr2)

print nPos

比较字符串

#strcmp(sStr1,sStr2)

sStr1 = 'strchr'

sStr2 = 'strch'

print cmp(sStr1,sStr2)

扫描字符串是否包含指定的字符

#strspn(sStr1,sStr2)

sStr1 = '12345678'

sStr2 = '456'

#sStr1 and chars both in sStr1 and sStr2

print len(sStr1 and sStr2)

字符串长度

#strlen(sStr1)

sStr1 = 'strlen'

print len(sStr1)

将字符串中的大小写转换

#strlwr(sStr1)

sStr1 = 'JCstrlwr'

sStr1 = sStr1.upper()

#sStr1 = sStr1.lower()

print sStr1

追加指定长度的字符串

#strncat(sStr1,sStr2,n)

sStr1 = '12345'

sStr2 = 'abcdef'

n = 3

sStr1 += sStr2[0:n]

print sStr1

字符串指定长度比较

#strncmp(sStr1,sStr2,n)

sStr1 = '12345'

sStr2 = '123bc'

n = 3

print cmp(sStr1[0:n],sStr2[0:n])

复制指定长度的字符

#strncpy(sStr1,sStr2,n)

sStr1 = ''

sStr2 = '12345'

n = 3

sStr1 = sStr2[0:n]

print sStr1

将字符串前n个字符替换为指定的字符

#strnset(sStr1,ch,n)

sStr1 = '12345'

ch = 'r'

n = 3

sStr1 = n \* ch + sStr1[3:]

print sStr1

扫描字符串

#strpbrk(sStr1,sStr2)

sStr1 = 'cekjgdklab'

sStr2 = 'gka'

nPos = -1

for c in sStr1:

if c in sStr2:

nPos = sStr1.index(c)

break

print nPos

翻转字符串

#strrev(sStr1)

sStr1 = 'abcdefg'

sStr1 = sStr1[::-1]

print sStr1

查找字符串

#strstr(sStr1,sStr2)

sStr1 = 'abcdefg'

sStr2 = 'cde'

print sStr1.find(sStr2)

分割字符串

#strtok(sStr1,sStr2)

sStr1 = 'ab,cde,fgh,ijk'

sStr2 = ','

sStr1 = sStr1[sStr1.find(sStr2) + 1:]

print sStr1

#或者

s = 'ab,cde,fgh,ijk'

print(s.split(','))

连接字符串

delimiter = ','

mylist = ['Brazil', 'Russia', 'India', 'China']

print delimiter.join(mylist)

截取字符串

str = '0123456789'

