## 第 2 章 变量和简单数据类型

**2.3 字符串**”” ‘’

修改字符串的大小写Title 英 /ˈtaɪtl 首字母大写

修改字符串upper全部大写**英式读音**：/ˈʌpə(r)/

修改字符串全部小写lower()如：car.lower()

**2.3.2 合并（拼接）字符串**

[“a”+”b”]集合

**2.3.3 使用制表符或换行符来添加空白**

两个星号（\*\* ）表示乘方运算

**2.3.3 使用制表符或换行符来添加空白**

空白 泛指任何非打印字符，如空格、制表符和换行符

\n换行符 \t制表符

**2.3.4 删除空白**

删除字符串末尾可使用方法rstrip()

以剔除字符串开头的空白lstrip()，

剔除字符串两端的空白strip()

要永久删除这个字符串中的空白，必须将删除操作的结果存回到变量中

### 2.4 数字

**2.4.1 整数：**在Python中，可对整数执行加（+ ）减（- ）乘（\* ）除（/ ）运算

**2.4.2 浮点数**0.1 + 0.1

**2.4.3 使用函数str() 避免类型错误**

将数值转换成字符串使用函数str() 避免类型错误

**个值为整数（int ）的变量**

**Float小数点**

### 2.5 注释

注释用井号（# ）标识

## 第 3 章 列表简介

用方括号（[] ）来表示列表，并用逗号来分隔其中的元素

bicycles = ['trek', 'cannondale', 'redline', 'specialized']

**3.1.1 访问列表元素**

print(bicycles[0])

**3.1.2 索引从0而不是1开始**

通过将索引指定为-1 ，可让Python返回最后一个列表元素

### 3.2 修改、添加和删除元素

1.添加到了列表末尾 append(‘’) 的读音为：英 /əˈpend/；将元素'a'

2.列表中插入元素 insert()

**3.2.3 从列表中删除元素**

1.使用del 语句删除元素del motorcycles[0]

2.方法pop() 可删除列表末尾的元素，并让你能够接着使用它

如果你不确定该使用del 语句还是pop() 方法，下面是一个简单的判断标准：如果你要从列表中删除一个元素，且不再以任何方式使用它，就使用del 语句；如果你要在删除元

素后还能继续使用它，就使用方法pop() 。4. 根据值删除元素remove()：motorcycles.remove('值')

### 3.3 组织列表

sort /sɔːt/

使用方法sort() 对列表进行永久性排序, 按字母顺序排列

只需向sort() 方法传递参数reverse=True 。“reverse” 的英式读音为：/rɪˈvɜːs/。列表按与字母顺序相反的顺序排列

使用函数sorted() 对列表进行临时按字母顺序排序, 如果你要按与字母顺序相反的顺序显示列表，也可向函数sorted() 传递参数reverse=True

倒着打印列表cars.reverse()

确定列表的长度len(AAA)

## 第 4 章 操作列表

### 4.1 遍历整个列表

使用for 循环

for 新变量 in 老集合数据:

**4.1.2 在for 循环中执行更多的操作**

magicians = ['alice', 'david', 'carolina']

for magician in magicians:

print(magician.title() + ", that was a great trick!")

打印时magician.title()新变量首字母大字

增加换行符”.\n”

**4.1.3 在for 循环结束后执行一些操作**

print("for结事后，加结束语")

### 4.2 避免缩进错误

FOR无缩进，就只会执行FOR最后面的一条数据执行一次结束

为避免意外缩进错误，请只缩进需要缩进的代码。在前面编写的程序中，只有要在for 循环中对每个元素执行的代码需要缩进。

### 4.3 创建数值列表

**4.3.1 使用函数range()**

英式读音：/reɪndʒ/ **一系列，范围；界限**：指一定的变化范围或幅度

For a in range(1,5) 可以给A赋值1到4的数据

**4.3.2 使用range() 创建数字列表**

可使用函数list() 将range() 的结果直接转换为列表

numbers = list(range(1,6)) 得出1, 2, 3, 4, 5

使用函数range() 时，还可指定步长。例如，下面的代码打印1~10内的偶数：even\_numbers = list(range(2,11,2))

在这个示例中，函数range() 从2开始数，然后不断地加2，直到达到或超过终值（11），因此输出如下：[2, 4, 6, 8, 10]

