《Python统计计算》

（秋季学期）

翟祥
北京林业大学

编程基础

第4章 字符串与正则表达式

- 最早的字符串编码是美国标准信息交换码ASCII，仅对10个数字、26个大写字英文字母、26个小写字英文字母及一些其它符号进行了编码。ASCII采用8位即1个字节，因此最多只能对256个字符进行编码。
- 随着信息技术的发展，各国的文字都需要进行编码，常见的编码有UTF-8，GB2312，GBK，CP936。
- UTF-8编码是国际通用的编码，以8位，即1字节表示英语(兼容ASCII)，1-3个字节表示中文及其它语言，UTF-8对全世界所有国家需要用到的字符进行了编码。
- 采用不同的编码意味着把同一字符存入文件时，写入的内容可能不同。

- GB2312是中国制定的中文编码，使用1个字节表示英语，2个字节表示中文；
- GBK是GB2312的扩充；
- CP936是微软在GBK基础上完成的编码；
- GB2312、GBK和CP936都是使用2个字节表示中文，UTF-8使用1-3个字节表示中文；
- Unicode是不同编码之间转换的基础。
- 在Windows平台上，input()函数从键盘输入的字符串默认为GBK编码，而Python程序的字符串编码使用#coding指定，如

```
#coding=utf-8
#coding:GBK
#-*-coding:utf-8 -*-
```

- Python 2.x对中文支持不够，需要在不同的编码之间转换。
- Python 2.7.11环境：

```
>>> s1='中国'
>>> s1
'\xd6\xd0\xb9\xfa'
>>> len(s1)
4
>>> s2=s1.decode('GBK') #将GBK编码的字符串转变成unicode编码
>>> s2
u'\u4e2d\u56fd'
>>> len(s2)
2
>>> s3=s2.encode('UTF-8')  #unicode编码转变为UTF-8编码
>>> s3
'\xe4\xb8\xad\xe5\x9b\xbd'
>>> len(s3)
6
>>> print s1,s2,s3
中国 中国 中国
```

- Python 3.x完全支持中文，无论是一个数字、英文字母或一个汉字，都按照一个字符对待和处理
- Python3.5.1环境：

```
>>> s = '中国山东烟台'
>>> len(s)
6
>>> s = 'SDIBT'
>>> len(s)
5
>>> s = '中国山东烟台SDIBT'
>>> len(s)
11
```

默认编码的查询与设置

- 查询系统默认编码可以在解释器中输入以下命令：

```
>>>import sys
>>> sys.getdefaultencoding()
```

- 设置默认编码时使用：

```
>>>import sys
>>>sys.setdefaultencoding('utf8')
```

4.1 字符串

- 在Python中，字符串也属于序列类型，除了支持序列通用方法（包括分片操作）以外，还支持特有的字符串操作方法。
- 字符串属于不可变序列类型

```
>>> testString = 'good'
>>> id(testString)
19046560
>>> testString[0] = 'b'

Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    testString[0] = 'b'
TypeError: 'str' object does not support item assignment
>>> testString = 'well'
>>> id(testString)
19047808
```

