# 输出

print("字符串")

print（）

a++不行，a=a+1可以

# 查找

.find 找到第一次匹配的地方，返回该位置前的位置，若未找到返回-1

eg：a="1561310125"

result=a.find("61")

则result的值为2

字符串格式化符

{} .format("aa")

# 屏幕输入

input（）

括号内为对输入内容的解释，会显示出来

# 绝对值

abs（）

# 检查是否有空的值

all（）

>>> all(['a', 'b', 'c', 'd']) #列表list，元素都不为空或0

True

>>> all(['a', 'b', '', 'd']) #列表list，存在一个为空的元素

False

>>> all([0, 1，2, 3]) #列表list，存在一个为0的元素

False

>>> all(('a', 'b', 'c', 'd')) #元组tuple，元素都不为空或0

True

>>> all(('a', 'b', '', 'd')) #元组tuple，存在一个为空的元素

False

>>> all((0, 1，2, 3)) #元组tuple，存在一个为0的元素

False

>>> all([]) # 空列表

True

>>> all(()) # 空元组

True

注意：空元组、空列表返回值为True，这里要特别注意

# 检查是否全为空

any（）

>>> any(['a', 'b', 'c', 'd']) #列表list，元素都不为空或0

True

>>> any(['a', 'b', '', 'd']) #列表list，存在一个为空的元素

True

>>> any([0, '', False]) #列表list,元素全为0,'',false

False

>>> any(('a', 'b', 'c', 'd')) #元组tuple，元素都不为空或0

True

>>> any(('a', 'b', '', 'd')) #元组tuple，存在一个为空的元素

True

>>> any((0, '', False)) #元组tuple，元素全为0,'',false

False

>>> any([]) # 空列表

False

>>> any(()) # 空元组

False

# 判断

bool（）

>>> bool(0)

False

>>> bool("abc")

True

>>> bool("")

False

>>> bool([])

False

>>> bool()

False

>>> issubclass(bool, int) #bool是一个subclass int

True

# 替换

.replace（原句子，新句子）

对象是一个句子

文件写入

file = open(文件路径，"w")

file.write("写入的内容")

file.close

bytearray ????

bytea ????

# 检查函数是否可以被调用

callable()

如果返回true，亦可能不能被调用，但是返回false则一定不能被调用

eg：

>>> callable(add)

False

>>> def add(a,b)

c = a + b

return c

callable(add)

True

# 返回整数i对应的ASCII字符

chr(i)

与ord()作用相反。

参数i：取值范围[0, 255]之间的正数。

版本：该函数在python2和python3各个版本中都可用。不存在兼容性问题。

classmethod() ????

compile() ????

将字符串或数字转化为复数

complex()

>>> complex(1, 2)

(1 + 2j)

#数字

>>> complex(1)

(1 + 0j)

#当做字符串处理

>>> complex("1")

(1 + 0j)

#注意：这个地方在“+”号两边不能有空格，也就是不能写成"1 + 2j"，应该是"1+2j"，否则会报错

>>> complex("1+2j")

(1 + 2j)

# delattr ????

# 结合多个列表

zip

eg：str1=["123","456","789"];

str2=["wang","zi","lin"];

zip(str1,str2)返回值为[("123","wang"),("456","zi"),("789","lin")];

# 求序列长度

len（）

>>>a = "hahaahwpcaonima" print(len(a))

15

>>>a = [123,345,445,"ww"]

>>>print(len(a))

4

# 从Soup中找到指定信息

find\_all（）

！class由于是保留格式，所以要用class\_

# 获取字符串

.get\_text()

！注意text后面括号

# 选择器匹配属于父元素的特定类型的第 N 个子元素的每个元素.

:nth-of-type()

解释：就是选择父元素下的选定元素（例如：:nth-of-type(1)就是选定父元素下的第一个元素，然后按照父元素的排列顺序组成列表）组成列表中的一个子元素

# 找出字符串，并消除空格

.stripped\_strings

# .format 格式化字符串

在Python 3.0中，%操作符通过一个更强的格式化方法format()进行了增强。对str.format()的支持已经被反向移植到了Python 2.6  
  
在2.6中，8-bit字符串和Unicode字符串都有一个format()方法，这个方法会把字符串当作一个模版，通过传入的参数进行格式化。这个用来格式化的模版使用大括号({,})作为特殊字符。  
  
