目录

[第二章 2](#_Toc43044443)

[第三章：字符串 4](#_Toc43044444)

[第四章：映射 6](#_Toc43044445)

[第五章：条件、循环及其他语句 8](#_Toc43044446)

[第六章：抽 象（组合程序） 11](#_Toc43044447)

# 第二章

切片**[0:10:1]**

来访问特定范围内的元素。指定起点和终点，还有步长。

运算符**in**

检查特定的值是否包含在序列中

内置函数len、min、max

包含元素个数、最小值、最大值

**函数list**  list('Hello')

使用字符串来创建列表

**del** 语句del names[2]

从列表中删除元素

**Append**

用于将一个对象附加到列表末尾

**.Clear**

就地清空列表的内容

**.Copy** a.copy()

方法 copy 复制列表

**.count**

方法count计算指定的元素在列表中出现了多少次。

**.extend**

方法extend让你能够同时将多个值附加到列表末尾

**.index**

方法index在列表中查找指定值第一次出现的索引

**.insert**  numbers.insert(3, 'four')

方法insert用于将一个对象插入列表。

**.pop**

方法pop从列表中删除一个元素（末尾为后一个元素），并返回这一元素。

**.remove**

方法remove用于删除第一个为指定值的元素。就地修改且不返回值的方法之一。

**.reverse**

方法reverse按相反的顺序排列列表中的元素

**.sort**

方法sort用于对列表就地排序。就地排序意味着对原来的列表进行修改。

x.sort(key=len)

x.sort(reverse=True)

函数**sorted**

获取排序后的列表的副本，

**元组：不可修改的序列**

(1, 2, 3)

函数**tuple**

它将一个序列作为参数，并将其转换为元组

**本章介绍的新函数**

**bytes(string, encoding[, errors])** 对指定的字符串进行编码，并以指定的方式处理错误 **cmath.sqrt(number)** 返回平方根；可用于负数

**float(object)** 将字符串或数字转换为浮点数

**help([object])** 提供交互式帮助

**input(prompt)** 以字符串的方式获取用户输入

**int(object)** 将字符串或数转换为整数

**math.ceil(number)** 以浮点数的方式返回向上圆整的结果

**math.floor(number)** 以浮点数的方式返回向下圆整的结果

**math.sqrt(number)** 返回平方根；不能用于负数

**pow(x, y[, z])** 返回x的y次方对z求模的结果

**print(object, ...)** 将提供的实参打印出来，并用空格分隔

**len(seq)** 返回序列的长度

**list(seq)** 将序列转换为列表

**max(args)** 返回序列或一组参数中的大值

**min(args)** 返回序列和一组参数中的小值

**reversed(seq)** 让你能够反向迭代序列

**sorted(seq)** 返回一个有序列表，其中包含指定序列中的所有元素

**tuple(seq)** 将序列转换为元组

**abs(number)** 返回指定数的绝对值

# 第三章：字符串

**%s**转换说明符

指出了要将值插入什么地方。s意味着将值视为字符串进行格式设置。如果指定的值不是字符串，将使用str将其转换为字符串。其他说明符将导致其他形式的转换。

**format**字符串方法

输入："{name} is approximately {value:.2f}.".format(value=pi, name="π")

输出：'π is approximately 3.14.'