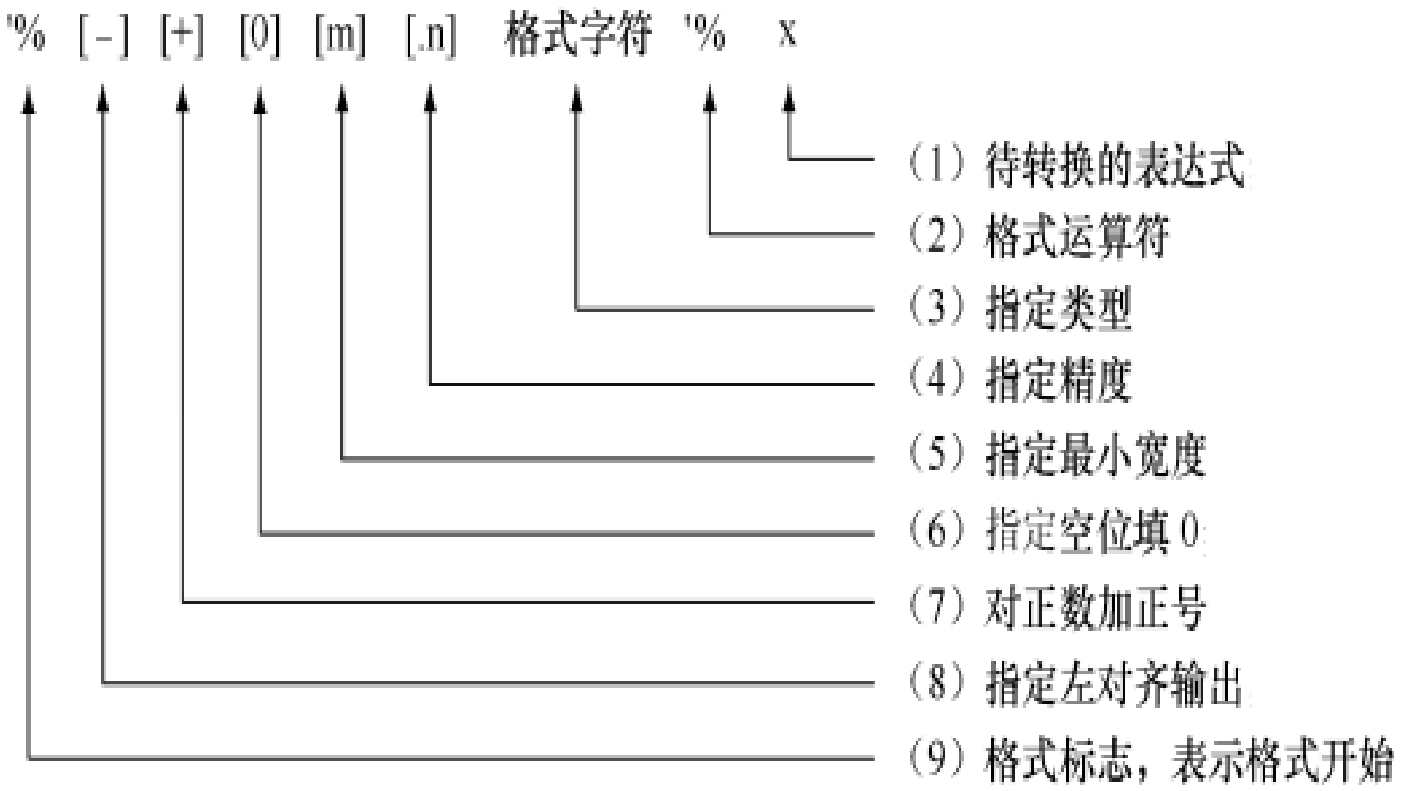
4.1 字符串

- Python字符串驻留机制：对于短字符串，将其赋值给多个不同的对象时，内存中只有一个副本，多个对象共享该副本。长字符串不遵守驻留机制。

```
>>> a='1234'
>>> b='1234'
>>> id(a)==id(b)
True
>>> a='1234'*50
>>> b='1234'*50
>>> id(a)==id(b)
False
>>> id(a)
55130160
>>> id(b)
55130416
```

- 判断一个变量s是否为字符串，应使用isinstance(s,basestring)。在Python3之前，字符串有str和unicode两种，其基类都是basestring。在Python3之后合二为一了，只提供str类型。
- 在Python3中，程序源文件默认为UTF-8编码，全面支持中文。

4.1.1 字符串格式化


% [-] [+] [0] [m] [.n] 格式字符 % x
(1) 待转换的表达式
(2) 格式运算符
(3) 指定类型
(4) 指定精度
(5) 指定最小宽度
(6) 指定空位填 0
(7) 对正数加正号
(8) 指定左对齐输出
(9) 格式标志，表示格式开始

4.1.1 字符串格式化

常用格式字符

格式字符	说明
%s	字符串 (采用str()的显示)
%r	字符串 (采用repr()的显示)
%c	单个字符
%b	二进制整数
%d	十进制整数
%i	十进制整数
%o	八进制整数
%x	十六进制整数
%e	指数 (基底写为e，以10为底数)
%E	指数 (基底写为E，以10为底数)
%f、%F	浮点数
%g	指数(e)或浮点数 (根据显示长度)
%G	指数(E)或浮点数 (根据显示长度)
%%	字符"%"