# Substitute positional argument 0 into the string.  
"User ID: {0}".format("root") -> "User ID: root"  
  
# Use the named keyword arguments  
 'User ID: {uid} Last seen: {last\_login}'.format(  
  uid='root',  
  last\_login = '5 Mar 2008 07:20') ->  
 'User ID: root Last seen: 5 Mar 2008 07:20'  
  
大括号可以写两遍来转义。  
  
format("Empty dict: {{}}") -> "Empty dict: {}"  
  
字段名字可以为整数，表示参数的位置。像{0}, {1}等。也可以是参数的名字。你以可以使用字段的组合来读取属性或者字典的key值。  
  
import sys  
'Platform: {0.platform}\nPython version: {0.version}'.format(sys) ->  
 'Platform: darwin\n  
 Python version: 2.6a1+ (trunk:61261M, Mar 5 2008, 20:29:41) \n  
 [GCC 4.0.1 (Apple Computer, Inc. build 5367)]'  
  
import mimetypes  
 'Content-type: {0[.mp4]}'.format(mimetypes.types\_map) ->  
 'Content-type: video/mp4'  
  
注意，当使用字典类型的表示方法，如[.mp4]时，你不需要引号将字符串(.mp4)引起来；它会查找用.mp4作为主键的值。以整数开头的字符串会被转换成一个整数。你不能在被格式化的字符串中写复杂的表达式。  
  
到此，我们已经演示了怎样替换指定的字段。我们还可以通过在格式化指示符后面添加一个冒号来进行精确格式化。例如：  
  
# Field 0: left justify, pad to 15 characters  
# Field 1: right justify, pad to 6 characters  
fmt = '{0:15} ${1:>6}'  
  
fmt.format('Registration', 35) ->  
 'Registration $ 35'  
  
fmt.format('Tutorial', 50) ->  
 'Tutorial $ 50'  
  
fmt.format('Banquet', 125) ->  
 'Banquet $ 125'  
  
格式化指示符可以通过嵌套进行引用。  
  
fmt = '{0:{1}}'  
  
  
width = 15  
fmt.format('Invoice #1234', width) ->  
 'Invoice #1234 '  
    
width = 35  
fmt.format('Invoice #1234', width) ->  
 'Invoice #1234 '  
  
可以指定所需长度的字符串的对齐方式。  
效果字符：  
< （默认）左对齐  
> 右对齐  
^ 中间对齐  
= （只用于数字）在小数点后进行补齐  
  
格式化指示符可以包含一个展示类型来控制格式。例如，浮点数可以被格式化为一般格式或用幂来表示。  
  
>>> '{0:g}'.format(3.75)  
 '3.75'  
>>> '{0:e}'.format(3.75)  
 '3.750000e+00'  
  
展示类型有很多。2.6的文档里有完整的列表。这里列出一些示例。  
  
'b' - 二进制。将数字以2为基数进行输出。  
'c' - 字符。在打印之前将整数转换成对应的Unicode字符串。  
'd' - 十进制整数。将数字以10为基数进行输出。  
'o' - 八进制。将数字以8为基数进行输出。  
'x' - 十六进制。将数字以16为基数进行输出，9以上的位数用小写字母。  
'e' - 幂符号。用科学计数法打印数字。用'e'表示幂。  
'g' - 一般格式。将数值以fixed-point格式输出。当数值特别大的时候，用幂形式打印。  
'n' - 数字。当值为整数时和'd'相同，值为浮点数时和'g'相同。不同的是它会根据区域设置插入数字分隔符。  
'%' - 百分数。将数值乘以100然后以fixed-point('f')格式打印，值后面会有一个百分号。  
  
类和类型可以定义一个\_\_format\_\_()方法来控制怎样格式化自己。它会接受一个格式化指示符作为参数：  
  
def \_\_format\_\_(self, format\_spec):  
 if isinstance(format\_spec, unicode):  
  return unicode(str(self))  
 else:  
  return str(self)  
  
还有一个内置的format()方法可以用来格式化一个值。它会调用类型的\_\_format\_\_()方法，并将格式化指示符作为参数传进去。  
  
>>> format(75.6564, '.2f')  
 '75.66'

# 去除字符串结尾空格及换行

.strip()

str = "wo shi niubi\n "  
print(str.strip())  
print("..")

#wo shi niubi

#..

# 将字符串分割

.split(“”)

a **=** "www.baidu.com/qwertyuiwdfghjk"  
b **=** a.split("/")  
print(b)

#['www.baidu.com', 'qwertyuiwdfghjk']

# 列表去重

temps **=** list(set(temps))