print str[0:3] #截取第一位到第三位的字符

print str[:] #截取字符串的全部字符

print str[6:] #截取第七个字符到结尾

print str[:-3] #截取从头开始到倒数第三个字符之前

print str[2] #截取第三个字符

print str[-1] #截取倒数第一个字符

print str[::-1] #创造一个与原字符串顺序相反的字符串

print str[-3:-1] #截取倒数第三位与倒数第一位之前的字符

print str[-3:] #截取倒数第三位到结尾

print(str[:-5:-3]) #逆序截取，输出96

print(str[:-4:-2])#逆序截取，输出97

print(str[:-6:-2])#逆序截取，输出975

**1.2.2正则表达式**

. 匹配除换行符以外的任意字符

\w 匹配字母或数字或下划线或汉字

\s 匹配任意的空白符

\d 匹配数字

\b 匹配单词的开始或结束

^ 匹配字符串的开始

$ 匹配字符串的结束

代码/语法 说明

\* 重复零次或更多次

+ 重复一次或更多次

? 重复零次或一次

{n} 重复n次

{n,} 重复n次或更多次

{n,m} 重复n到m次

[aeiou]就匹配任何一个英文元音字母，[.?!]匹配标点符号(.或?或!)。

[0-9]代表的含意与\d就是完全一致的：一位数字；同理[a-z0-9A-Z\_]也完全等同于\w

\ba\w\*\b匹配以字母a开头的单词——先是某个单词开始处(\b)，然后是字母a,然后是任意数量的字母或数字(\w\*)，最后是单词结束处(\b)。

\d+匹配1个或更多连续的数字。这里的+是和\*类似的元字符，不同的是\*匹配重复任意次(可能是0次)，而+则匹配重复1次或更多次。

\b\w{6}\b 匹配刚好6个字符的单词。

反义

有时需要查找不属于某个能简单定义的字符类的字符。比如想查找除了数字以外，其它任意字符都行的情况，这时需要用到反义：

表3.常用的反义代码 代码/语法 说明

\W 匹配任意不是字母，数字，下划线，汉字的字符

\S 匹配任意不是空白符的字符

\D 匹配任意非数字的字符

\B 匹配不是单词开头或结束的位置

[^x] 匹配除了x以外的任意字符

[^aeiou] 匹配除了aeiou这几个字母以外的任意字符

例子：\S+匹配不包含空白符的字符串。

]+>匹配用尖括号括起来的以a开头的字符串。

常用的处理选项 名称 说明

IgnoreCase(忽略大小写) 匹配时不区分大小写。

Multiline(多行模式) 更改^和$的含义，使它们分别在任意一行的行首和行尾匹配，而不仅仅在整个字符串的开头和结尾匹配。(在此模式下,$的精确含意是:匹配\n之前的位置以及字符串结束前的位置.)

Singleline(单行模式) 更改.的含义，使它与每一个字符匹配（包括换行符\n）。

IgnorePatternWhitespace(忽略空白) 忽略表达式中的非转义空白并启用由#标记的注释。

ExplicitCapture(显式捕获) 仅捕获已被显式命名的组。

尚未详细讨论的语法 代码/语法 说明

\a 报警字符(打印它的效果是电脑嘀一声)

\b 通常是单词分界位置，但如果在字符类里使用代表退格

\t 制表符，Tab

\r 回车

\v 竖向制表符

\f 换页符

\n 换行符

\e Escape

\0nn ASCII代码中八进制代码为nn的字符

\xnn ASCII代码中十六进制代码为nn的字符

\unnnn Unicode代码中十六进制代码为nnnn的字符

\cN ASCII控制字符。比如\cC代表Ctrl+C

\A 字符串开头(类似^，但不受处理多行选项的影响)

\Z 字符串结尾或行尾(不受处理多行选项的影响)

\z 字符串结尾(类似$，但不受处理多行选项的影响)

\G 当前搜索的开头

\p{name} Unicode中命名为name的字符类，例如\p{IsGreek}

(?>exp) 贪婪子表达式

(?-exp) 平衡组

(?im-nsx:exp) 在子表达式exp中改变处理选项

(?im-nsx) 为表达式后面的部分改变处理选项

(?(exp)yes|no) 把exp当作零宽正向先行断言，如果在这个位置能匹配，使用yes作为此组的表达式；否则使用no

(?(exp)yes) 同上，只是使用空表达式作为no

(?(name)yes|no) 如果命名为name的组捕获到了内容，使用yes作为表达式；否则使用no

(?(name)yes) 同上，只是使用空表达式作为no

1. 数字相关

（1）非负整数

import re

nStr = “123″

p = re.compile(’^\d+$’,re.S) 非负整数（正整数与零）

if p.match(nStr):

print “exists”

else:

print “not”

（2）正整数

import re

nStr = “123″

p = re.compile(’^[0-9]\*[1-9][0-9]\*$’,re.S) 正整数（不包括零在内）

if p.match(nStr):

print “exists”

else:

print “not”

（3） 非正整数（负整数+0）

import re

nStr = “-123″

p = re.compile(’^((-\d+)|(0+))$’,re.S)

if p.match(nStr):

print “exists”

else:

print “not”

（4）负整数

import re

nStr = “-123″

p = re.compile(’^-[0-9]\*[1-9][0-9]\*$’,re.S)

if p.match(nStr):

print “exists”

else:

print “not”

（5）整数

import re

nStr = “123″

p = re.compile(’^-?\d+$’,re.S)

if p.match(nStr):

print “exists”

else:

print “not”

（6）非负浮点数（正浮点数 + 0）

import re

nStr = “0.123″

p = re.compile(’^\d+(\.\d+)?$’,re.S)

if p.match(nStr):

print “exists”

else:

print “not”

(7)正浮点数

import re

nStr = “0.123″

p = re.compile(’^(([0-9]+\.[0-9]\*[1-9][0-9]\*)|([0-9]\*[1-9][0-9]\*\.[0-9]+)|([0-9]\*[1-9][0-9]\*))$’,re.S)

if p.match(nStr):

print “exists”

else:

print “not”