**4.3.3 对数字列表执行简单的统计计算**

几个专门用于处理数字列表的Python函数。例如，你可以轻松地找出数字列表的最大值、最小值和总和：

>>> digits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]

>>> min(digits)

0

>>> max(digits)

9

>>> sum(digits)

45

### 4.4 使用列表的一部分

players = ['charles', 'martina', 'michael', 'florence', 'eli']

print(players[0:3])

### 4.5 元组

dimensions = (200, 50)

无组和数据集不同的是一个是[]一个是(),且元组不可改变

**4.5.2 遍历元组中的所有值**

使用for 循环来遍历元组中的所有值

dimensions = (200, 50)

for dimension in dimensions:

print(dimension)

虽然元组数据不可单独修改，但是整体赋值却是可以

如上面的修改

dimensions = (300, 60)

for dimension in dimensions:

print(dimension)

### 4.6 设置代码格式

4.6.2 **缩进**

PEP 8建议每级缩进都使用四个空格，这既可提高可读性，又留下了足够的多级缩进空间。

**行长：**很多Python程序员都建议每行不超过80字符

**空行：**要将程序的不同部分分开，可使用空行。不要在程序文件中过多地使用空行

## 第 5 章 if 语句

5.2.1 检查是否相等检查是否相等（==）

5.2.2 检查是否相等时不考虑大小写

5.2.3 检查是否不相等可结合使用惊叹号和等号（!= ）

5.2.4 比较数字age == 18

条件语句中可包含各种数学比较，如小于、小于等于、大于、大于等于：

age = 19 >>>

age < 21 True

age <= 21 True

age > 21 False

age >= 21 False

**5.2.5 检查多个条件**

**1. 使用and 检查多个条件**

❶ >>> age\_0 = 22 >>> age\_1 = 18

❷ >>> age\_0 >= 21 and age\_1 >= 21 False

❸ >>> age\_1 = 22 >>> age\_0 >= 21 and age\_1 >= 21 True

**2. 使用or 检查多个条件**

❶ >>> age\_0 = 22 >>> age\_1 = 18

❷ >>> age\_0 >= 21 or age\_1 >= 21 True

❶ >>> age\_0 = 18 >>> age\_0 >= 21 or age\_1 >= 21 False

**5.2.6 检查特定值是否包含在列表中：in**

要判断特定的值是否已包含在列表中，可使用关键字in

>> requested\_toppings = ['mushrooms', 'onions', 'pineapple']

❶ >>> 'mushrooms' in requested\_toppings True

❷ >>> 'pepperoni' in requested\_toppings False

**5.2.7 检查特定值是否不包含在列表中： not in**

banned\_users = ['andrew', 'carolina', 'david'] user = 'marie'

❶ if user not in banned\_users: print(user.title() + ", you can post a response if you wish.")

**5.2.8 布尔表达式**

与条件表达式一样，布尔表达式的结果要么为True ，要么为False 。

game\_active = True

can\_edit = False

### 5.3 if 语句

在if 语句中，缩进的作用与for 循环中相同。如果测试通过了，将执行if 语句后面所有缩进的代码行，否则将忽略它们。

age = 19

if age >= 18:

print("You are old enough to vote!")

print("Have you registered to vote yet?")

**5.3.2 if-else 语句**

age = 17

❶ if age >= 18:

print("You are old enough to vote!")

print("Have you registered to vote yet?")

❷ else:

print("Sorry, you are too young to vote.")

print("Please register to vote as soon as you turn 18!")

**5.3.3 if-elif-else 结构**

age = 12

if age < 4:

price = 0

elif age < 18:

price = 5

elif age < 65:

price = 10

else:

price = 5

print("Your admission cost is $" + str(price) + ".")

**5.3.4 使用多个elif 代码块**

**5.3.5 省略else 代码块**

Python并不要求if-elif 结构后面必须有else 代码块。在有些情况下，else 代码块很有用；而在其他一些情况下，使用一条elif 语句来处理特定的情形更清晰：else 是一条包罗万象的语句，只要不满足任何if 或elif 中的条件测试，其中的代码就会执行，这可能会引入无效甚至恶意的数据。

age = 12

if age < 4:

price = 0

elif age < 18:

price = 5

elif age < 65:

price = 10

elif age >= 65:

price = 5 print("Your admission cost is $" + str(price) + ".")