4.1.1 字符串格式化

```
>>> x=1235
>>> so="%o" % x
>>> so
"2323"
>>> sh="%x" % x
>>> sh
"4d3"
>>> se="%e" % x
>>> se
"1.235000e+03"
>>> chr(ord("3")+1)
"4"
>>> "%s"%65
"65"
>>> "%s"%65333
"65333"
>>> "%d"%"555"
Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    "%d"%"555"
TypeError: %d format: a number is required, not str
```

4.1.1 字符串格式化

- 使用format方法进行格式化

```
>>> print("The number {0:,} in hex is: {0:#x}, the number {1} in oct
    is {1:#o}".format(5555,55))
The number 5,555 in hex is: 0x15b3, the number 55 in oct is 0o67

>>> print("The number {1:,} in hex is: {1:#x}, the number {0} in oct
    is {0:#o}".format(5555,55))
The number 55 in hex is: 0x37, the number 5555 in oct is 0o12663

>>> print("my name is {name}, my age is {age}, and my QQ is
    {qq}".format(name = "Dong Fuguo",age = 37,qq = "306467355"))
my name is Dong Fuguo, my age is 37, and my QQ is 306467355

>>> position = (5,8,13)
>>> print("X:{0[0]};Y:{0[1]};Z:{0[2]}".format(position))
X:5;Y:8;Z:13
```

```
>>> p1=(5,8,13)
>>> p2=(3,6,9)
>>> print("X:{0[0]},{1[0]};Y:{0[1]},{1[1]};Z:{0[2]},{1[2]}".format(p1,p2))
X:5,3;Y:8,6;Z:13,9
```

```
>>> weather = [("Monday","rain"),("Tuesday","sunny"),("Wednesday",
    "sunny"),("Thursday","rain"),("Friday","Cloudy")]
>>> formatter = "Weather of '{0[0]}' is '{0[1]}'".format
>>> for item in map(formatter,weather):
        print(item)

Weather of 'Monday' is 'rain'
Weather of 'Tuesday' is 'sunny'
Weather of 'Wednesday' is 'sunny'
Weather of 'Thursday' is 'rain'
Weather of 'Friday' is 'Cloudy'
```

4.1.2 字符串常用方法

- find( )、rfind()、index()、rindex()、count()

find()和rfind方法分别用来查找一个字符串在另一个字符串指定范围（默认是整个字符串）中首次和最后一次出现的位置，如果不存在则返回-1；index()和rindex()方法用来返回一个字符串在另一个字符串指定范围中首次和最后一次出现的位置，如果不存在则抛出异常；count()方法用来返回一个字符串在另一个字符串中出现的次数。

```
string.find(str, beg=0, end=len(string))
```

4.1.2 字符串常用方法

```
>>> s="apple,peach,banana,peach,pear"
>>> s.find("peach")
6

>>> s.find("peach",7)
19

>>> s.find("peach",7,20)
-1

>>> s.rfind('p')
25

>>> s.index('p')
1

>>> s.index('pe')
6
```

```
>>> s.index('pear')
25

>>> s.index('ppp')
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in
<module>
    s.index('ppp')
ValueError: substring not found

>>> s.count('p')
5

>>> s.count('pp')
1

>>> s.count('ppp')
0
```

4.1.2 字符串常用方法

- split()、rsplit()、partition()、rpartition()

split()和rsplit()方法分别用来以指定字符为分隔符，将字符串从左端和右端开始将其分割成多个字符串，并返回包含分割结果的列表；

partition()和rpartition()用来以指定字符串为分隔符将原字符串分割为3部分，即分隔符前的字符串、分隔符字符串、分隔符后的字符串，如果指定的分隔符不在原字符串中，则返回原字符串和两个空字符串。

4.1.2 字符串常用方法

```
>>> s="apple,peach,banana,pear"
>>> li=s.split(",")
>>> li
["apple", "peach", "banana", "pear"]
>>> s.partition(',')
('apple', ',', 'peach,banana,pear')
>>> s.rpartition(',')
('apple,peach,banana', ',', 'pear')
>>> s.rpartition('banana')
('apple,peach,', 'banana', ',pear')
>>>s = "2014-10-31"
>>> t=s.split("-")
>>> print(t)
['2014', '10', '31']
>>> print(list(map(int, t)))
[2014, 10, 31]
```