(9)非正浮点数（负浮点数 + 0）

import re

nStr = “-0.123″

p = re.compile(’^((-\d+(\.\d+)?)|(0+(\.0+)?))$’,re.S)

if p.match(nStr):

print “exists”

else:

print “not”

(10)浮点数

import re

nStr = “-0.123″

p = re.compile(’^(-?\d+)(\.\d+)?$’,re.S)

if p.match(nStr):

print “exists”

else:

print “not”

2 字符相关

2.1 26个英文字母组成的字符串

import re

nStr = “abck”

p = re.compile(’^[A-Za-z]+$’,re.S)

if p.match(nStr):

print “exists”

else:

print “not”

如果是大写的话就是：^[A-Z]+$

如果是小写的话就是：^[a-z]+$

如果是数字与字母组合：^[A-Za-z0-9]+$

如果由数字、26个字母、或下划线组成的：^\w+$

3 .HTML相关

(1) 匹配<script>标签的

#coding:utf-8

import re

nStr = “<script src=’…js’></script>”

p = re.compile(’<\s\*script[^>]\*>[^<]\*<\s\*/\s\*script\s\*>’,re.I)

if p.match(nStr):

print “exists”

else:

print “not”

(2) 匹配<style>标签

#coding:utf-8

import re

nStr = “<style></style>”

p = re.compile(’<\s\*style[^>]\*>[^<]\*<\s\*/\s\*style\s\*>’,re.I)

if p.match(nStr):

print “exists”

else:

print “not”

(3) 匹配HTML标签

#coding:utf-8

import re

nStr = “<div>”

p = re.compile(’</?\w+[^>]\*>’,re.I)

if p.match(nStr):

print “exists” 输出

else:

print “not”

4 .URL相关

(1) 匹配EMAIL地址

import re

nStr = “hkebao@126.com”

p = re.compile(’^[\w-]+(\.[\w-]+)\*@[\w-]+(\.[\w-]+)+$’,re.S)

if p.match(nStr):

print “exists”

else:

print “not”

(2)匹配URL

import re

nStr = “http://localhost”

p = re.compile(’^[a-zA-z]+://(\w+(-\w+)\*)(\.(\w+(-\w+)\*))\*(\?\S\*)?$’,re.S)

if p.match(nStr):

print “exists”

else:

print “not”

re.match

re.match 尝试从字符串的开始匹配一个模式，如：下面的例子匹配第一个单词。

1. import re

2.

3. text = “JGood is a handsome boy, he is cool, clever, and so on…”

4. m = re.match(r”(\w+)\s”, text)

5. if m:

6. print m.group(0), ‘\n’, m.group(1)

7. else:

8. print ‘not match’

re.search

　　re.search函数会在字符串内查找模式匹配,只到找到第一个匹配然后返回，如果字符串没有匹配，则返回None。

1. import re

2.

3. text = “JGood is a handsome boy, he is cool, clever, and so on…”

4. m = re.search(r’\shan(ds)ome\s’, text)

5. if m:

6. print m.group(0), m.group(1)

7. else:

8. print ‘not search’

re.match与re.search的区别：re.match只匹配字符串的开始，如果字符串开始不符合正则表达式，则匹配失败，函数返回None；而re.search匹配整个字符串，直到找到一个匹配。

re.sub

　　re.sub用于替换字符串中的匹配项。下面一个例子将字符串中的空格 ‘ ‘ 替换成 ‘-’ :

1. import re

2.

3. text = “abcde”

4. print re.sub(r’\s+’, ‘-’, text)

re.sub(r’\s’, lambda m: ‘[' + m.group(0) + ']‘, text, 0)；将字符串中的空格’ ‘替换为’[ ]‘。

re.split

　　可以使用re.split来分割字符串，如：re.split(r’\s+’, text)；将字符串按空格分割成一个单词列表。

re.findall

　　re.findall可以获取字符串中所有匹配的字符串。如：re.findall(r’\w\*oo\w\*’, text)；获取字符串中，包含’oo’的所有单词。

Python 匹配任意字符（包括换行符）的正则表达式写法

(.\*)

结果运行之后才发现，无法获得换行之后的文本。于是查了一下手册，才发现正则表达式中，“.”(点符号)匹配的是除了换行符“\n”以外的所有字符。

以下为正确的正则表达式匹配规则：

([\s\S]\*)

同时，也可以用 “([\d\D]\*)”、“([\w\W]\*)” 来表示。

在文本文件里, 这个表达式可以匹配所有的英文

/[ -~]/