**5.3.6 测试多个条件**

有多个条件为True ，且你需要在每个条件为True 时都采取相应措施时，适合使用这种方法。

❶ requested\_toppings = ['mushrooms', 'extra cheese']

❷ if 'mushrooms' in requested\_toppings:

print("Adding mushrooms.")

❸ if 'pepperoni' in requested\_toppings:

print("Adding pepperoni.")

❹ if 'extra cheese' in requested\_toppings:

print("Adding extra cheese.")

print("\nFinished making your pizza!")

### 5.4 使用if 语句处理列表

print("\n如果比萨店的青椒用完了，该如何处理呢？")

bss=["青椒","香菜","葱花"]

for bs in bss:

if bs=="青椒":

print("青椒用完了")

else:

print("目前还有"+bs)

**5.4.2 确定列表不是空的**

requested\_toppings = []

if requested\_toppings:

for requested\_topping in requested\_toppings:

print("Adding " + requested\_topping + ".")

print("\nFinished making your pizza!")

else:

print("Are you sure you want a plain pizza?")

**5.4.3 使用多个列表**

**available\_toppings = ['mushrooms', 'olives', 'green peppers','pepperoni', 'pineapple', 'extra cheese']**

**requested\_toppings = ['mushrooms', 'french fries', 'extra cheese']**

**for requested\_topping in requested\_toppings:**

**if requested\_topping in available\_toppings:**

**print("Adding " + requested\_topping + ".")**

**else:**

**print("Sorry, we don't have " + requested\_topping + ".")**

**print("\nFinished making your pizza!")**

## 第 6 章 字典

### 6.1 一个简单的字典

alien\_0 = {'color': 'green', 'points': 5}

print(alien\_0['color']+" "+str(alien\_0['points']))

结果: green 5

### 6.2 使用字典

字典用放在花括号{} 中的一系列键—值对表示，如前面的示例所示：

alien\_0 = {'color': 'green'}

**6.2.1 访问字典中的值**

alien\_0 = {'color': 'green'}

print(alien\_0['color'])**6.2.2 添加键—值对**

alien\_0 = {'color': 'green', 'points': 5}

print(alien\_0)

alien\_0['x\_position'] = 0

alien\_0['y\_position'] = 25

print(alien\_0)

**6.2.3 先创建一个空字典**

alien\_0 = {}

alien\_0['color'] = 'green'

alien\_0['points'] = 5

print(alien\_0)

**6.2.4 修改字典中的值**

alien\_0 = {'color': 'green'}

print("The alien is " + alien\_0['color'] + ".")

alien\_0['color'] = 'yellow'

print("The alien is now " + alien\_0['color'] + ".")

外星人从绿色改为黄色

**6.2.5 删除键—值对**

alien\_0 = {'color': 'green', 'points': 5}

print(alien\_0)

del alien\_0['points']

print(alien\_0)

### 6.3 遍历字典

.items()如user.items():

遍历显示所有的胎圈的列子：

bead={'K712B722':'138456  478  35A',

'3817B730':'1386HC 484 55L',

'K415B7MO':'1384HC 484 35A防爆胎',

'K421B823':'1464HC 484 30A防爆胎',

print("\n显示库存里的所有胎圈")

for sd,tq in bead.items():

print("成代:"+sd+"\t胎圈:"+tq)

print("所有库存显示完毕")

**6.3.2 遍历字典中的所有键**

在不需要使用字典中的值时，方法keys() 很有用

显示所有成代

print("只显示所有的成代")

for sd in bead.keys():

print(sd)

如果成代是K712B722和3817B730才显示胎圏：

print("\n如果成代是K712B722和3817B730则显示胎圏")

xianshi=['K712B722','3817B730']

for sd in bead.keys():

if sd in xianshi:

print(sd+"\t"+bead[sd])

print("\n根据成代查找如果没有此成代则显示无此成代")

k='k712b722'

if k not in bead.keys():

print(k+"没有此成代")