4.1.2 字符串常用方法

对于split()和rsplit()方法，如果不指定分隔符，则字符串中的任何空白符号（包括空格、换行符、制表符等等）都将被认为是分隔符，返回包含最终分割结果的列表。

```
>>> s = 'hello world \n\n My name is Dong   '
>>> s.split()
['hello', 'world', 'My', 'name', 'is', 'Dong']
>>> s = '\n\nhello world \n\n\n My name is Dong   '
>>> s.split()
['hello', 'world', 'My', 'name', 'is', 'Dong']
>>> s = '\n\nhello\t\t world \n\n\n My name\t is Dong   '
>>> s.split()
['hello', 'world', 'My', 'name', 'is', 'Dong']
```

4.1.2 字符串常用方法

split()和rsplit()方法还允许指定最大分割次数，例如：

```
>>> s = '\n\nhello\t\t world \n\n\n My name is Dong   '
>>> s.split(None,1) #或略了开头的空
['hello', 'world \n\n\n My name is Dong   ']
>>> s.rsplit(None,1)
['\n\nhello\t\t world \n\n\n My name is', 'Dong']
>>> s.split(None,2)
['hello', 'world', 'My name is Dong   ']
>>> s.rsplit(None,2)
['\n\nhello\t\t world \n\n\n My name', 'is', 'Dong']
>>> s.split(None,5)
['hello', 'world', 'My', 'name', 'is', 'Dong   ']
>>> s.split(None,6)
['hello', 'world', 'My', 'name', 'is', 'Dong']
```

4.1.2 字符串常用方法

- 字符串联接join( )

例子：

```
>>> li=["apple", "peach", "banana", "pear"]
>>> sep=","
>>> s=sep.join(li)
>>> s
"apple,peach,banana,pear"
```

不推荐使用+连接字符串，优先使用join()方法

```
#CompareJoinAndPlusForStringConnection.py
import timeit
strlist=['This is a long string that will not keep in memory.' for n in xrange(10)]

def use_join():
    return ''.join(strlist)

def use_plus():
    result=''
    for strtemp in strlist:
        result=result+strtemp
    return result

if __name__ == '__main__':
    times=1000
    jointimer = timeit.Timer('use_join()','from __main__ import use_join')
    print 'time for join:',jointimer.timeit(number=times)
    plustimer = timeit.Timer('use_plus()','from __main__ import use_plus')
    print 'time for plus:',plustimer.timeit(number=times)
```

4.1.2 字符串常用方法

- lower()、upper()、capitalize()、title()、swapcase()

这几个方法分别用来将字符串转换为小写、大写字符串、将字符串首字母变为大写、将每个单词的首字母变为大写以及大小写互换。

```
>>> s="What is Your Name?"
>>> s2=s.lower()
>>> s2
"what is your name?"
>>> s.upper()
"WHAT IS YOUR NAME?"
>>> s2.capitalize()
"What is your, name?"
>>> s.title()
'What Is Your Name?'
>>> s.swapcase()
'wHAT IS yOUR nAME?'
```

4.1.2 字符串常用方法

- 查找替换replace( )

```
>>> s="中国，中国"
>>> print s
中国，中国
>>>s2=s.replace("中国", "中华人民共和国")
>>> print s2
中华人民共和国，中华人民共和国
```

4.1.2 字符串常用方法

- 生成映射表函数maketrans和按映射表关系转换字符串函数translate

```
>>> import string
>>> table=string.maketrans("abcdef123","uvwxyz@#$")
>>> s="Python is a greate programming language. I like it!"
>>> s.translate(table)
"Python is u gryuty progrumming lunguugy. I liky it!"
>>> s.translate(table,"gtm") #第二个参数表示要删除的字符
"Pyhon is u ryuy proruin lunuuy. I liky i!"
```

4.1.2 字符串常用方法

- strip()、rstrip()、lstrip()

这几个方法分别用来删除两端、右端或左端的空格或连续的指定字符。

```
>>> s=" abc  "
>>> s2=s.strip( )
>>> s2
"abc"
>>> "aaaassddf".strip("a")
"ssddf"
>>> "aaaassddf".strip("af")
"ssdd"
>>> "aaaassddfaaa".rstrip("a")
'aaaassddf'
>>> "aaaassddfaaa".lstrip("a")
'ssddfaaa'
```