字段名：索引或标识符，指出要设置哪个值的格式并使用结果来替换该字段。

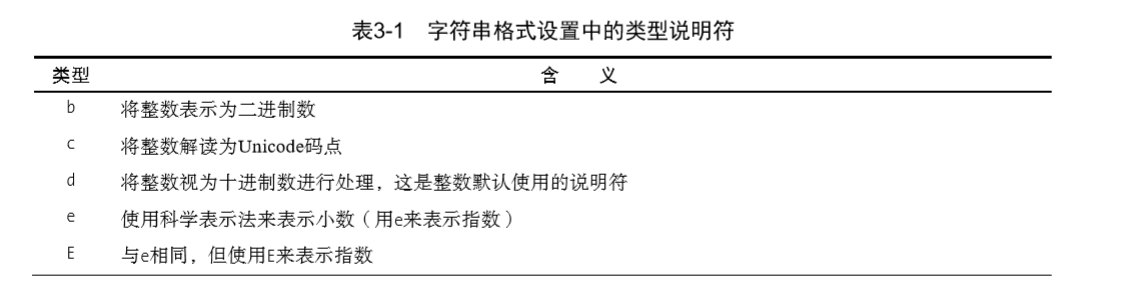
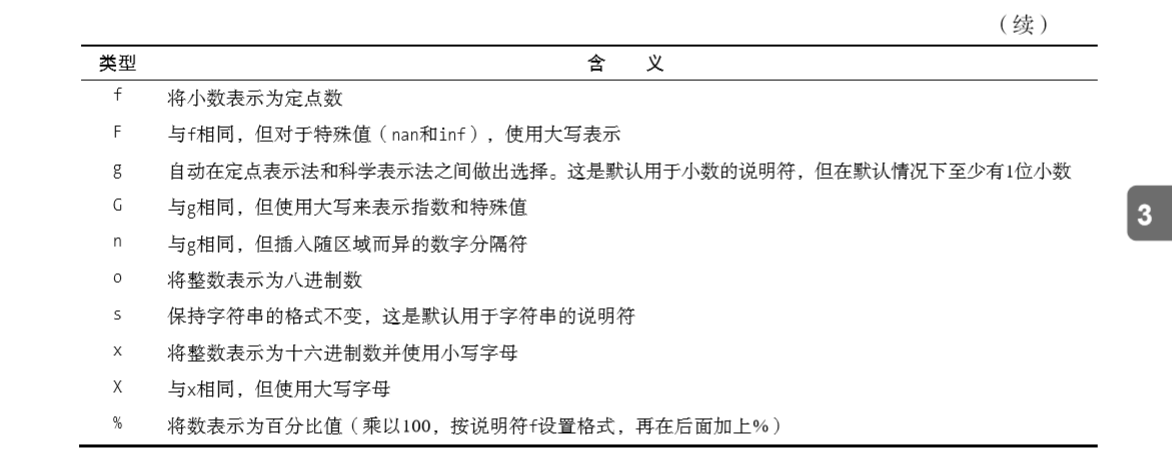
转换标志：跟在叹号后面的单个字符。当前支持的字符包括r(表示repr)、s(表示str)和a(表示ascii)

格式说明符：跟在冒号后面的表达式（这种表达式是使用微型格式指定语言表示的）。

输入：print("{pi!s} {pi!r} {pi!a}".format(pi="π"))

输出：π 'π' '\u03c0'

**格式说明(😊)(😊)(😊)（：）**

要指定左对齐、右对齐和居中，可分别使用<、>和^。

**.center**

方法center通过在两边添加填充字符（默认为空格）让字符串居中。

**.find**

方法find在字符串中查找子串。如果找到，就返回子串的第一个字符的索引，否则返回-1。

subject.find('!!!', 0, 16) # 同时指定了起点和终点(指定的搜索范围包含起点，但不包含终点。)

**附录B：rfind、index、rindex、count、startswith、endswith。**

**join**

join是一个非常重要的字符串方法，用于合并序列的元素(合并序列的元素必须都是字符串)

lower

方法lower返回字符串的小写版本。

**另请参见：islower、istitle、isupper、translate。**

**附录B：capitalize、casefold、swapcase、title、upper。**

**.replace**

方法replace将指定子串都替换为另一个字符串，并返回替换后的结果。

**附录B：expandtabs。**

**.split**

split是一个非常重要的字符串方法，用于将字符串拆分为序列。需要指定分隔符

**附录B：partition、rpartition、rsplit、splitlines。**

**.strip**

方法strip将字符串开头和末尾的空白（但不包括中间的空白）删除，并返回删除后的结果。还可在一个字符串参数中指定要删除哪些字符。(只删除开头或末尾的指定字符)

translate

方法translate替换字符串的特定部分，但不同的是它只能进行单字符替换。

这个方法的优势在于能够同时替换多个字符，因此效率比replace高。

使用translate前必须创建一个**转换表**。

要创建转换表，可对字符串类型str调用方法maketrans

table = str.maketrans('cs', 'kz', ' ') (1)指定字符(2)对应的替换字符 (3)指定删除的字符

# 第四章：映射

创建和使用字典

字典以类似于下面的方式表示：

phonebook = {'Alice': '2341', 'Beth': '9102', 'Cecil': '3258'}

函数 dict

可使用函数dict从其他映射（如其他字典）或键值对序列创建字典。

**len(d)**返回字典d包含的项（键值对）数。 

**d[k]**返回与键k相关联的值。 

**d[k] = v**将值v关联到键k。 

**del d[k]**删除键为k的项。 

**k in d**检查字典d是否包含键为k的项。

**简单的数据库**

people = {

'Alice': { 'phone': '2341', 'addr': 'Foo drive 23' },

'Beth': { 'phone': '9102', 'addr': 'Bar street 42' },

'Cecil': { 'phone': '3158', 'addr': 'Baz avenue 90' }

}

labels = { 'phone': 'phone number', 'addr': 'address'}

name = input('Name: ')

request = input('Phone number (p) or address (a)? ')

if request == 'p': key = 'phone'

if request == 'a': key = 'addr'

if name in people:print("{}'s {} is {}.".format(name, labels[key], people[name][key]))