**6.3.3 按顺序遍历字典中的所有键**sorted()

* 可使用函数sorted() **英式读音**：/ˈsɔːtɪd/来获得按特定顺序排列的键列表的副本：

print("\n可使用函数sorted() 来获得按特定顺序排列的键列表的副本：")

abc={"b":2,"c":3,"a":1}

for nm in sorted(abc.keys()):

print(nm)

结果显示：

a

b

c

**6.3.4 遍历字典中的所有值**values() /ˈvæljuːz/

print("遍历字典中的所有值")

for zhi in abc.values():

print(zhi)

结果显示：

2

3

1

**为剔除重复项，可使用集 合（set）。集合 类似于列表，但每个元素都必须是独一无二的：set**

print("遍历字典剔除重复项值")

abc={"b":"bb","c":"cc","a":"aa","d":"bb"}

for cf in set(abc.values()):

print(cf)

显示结果：

aa

bb

cc

### 6.4 嵌套

**6.4.1 字典列表**

print("\n列表")

alien\_0 = {'color': 'green', 'points': 5}

alien\_1 = {'color': 'yellow', 'points': 10}

alien\_2 = {'color': 'red', 'points': 15}

aliens = [alien\_0, alien\_1, alien\_2]

for alien in aliens:

print(alien)

结果显示：

列表

{'color': 'green', 'points': 5}

{'color': 'yellow', 'points': 10}

{'color': 'red', 'points': 15}

使用range() 生成了20个外星人

print("\n使用代码自动生成20个机器人")

aa=[]

for a in range(20):

new\_a={'color': 'green', 'points': 5, 'speed': 'slow'}

aa.append(new\_a)

for a in aa[:5]:

print(a)

print("...")

print("创建了"+str(len(aa))+"机器人")

print("\n批量修改上面的20个机器人将前三个外星人修改为黄色的、速度为中等且值10个点")

for a in aa[0:3]:

if a['颜色']=='绿色':

a['颜色']='黄色'

a['值']=10

a['速度']='中等'

for a in aa[0:5]:

print(a)

print("...")

print("\n在上面的基石上继续修改将黄色外星人改为移动速度快且值15个点的红色外星人")

for a in aa[0:6]:

if a['颜色']=='绿色':

a['颜色']='黄色'

a['值']=10

a['速度']='中等'

elif a['颜色']=='黄色':

a['颜色']='红色'

a['值']=15

a['速度']='快速'

for a in aa[0:10]:

print(a)

print("...")

**6.4.2 在字典中存储列表**

print("\n6.4.2 在字典中存储列表,一键存多个值")

ps = {'crust':'thick',

'toppings':['mushrooms','extra cheese'],

}

print(ps['crust'])

for p in ps['toppings']:

print(p)

结果：

thick

mushrooms

extra cheese

print("\n显示变量的所有值")

favorite\_languages = {

'jen': ['python', 'ruby'],

'sarah': ['c'],

'edward': ['ruby', 'go'],

'phil': ['python', 'haskell'],

}

for aa,bb in favorite\_languages.items():

print("\n"+aa+"所有的值是")

for b in bb:

print(b)

print("...")

结果：

显示变量的所有值

jen所有的值是

python

ruby

sarah所有的值是

c

edward所有的值是

ruby

go

phil所有的值是

python

haskell

...

**6.4.3 在字典中存储字典**

**print("\n字典中存储字典")**

**print("------------------")**

**yhs = {**

**'aeinstein': {**

**'first': 'albert',**

**'last': 'einstein',**

**'location': 'princeton',**

**},**

**'mcurie': {**

**'first': 'marie',**

**'last': 'curie',**

**'location': 'paris',**

**},**

**}**

**for yh,mz in yhs.items():**

**print("\n"+yh.title())**

**qm=mz['first']+" "+mz['last']**

**dz=mz['location']**

**print("全名是："+qm.title())**

**print("地址是："+dz.title())**

结果：

字典中存储字典

------------------

Aeinstein

全名是：Albert Einstein

地址是：Princeton

Mcurie

全名是：Marie Curie

地址是：Paris

## 第 7 章 用户输入和while 循环

### 7.1 函数input() 的工作原理

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

print("Hello, " + name + "!")

有时候，提示可能超过一行

第1行将消息的前半部分存储在变量prompt 中；在第2行中，运算符+= 在存储在prompt 中的字符串末尾附加一个字符串。

prompt = "If you tell us who you are, we can personalize the messages you see." prompt += "\nWhat is your first name? "

name = input(prompt)

print("\nHello, " + name.title() + "!")