4.1.2 字符串常用方法

内置函数eval()

```
>>> eval("3+4")
7
>>> a = 3
>>> b = 5
>>> eval('a+b')
8
>>> import math
>>> eval('help(math.sqrt)')
Help on built-in function sqrt in module math:
sqrt(...)
    sqrt(x)
Return the square root of x.
>>> eval('math.sqrt(3)')
1.7320508075688772
```

4.1.2 字符串常用方法

```
>>> eval('aa')
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    eval('aa')
  File "<string>", line 1, in <module>
NameError: name 'aa' is not defined

>>> a = input("Please input a value:")
Please input a value:"__import__('os').startfile(r'C:\Windows\notepad.exe')"
>>> eval(a)
>>> eval("__import__('os').system('md testtest')")
```

4.1.2 字符串常用方法

- 成员判断

```
>>> "a" in "abcde"
True
>>> "j" in "abcde"
False
```

- s.startswith(t)、s.endswith(t)

判断字符串是否以指定字符串开始或结束

```
>>> import os
>>> [filename for filename in os.listdir(r'c:\\') \
      if filename.endswith(('.bmp','.jpg','.gif'))]
```

4.1.2 字符串常用方法

- center()、ljust()、rjust()

返回指定宽度的新字符串，原字符串居中、左对齐或右对齐出现在新字符串中，如果指定宽度大于字符串长度，则使用指定的字符（默认为空格）进行填充。

```
>>> 'Hello world!'.center(20)
'    Hello world!    '
>>> 'Hello world!'.center(20,'=')
'====Hello world!===='
>>> 'Hello world!'.ljust(20,'=')
'Hello world!========'
>>> 'Hello world!'.rjust(20,'=')
'========Hello world!'
```

4.1.3 字符串常量

```
>>> import string
>>> string.digits
'0123456789'
>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
>>> string.letters
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
>>> string.printable
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQ
   RSTUVWXYZ!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~
   \t\n\r\x0b\x0c'
>>> string.lowercase
'abcdefghijklmnopqrstuvwxyz'
>>> string.uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

4.1.3 字符串常量

- 随机密码生成原理

```
>>> import string
>>> x = string.digits + string.ascii_letters + string.punctuation
>>> x
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXY
Z!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
>>> import random
>>> ''.join([random.choice(x) for i in range(8)])
'H\\{.#=)g'
>>> ''.join([random.choice(x) for i in range(8)])
'(CrZ[44M'
>>> ''.join([random.choice(x) for i in range(8)])
'o_?[M>iF'
>>> ''.join([random.choice(x) for i in range(8)])
'n<[I)5V@'
```

4.1.4 可变字符串

- 在Python中，字符串属于不可变对象，不支持原地修改，如果需要修改其中的值，只能重新创建一个新的字符串对象。然而，如果确实需要一个支持原地修改的unicode数据对象，可以使用io.StringIO对象或array模块。

```
>>> import io
>>> s = "Hello, world “
>>> id(s)
55086960
>>> sio = io.StringIO(s)
>>> id(sio)
54677368
>>> sio.getvalue()
'Hello, world'
```

```
>>> sio.seek(7)
7
>>> sio.write("there!")
6
>>> id(sio)
54677368
>>> sio.getvalue()
'Hello, there!'
```

4.1.4 可变字符串

```
>>> import array
>>> a = array.array('u', s)   # unicode数据对象
>>> id(a)
55136176
>>> print(a)
array('u', 'Hello, world')
>>> a[0] = 'y'
>>> print(a)
array('u', 'yello, world')
>>> id(a)
55136176