**format\_map**

指出你将通过一个映射来提供所需的信息。

"Cecil's phone number is {Cecil}.".format\_map(phonebook)

字典方法

**.clear**

方法clear删除所有的字典项，这种操作是就地执行的（就像list.sort一样），因此什么都不 返回（或者说返回None）。

**.copy ()**

方法copy返回一个新字典，其包含的键值对与原来的字典相同（这个方法执行的是浅复制， 因为值本身是原件，而非副本）。

**.fromkeys ()**

方法fromkeys创建一个新字典，其中包含指定的键，且每个键对应的值都是None。

**.get ()**

方法get为访问字典项提供了宽松的环境。通常，如果你试图访问字典中没有的项，将引发 错误。

**items**

方法items返回一个包含所有字典项的列表，其中每个元素都为(key, value)的形式。字典项 在列表中的排列顺序不确定。

**keys**

方法keys返回一个字典视图，其中包含指定字典中的键。

**pop**

方法pop可用于获取与指定键相关联的值，并将该键值对从字典中删除。

**popitem**

方法popitem随机地弹出一个字典项，因为字典项的顺序是不确定的,没有“后一个元素”的概念。(list.popitem)

**setdefault**  d.setdefault('name', 'N/A')

方法setdefault有点像get，因为它也获取与指定键相关联的值，但除此之外，setdefault 还在字典不包含指定的键时，在字典中添加指定的键-值对。

**update**

方法update使用一个字典中的项来更新另一个字典。

**values**

方法values返回一个由字典中的值组成的字典视图。不同于方法keys，方法values返回的视 图可能包含重复的值。

dict(seq)

从键值对、映射或关键字参数创建字典

items = [('name', 'Gumby'), ('age', 42)]

d = dict(items)

# 第五章：条件、循环及其他语句

**print**

你可同时打印多个表达式，条件是用逗号分隔它们

>>> print('Age:', 42)

Age: 42

**同时（并行）给多个变量赋值**

**x, y, z = 1, 2, 3**

**代码块：缩进的乐趣**

在Python中，使用冒号（:）指出 接下来是一个代码块，并将该代码块中的每行代码都缩进相同的程度

**条件和条件语句**

用作布尔表达式（如用作if语句中的条件）时，下面的值都将被解释器视为假： False None 0 "" () [] {}

类型bool布尔

**布尔运算符：and、or、not**

**if条件和条件语句结构**

if name.endswith('Gumby'):

print('Hello, Mr. Gumby')

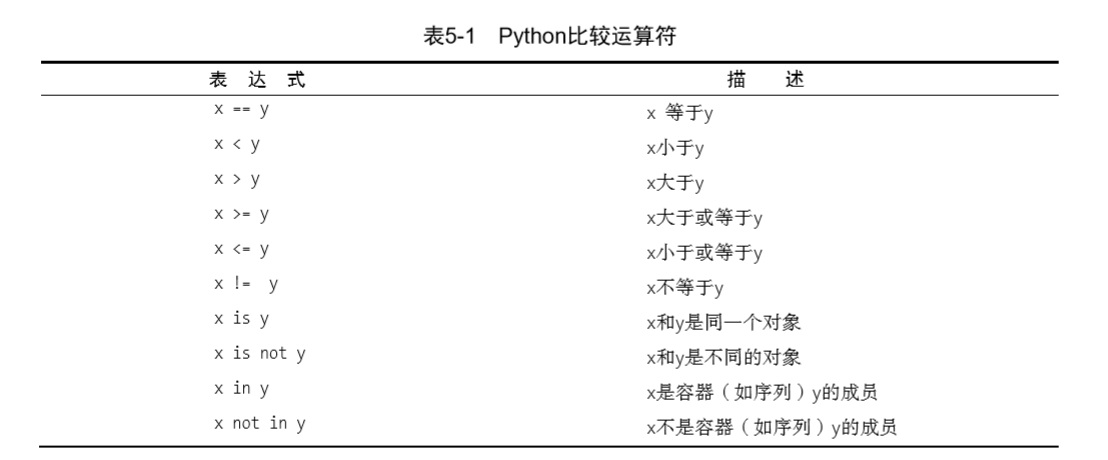
elif语句：

else:

print('Hello, stranger')

三目运算符

status = "friend" if name.endswith("Gumby") else "stranger"



**while 循环**

while语句非常灵活，可用于在条件为真时反复执行代码块。

name = ''

while not name:

name = input('Please enter your name: ')

print('Hello, {}!'.format(name))

**for 循环**

words = ['this', 'is', 'an', 'ex', 'parrot'] for word in words: print(word)