输入guyue后结果显示：

If you tell us who you are, we can personalize the messages you see.

What is your first name? gu yue

Hello, Gu Yue!

**7.1.2 使用int() 来获取数值输入**

**a=input("请输入你的年龄")**

**a=int(a)**

**print(a+1)**

显示：

请输入你的年龄21

22

通过输入的数据做出判断

a=input("请输入你的身高看是否满足坐过山车的身高要求：")

a=int(a)

if a>=18:

print("你符合要求")

else:

print("你不符合")显示结果：

请输入你的身高看是否满足坐过山车的身高要求：18

你符合要求

**7.1.3 求模运算符**

（%）它将两个数相除并返回余数：

如果一个数可被另一个数整除，余数就为0，因此求模运算符将返回0。你可利用这一点来判断一个数是奇数还是偶数：

a=input("判断奇数还是偶数：")

a=int(a)

if a % 2 == 0:

print("这个数是整数")

else:

print("这个是奇数")

结果显示：

判断奇数还是偶数：4

这个数是整数

**7.1.4 在Python 2.7中获取输入**

如果你使用的是Python 2.7，应使用函数raw\_input() 来提示用户输入。这个函数与Python 3中的input() 一样，也将输入解读为字符串。

### 7.2 while 循环简介

英式发音： /waɪl/

for 循环用于针对集合中的每个元素都一个代码块，而while 循环不断地运行，直到指定的条件不满足为止。

a=1

while a <=3:

print(a)

a += 1

结果显示：

1

2

3

**7.2.2 让用户选择何时退出**

sy = "请输入正确密码"

sy +="\n输入后ENTER确认:"

mm = 0

while mm != 1234:

mm = int(input(sy))

print(mm)

请输入正确密码

输入后ENTER确认:1234

1234

这个程序很好，唯一美中不足的是，它将1234密码打印了出来，解决办法加入IF

sy = "请输入正确密码"

sy +="\n输入后ENTER确认:"

mm = 0

while mm != 1234:

mm = int(input(sy))

if mm != 1234:

print(str(mm)+" 密码错误！")

print("\n")

else:

print("密码正确")

请输入正确密码

输入后ENTER确认:1111

1111 密码错误！

请输入正确密码

输入后ENTER确认:1234

密码正确

**7.2.3 使用标志**

在要求很多条件都满足才继续运行的程序中，可定义一个变量，用于判断整个程序是否处于活动状态。这个变量被称为标志 ，充当了程序的交通信号灯，让他在True 时继续运行，False 时停止运行。注意首字母需要是大写

prompt = "告诉我一些事，我会复述给你听:"

prompt += "\n输入quit退出:"

active = True

while active:

message = input(prompt)

if message == 'quit':

active = False

else:

print(message)

print("\n")

告诉我一些事，我会复述给你听:

输入quit退出:quit

**7.2.4 使用break 退出循环**

**英音 /breɪk/，中断，中止**

注意 在任何Python循环中都可使用break 语句。例如，可使用break 语句来退出遍历列表或字典的for 循环。

sr = '请输入一个你去过的城市的名字'

sr += '\n输入q结束:'

while True:

city = input(sr)

if city == 'q':

break

else:

print("我很想去"+city)

print("\n")

请输入一个你去过的城市的名字

输入q结束:q

**7.2.5 在循环中使用continue**

英音 /kənˈtɪnjuː/（使）继续；（使）延续

print("但只打印其中奇数的循环")

oushu = 0

while oushu < 6:

oushu += 1

if oushu % 2 ==0:

continue

print(oushu)

结果显示：

但只打印其中奇数的循环

1

3

5

**7.2.6 避免无限循环**

x = 1

while x <= 5:

print(x)

如这个

### 7.3 使用while 循环来处理列表和字典

print("未验证的用记添加进入已验证用户里去")

wyzyh = ["a","b","c"]

yyzyh = ["aa","bb","cc"]

while wyzyh:

yyz = wyzyh.pop()

print("已验证并重未验证中删除的数据"+yyz)

yyzyh.append(yyz)

print("全部验证过的数据")

for yh in yyzyh:

print(yh)