>>> b=a.tounicode()
>>> b
'yello, world'
>>> id(b)
55206768
```

4.2 正则表达式

- 正则表达式是字符串处理的有力工具和技术。
- 正则表达式使用某种预定义的模式去匹配一类具有共同特征的字符串，可以快速、准确地完成复杂的查找、替换等处理要求。
- Python中，re模块提供了正则表达式操作所需要的功能。

4.2.1 正则表达式元字符

元字符	功能说明
.	匹配除换行符以外的任意单个字符
*	匹配位于*之前的字符或子模式的0次或多次出现
+	匹配位于+之前的字符或子模式的1次或多次出现
-	在[]之内用来表示范围
|	匹配位于|之前或之后的字符
^	匹配行首，匹配以^后面的字符开头的字符串
$	匹配行尾，匹配以$之前的字符结束的字符串
?	匹配位于?之前的0个或1个字符。当此字符紧随任何其他限定符（*、+、?、{n}、{n,}、{n,m}）之后时，匹配模式是“非贪心的”。“非贪心的”模式匹配搜索到的、尽可能短的字符串，而默认的“贪心的”模式匹配搜索到的、尽可能长的字符串。例如，在字符串“oooo”中，“o+?”只匹配单个“o”，而“o+”匹配所有“o”
\	表示位于\之后的为转义字符
\num	此处的num是一个正整数。例如，“(.)\1”匹配两个连续的相同字符
\f	换页符匹配
\n	换行符匹配

4.2.1 正则表达式元字符

元字符	功能说明
\r	匹配一个回车符
\b	匹配单词头或单词尾
\B	与\b含义相反
\d	匹配任何数字，相当于[0-9]
\D	与\d含义相反，等效于[^0-9]
\s	匹配任何空白字符，包括空格、制表符、换页符，与[\f\n\r\t\v] 等效
\S	与\s含义相反
\w	匹配任何字母、数字以及下划线，相当于[a-zA-Z0-9_]
\W	与\w含义相反，与“[^A-Za-z0-9_]”等效
()	将位于()内的内容作为一个整体来对待
{}	按{}中的次数进行匹配
[]	匹配位于[]中的任意一个字符
[^xyz]	反向字符集，匹配除x、y、z之外的任何字符
[a-z]	字符范围，匹配指定范围内的任何字符
[^a-z]	反向范围字符，匹配除小写英文字母之外的任何字符

4.2.1 正则表达式元字符

- 最简单的正则表达式是普通字符串，可以匹配自身
- '[pjc]ython'可以匹配'python'、'jython'、'cython'
- '[a-zA-Z0-9]'可以匹配一个任意大小写字母或数字
- '[^abc]'可以匹配任意除'a'、'b'、'c'之外的一个字符
- 'python|perl'或'p(ython|erl)'都可以匹配'python'或'perl'
- 子模式后面加上问号表示可选。
  r'(http://)?(www\.)?python\.org'只能匹配'http://www.python.org'、'http://python.org'、'www.python.org'和'python.org'
- '^http'只能匹配所有以'http'开头的字符串
- (pattern)*：允许模式重复0次或多次
- (pattern)+：允许模式重复1次或多次
- (pattern){m,n}：允许模式重复m~n次

4.2.1 正则表达式元字符

- '(a|b)*c'：匹配多个（包含0个）a或b，后面紧跟一个字母c。
- 'ab{1,}'：等价于'ab+'，匹配以字母a开头后面带1个至多个字母b的字符串。
- '^[a-zA-Z]{1}([a-zA-Z0-9._]){4,19}$'：匹配长度为5-20的字符串，必须以字母开头、可带数字、“_”、“.”的字串。
- '^(\w){6,20}$'：匹配长度为6-20的字符串，可以包含字母、数字、下划线。
- '^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$'：检查给定字符串是否为合法IP地址。
- '^(13[4-9]\d{8})|(15[01289]\d{8})$'：检查给定字符串是否为移动手机号码。
- '^[a-zA-Z]+$'：检查给定字符串是否只包含英文字母大小写。
- '^\w+@(\w+\.)+\w+$'：检查给定字符串是否为合法电子邮件地址。

4.2.1 正则表达式元字符

- '^(\-)?\d+(\.\d{1,2})?$'：检查给定字符串是否为最多带有2位小数的正数或负数。
- '[\u4e00-\u9fa5]'：匹配给定字符串中所有汉字。
- '^\d{18}|\d{15}$'：检查给定字符串是否为合法身份证格式。
- '\d{4}-\d{1,2}-\d{1,2}'：匹配指定格式的日期，例如2016-1-31。
- '^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[,._]).{8,}$'：检查给定字符串是否为强密码，必须同时包含英语字母大写字母、英文小写字母、数字或特殊符号（如英文逗号、英文句号、下划线），并且长度必须至少8位。
- "(?!.*[\'\"\/;=%?]).+"：如果给定字符串中包含’、”、/、;、=、%、?则匹配失败，关于子模式语法请参考表4-4。
- '(.)\\1+'：匹配任意字符的一次或多次重复出现。

4.2.2 re模块主要方法

- compile(pattern[,flags]):创建模式对象
- search(pattern,string[,flags]):在字符串中寻找模式
- match(pattern,string[,flags]):从字符串的开始处匹配模式
- findall(pattern,string[,flags]):列出字符串中模式的所有匹配项
- split(pattern,string[,maxsplit=0]):根据模式匹配项分割字符串
- sub(pat,repl,string[,count=0]):将字符串中所有pat的匹配项用repl替换
- escape(string):将字符串中所有特殊正则表达式字符转义

其中flags的值可以是re.I（忽略大小写）、re.L、re.M（多行匹配模式）、re.S（使元字符.也匹配换行符）、re.U（匹配Unicode字符）、re.X（忽略模式中的空格，并可以使用#注释）的不同组合（使用|进行组合）。

4.2.3 直接使用re模块方法