**创建范围的内置函数rang(0,10)**

包含起始位置（这里为0），但不包含结束位置（这里为10）。

***只要能够使用for循环，就不要使用while循环。***

**迭代字典**

d = {'x': 1, 'y': 2, 'z': 3}

for key, value in d.items():

print(key, 'corresponds to', value)

**并行迭代(i是用作循环索引的变量的标准名称。)**

for i in range(len(names)):

print(names[i], 'is', ages[i], 'years old')

并行迭代工具是**内置函数zip**，它将两个序列“缝合”起来，并返回一个由元组组成的序列。

list(zip(names, ages))

[('anne', 12), ('beth', 45), ('george', 32), ('damon', 102)]

**内置函数 enumerate**

strings = ['sed','seds', 'sedss','sedsw','qw']

for index,string in enumerate(strings):

if 'sed' in string:

strings[index]='wa'

**跳出循环**

**break**

要结束（跳出）循环，可使用break。

**continue**

语句continue没有break用得多。它结束当前迭代，并跳到下一次迭代开头。

**while True/break(while true 导致循环永不结束；if 语句条件满足时调用break)**

while True:

name = input('Please enter your name: ')

if not name: break

print('Hello, {}!'.format(name))

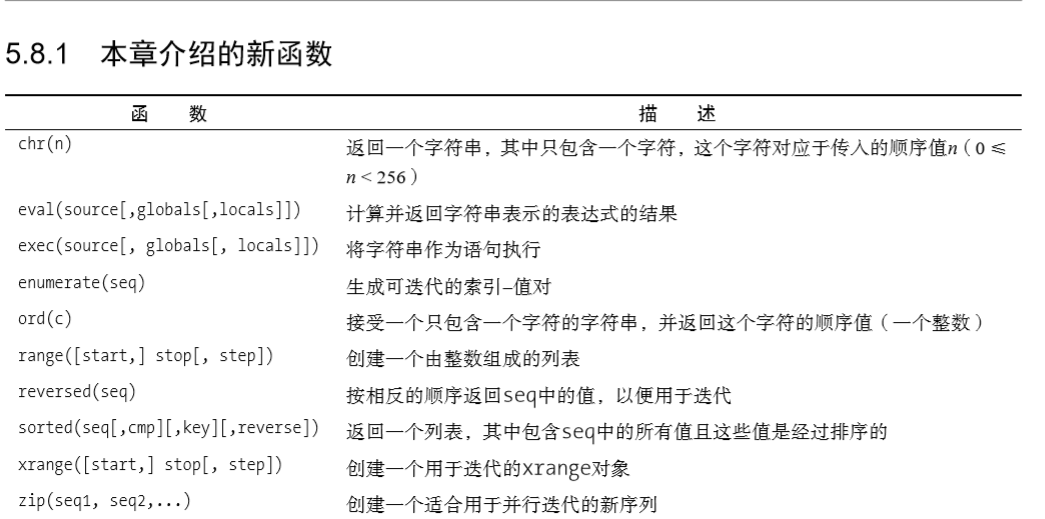
break正常结束循环？如何判断循环正常结束→利用else子句

**pass语句**

那么为何需要一条什么都不做的语句呢？在你编写代码时，可将其用作占位符。例如，你可 能编写了一条if语句并想尝试运行它，但其中缺少一个代码块。

**del语句**

对于你不再使用的对象，Python通常会将其删除



# 第六章：抽 象（组合程序）

**def语句**

自定义函数

**def** hello(name):

**return** 'Hello, ' + name + '!'

**内置函数callable**

判断某个对象是否可调用

**内置函数help**

在交互式解释器中，可使用它获取有关函数的信息，其中包含 函数的文档字符串。

**给函数编写文档**

要给函数编写文档，以确保其他人能够理解，可添加注释（以#打头的内容）。

#初始化字典

def init(data):

data['first'] = {}

data['middle'] = {}

data['last'] = {}

storage = {}

init(storage)

storage

{'middle': {}, 'last': {}, 'first': {}}

#获取人员姓名

def lookup(data, label, name):

return datalabel].get(name)

def init(data):

data['first'] = {}

data['middle'] = {}

data['last'] = {}

def store(data, full\_name):

names = full\_name.split()

if len(names) == 2: names.insert(1, '')

labels = 'first', 'middle', 'last'

for label, name in zip(labels, names):

people = lookup(data, label, name)

if people: people.append(full\_name)

else: data[label][name] = [full\_name]

me = 'Magnus Lie Hetland'

storage['first']['Magnus'] = [me]

storage['middle']['Lie'] = [me]

storage['last']['Hetland'] = [me]