显示结果：

未验证的用记添加进入已验证用户里去

已验证并重未验证中删除的数据c

已验证并重未验证中删除的数据b

已验证并重未验证中删除的数据a

全部验证过的数据

aa

bb

cc

c

b

a

**7.3.2 删除包含特定值的所有列表元素**

使用函数remove() 来删除列表中的特定值

print("删除多个值为'cat' 的元素")

ps = ['dog', 'cat', 'dog', 'goldfish', 'cat', 'rabbit', 'cat']

print(ps)

while 'cat' in ps:

ps.remove('cat')

print(ps)

结果显示

删除多个值为'cat' 的元素

['dog', 'cat', 'dog', 'goldfish', 'cat', 'rabbit', 'cat']

['dog', 'dog', 'goldfish', 'rabbit']

**7.3.3 使用用户输入来填充字典**

print("填写调查问卷，分别录入名字和爱好")

wenj = {}

zhix = True

while zhix:

mingz = input("请输入你的名字：")

aih = input("请输入你的爱好: ")

wenj[mingz] = aih

jix = input("继续请输入Y，退出请输入N: ")

print(" ")

if jix.upper() == "N" :

zhix = False

print("打印本次调查问卷的所有资料")

for m,a in wenj.items():

print(m+"的爱好是"+a+".")结果：

填写调查问卷，分别录入名字和爱好

请输入你的名字：胡

请输入你的爱好: 吃

继续请输入Y，退出请输入N: Y

请输入你的名字：李

请输入你的爱好: 玩

继续请输入Y，退出请输入N: N

打印本次调查问卷的所有资料

胡的爱好是吃.

李的爱好是玩.

## 第 8 章 函数

### 8.1 定义函数def

如：def greet\_user():

def greet\_user():

"""显示简单的问候语"""

print("Hello!")

greet\_user()

结果

Hello!

**8.1.1 向函数传递信息**

def a(mingz):

"""简单的问候"""

print("你好："+mingz+"!")

a('胡')

结果

你好：胡!

**8.1.2 实参和形参**

def a(mingz):定义的mingz是一个形参，a('胡')这个胡就是实参，将胡传递给了函数a() ，这个 值被存储在形参mingz 中。

### 8.2 传递实参

**8.2.1 位置实参**

鉴于函数定义中可能包含多个形参，因此函数调用中也可能包含多个实参。向函数传递实参的方式很多，可使用位置实参 ，这要求实参的顺序与形参的顺序相同；也可使用关键 字实参 ，其中每个实参都由变量名和值组成；还可使用列表和字典。下面来依次介绍这些方式。

print("定一个什么类型的宠物他的名字是什么")

def congw(mao,laos):

"""显示宠物信息"""

print("猫叫" + mao+" "+"老鼠叫" + laos)

congw('汤姆','杰瑞')

""" **1. 调用函数多次**"""

congw('TOM','Jerry')

结果：

定一个什么类型的宠物他的名字是什么

猫叫汤姆 老鼠叫杰瑞

猫叫TOM 老鼠叫Jerry

**2. 位置实参的顺序很重要**

**8.2.2 关键字实参**

print("定一个什么类型的宠物他的名字是什么")

def congw(mao,laos):

"""显示宠物信息"""

print("猫叫" + mao+" "+"老鼠叫" + laos)

"""**以下就是关键字实参**"""

congw(mao='汤姆',laos='杰瑞')

**8.2.3 默认值**

print("设置函数的默认值")

def a(mao,gou='杰瑞'):

print(mao,gou)

a(mao='汤姆')

"""或者a('汤姆')"""

结果

设置函数的默认值

汤姆 杰瑞

**8.2.4 等效的函数调用**

# 一条名为Willie的小狗

describe\_pet('willie')

describe\_pet(pet\_name='willie')

# 一只名为Harry的仓鼠

describe\_pet('harry', 'hamster')

describe\_pet(pet\_name='harry', animal\_type='hamster')

describe\_pet(animal\_type='hamster', pet\_name='harry') **8.2.5 避免实参错误**

### 8.3 返回值return

英音 /rɪˈtɜːn/ 返回

**8.3.1 返回简单值**

print("设置函数的默认值")

def a(mao,gou='杰瑞'):

print(mao,gou)

a(mao='汤姆')