```
>>> import re
>>> text = 'alpha. beta....gamma delta'
>>> re.split( '[\. ]+' ,text) #注意：方括号中点的后面有一个空格
['alpha', 'beta', 'gamma', 'delta']
>>> re.split('[\. ]+',text,maxsplit=2)#分割2次
['alpha', 'beta', 'gamma delta']
>>> re.split('[\. ]+',text,maxsplit=1)#分割1次
['alpha', 'beta....gamma delta']
>>> pat = '[a-zA-Z]+'
>>> re.findall(pat,text) #查找所有单词
['alpha', 'beta', 'gamma', 'delta']
```

4.2.3 直接使用re模块方法

```
>>> pat = '{name}'
>>> text = 'Dear {name}...'
>>> re.sub(pat,'Mr.Dong',text) #字符串替换
'Dear Mr.Dong...'
>>> s = 'a s d'
>>> re.sub('a|s|d','good',s) #字符串替换
'good good good'
>>> re.escape('http://www.python.org') #字符串转义
'http\\:\\/\\/www\\.python\\.org'
```

4.2.3 直接使用re模块方法

```
>>> print re.match('done|quit','done') #匹配成功
<_sre.SRE_Match object at 0x00B121A8>
>>> print re.match('done|quit','done!') #匹配成功
<_sre.SRE_Match object at 0x00B121A8>
>>> print re.match('done|quit','doe!') #匹配不成功
None
>>> print re.match('done|quit','d!one!') #匹配不成功
None
```

4.2.3 直接使用re模块方法

```
>>> m=re.match(r'www\.(.*)\..{3}','www.python.org')

#matchobject.group()包含了所有匹配的内容，
#等价于matchobject.group(0)，此时的0表示所有的匹配
>>> m.group(0)
'www.python.org '

#matchobject.group(n)，表示返回正则表达式中的第n个组()匹配的内
    容
>>> m.group(1)
'python'

>>> m.start(1)
4
>>> m.end(1)
10
>>> m.span(1)
(4, 10)
```

4.2.3 直接使用re模块方法

- 删除字符串中重复的空格

```
>>> import re
>>> s='aaa      bb      c d e   fff    '
>>> re.split('[\s]+',s)
['aaa', 'bb', 'c', 'd', 'e', 'fff', '']
>>> re.split('[\s]+',s.strip())
['aaa', 'bb', 'c', 'd', 'e', 'fff']
>>> ' '.join(re.split('[\s]+',s.strip()))
'aaa bb c d e fff'
>>> ' '.join(re.split('\s+',s.strip()))
'aaa bb c d e fff'
>>> re.sub('\s+',' ',s.strip())
'aaa bb c d e fff'
>>> s
'aaa
>>> s.split()#也可以不使用正则表达式
['aaa', 'bb', 'c', 'd', 'e', 'fff']
>>> ' '.join(s.split())
'aaa bb c d e fff'
```

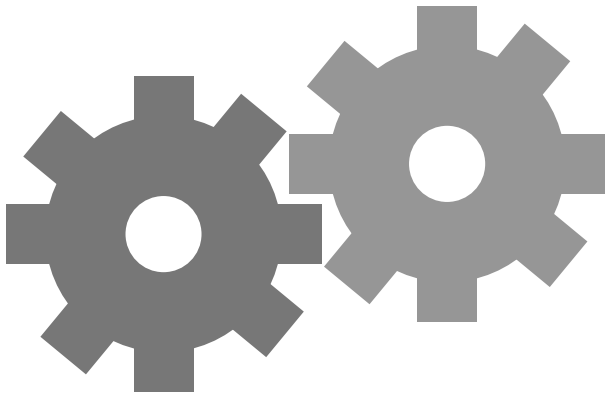
4.2.3 直接使用re模块方法

- 使用以'\'开头的元字符