"""a('汤姆')"""

print("返回简单值将a和b合并在一起")

def abc(a,b):

hb = a + b

return hb.title()

"""返回两个值的合并信息"""

hbp = abc('aaa','bbb')

print(hbp)

结果

返回简单值将a和b合并在一起

Aaabbb

**8.3.2 让实参变成可选的**

指定一个默认值——空字符串

print("少数人的名字有三个词组，就是说中间也有名字的如何正确的显示表达运用空字符串")

def mingz(a,c,b=''):

if c:

quanm = a+b+c

else:

quanm =a+c

return quanm.title()

dingymz = mingz('aaa','ccc','bbb')

print(dingymz)

dingymz = mingz('aaa','bbb')

print(dingymz)

Aaabbbccc

Aaabbb

**8.3.3 返回字典**

函数可返回任何类型的值，包括列表和字典等较复杂的数据结构

print("\n返回字典")

def aa(a,b):

bb = {'你':a,'我':b}

return bb

"""返回的时候把你，和我分别返回进了字典"""

c = "好"

cc = aa(c,c)

print(cc)

结果

返回字典

{'你': '好', '我': '好'}

def build\_person(first\_name, last\_name, age=''):

"""返回一个字典，其中包含有关一个人的信息"""

person = {'first': first\_name, 'last': last\_name}

if age:

person['age'] = age

return person

musician = build\_person('jimi', 'hendrix', age=27)

print(musician)

结果

{'first': 'jimi', 'last': 'hendrix', 'age': 27}

**8.3.4 结合使用函数和while 循环**

print("\n使用姓名跟用户打招呼")

def mingz(xing,ming):

quanm = xing+ming

return quanm

while True:

print("请告诉我你的名字")

print("输入Q退出")

x = input("xing:")

if x.upper() == 'Q':

break

m =input("ming:")

if m.upper == 'Q':

break

jiesqm = mingz(x,m)

print("你好:"+ jiesqm)

结果

使用姓名跟用户打招呼

请告诉我你的名字

输入Q退出

xing:胡

ming:涛

你好:胡涛

请告诉我你的名字

输入Q退出

xing:q

### 8.4 传递列表

print("向所有用户发一个消息")

def fasong(yongh):

for yh in yongh:

xiaox = "你好：" + yh

print(xiaox)

yonghu = ["李先生","张先生","胡先生"]

fasong(yonghu)

结果

向所有用户发一个消息

你好：李先生

你好：张先生

你好：胡先生

**8.4.1 在函数中修改列表**

print("打印的设计存储在一个列表中，打印后移到另一个列表中")

daidayin = ['a','b','c']

dayinhou = []

while daidayin:

shanchu = daidayin.pop()

print("已打印并转移的数据:" + shanchu)

dayinhou.append(shanchu)

print("显示所有打印好的数据")

for xs in dayinhou:

print(xs)

结果

打印的设计存储在一个列表中，打印后移到另一个列表中

已打印并转移的数据:c

已打印并转移的数据:b

已打印并转移的数据:a

显示所有打印好的数据

c

b

a

print("\n用函数重新设计以上程序")

def shuju(daidy,yidy):

"""存放待打印和已经打印的数据"""

while daidy:

shanchu = daidy.pop()

print(shanchu+"已打印")

yidy.append(shanchu)

def dayinsj(yidy):

"""已经打印的数据"""

print("显示所有已经打印的数")

for dysy in yidy:

print(dysy)

"""给待打印数据"""

daidy = ['a','b','c']

yidy = []

shuju(daidy,yidy)

dayinsj(yidy)

结果

用函数重新设计以上程序

c已打印

b已打印

a已打印

显示所有已经打印的数

c

b

a

**8.4.2 禁止函数修改列表**

可向函数传递列表的副本而不是原件；这样函数所做的任何修改都只影响副本，而丝毫不影响原件。

要将列表的副本传递给函数，可以像下面这样做：

function\_name(list\_name[:])

切片表示法[:] 创建列表的副本。

print\_models(unprinted\_designs[:], completed\_models)

### 8.5 传递任意数量的实参

下面的函数只有一个形参 **\*** ，但不管调用语句提供了多少实参，这个

形参都将它们统统收入囊中

def make\_pizza(\*toppings):

"""打印顾客点的所有配料"""

print(toppings)

make\_pizza('pepperoni')

make\_pizza('mushrooms', 'green peppers', 'extra cheese')

结果

('pepperoni',)

('mushrooms', 'green peppers', 'extra cheese')

print("顾客点菜系统")

print("------------------")

def caidan(\*caim):

for cm in caim:

print("顾客点了"+cm)

caidan('土豆丝')

caidan('土豆丝',"回锅肉","鱼","鸡")

结果

顾客点菜系统

------------------

顾客点了土豆丝

顾客点了土豆丝

顾客点了回锅肉

顾客点了鱼

顾客点了鸡

**8.5.1 结合使用位置实参和任意数量实参**

如果要让函数接受不同类型的实参，必须在函数定义中将接纳任意数量实参的形参放在最后

print("顾客点菜系统")

print("------------------")

def caidan(guige,\*caim):

print("顾客点的是："+guige+"的：")

for cm in caim:

print(cm)

caidan('大份','土豆丝')

caidan('小份','土豆丝',"回锅肉","鱼","鸡")

结果

顾客点菜系统

------------------

顾客点的是：大份的：

土豆丝

顾客点的是：小份的：

土豆丝

回锅肉

鱼

鸡

**8.5.2 使用任意数量的关键字实参**\*\*

def abcd(a,b,\*\*cd):

kong = {}

kong['aaa'] = a

kong['bbb'] = b

for aa,bb in cd.items():

kong[aa] = bb

return kong

yongh = abcd('albert', 'einstein',

location='princeton',

field='physics')

print(yongh)

结果

{'aaa': 'albert', 'bbb': 'einstein', 'location': 'princeton', 'field': 'physics'}

### 8.6 将函数存储在模块中

函数的优点之一是，使用它们可将代码块与主程序分离。import 语句允许在当前运行的程序文件中使用模块中的代码。

**8.6.1 导入整个模块**

"""这是一个打印蛋糕尺寸和口味的模块文档名字为dangao.py"""

def dangaody(chicun,\*beizhu):

print("蛋糕的尺寸是："+str(chicun)+",客户要求备注如下：")

for bz in beizhu:

print("- "+bz)

"""另外的文件调用蛋糕下单模块import dangao"""

import dangao

dangao.dangaody(19,'植物奶油')

dangao.dangaody(17,'蜡烛','餐具','植物奶油')

结果

蛋糕的尺寸是：19,客户要求备注如下：

- 植物奶油

蛋糕的尺寸是：17,客户要求备注如下：

- 蜡烛

- 餐具

- 植物奶油

**8.6.2 导入特定的函数from import**

通过用逗号分隔函数名，可根据需要从模块中导入任意数量的函数：

from 文档文 import 函数1, 函数2, 函数3

from dangao import dangaody

"""调用dangao文件的的dangaody函数"""

dangaody(17,'草莓')

结果

蛋糕的尺寸是：17,客户要求备注如下：

- 草莓

**8.6.3 使用as 给函数指定别名**

from dangao import dangaody as dgdy

"""将dangao文件的dangaody函数在此文档改名为dgdy"""

dgdy(17,'草莓')

结果

蛋糕的尺寸是：17,客户要求备注如下：

- 草莓

**8.6.4 使用as 给模块指定别名**

import dangao as dg

"""将dangao模块改名为dg"""

dg.dangaody(17,'草莓')

结果如上

**8.6.5 导入模块中的所有函数**

使用星号（\* ）运算符可让Python导入模块中的所有函数：

from dangao import \*

### 8.7 函数编写指南

编写函数时，需要牢记几个细节。应给函数指定描述性名称，且只在其中使用小写字母和下划线。描述性名称可帮助你和别人明白代码想要做什么。给模块命名时也应遵循上述

约定。

给形参指定默认值时，等号两边不要有空格：

def function\_name(parameter\_0, parameter\_1='default value')对于函数调用中的关键字实参，也应遵循这种约定：

function\_name(value\_0, parameter\_1='value')

## 第 9 章 类

### 9.1 创建和使用类class

9.1.1 创建类

class cat():#创建了一个空类

pass#防止报错

class dog:#简便的空类