```
>>> import re
>>> example = 'ShanDong Institute of Business and Technology'
>>> re.findall('\\ba.+?\\b',example)#以a开头的完整单词
['and']
>>> re.findall('\\Bo.+?\\b',example)#含有o字母的单词中第一个非首
    字母o后面的剩余部分
['ong', 'ology']
>>> re.findall('\\b\w.+?\\b',example)#所有单词
['ShanDong', 'Institute', 'of', 'Business', 'and', 'Technology']
>>> re.findall(r'\b\w.+?\b',example)#使用原始字符串，减少需要输
    入的符号数量
['ShanDong', 'Institute', 'of', 'Business', 'and', 'Technology']
>>> re.findall('\d\.\d\.\d','Python 2.7.8')#x.x.x的数字形式
['2.7.8']
>>> re.split('\s',example)#使用任何空白字符分割字符串
['ShanDong', 'Institute', 'of', 'Business', 'and', 'Technology']
```

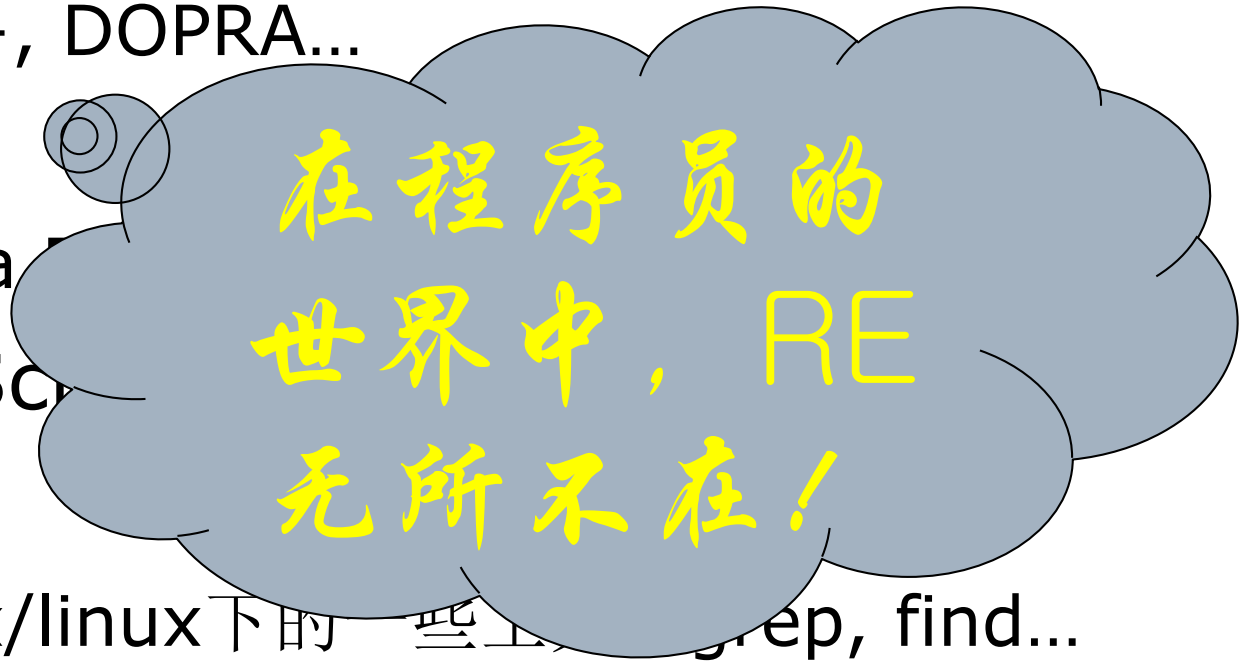
例一：检查email地址是否合法

```
import re
email = raw_input(" input email:")
if re.match("^\w+@(\w+\.)+\w+$", email):
    print email, "is a valid email."
else:
    print email, "is an invalid email."
```



RE的应用

- 开发工具
  - perl, python, php, .Net, Java, Java Script, C++, DOPRA…
- 编辑器
  - Ultra
- XML Sc
- 其他
  - Unix/linux下的 grep, find…
  - DOS命令：dir



python™

谢谢Q/A