第5章 函数设计初步

5.1 函数定义

5.1.1 程序设计函数的起源

数学中的函数 y=f(x)可以实现某种数据运算功能，例如 y=sin(x)用来计算自变量 x 的正弦值 y。程序设计中也有函数的概念。程序中的函数是可以实现某个特定功能的小程序块(block)。每当程序需要实现该特定功能时，只需调用事先写好的函数，不必每次重复编写相同功能的代码。当需要改变函数功能时，只需要修改函数中的代码，则程序中所有调用该函数的地方都会同步修改。因此使用函数可以实现代码复用，提高编程效率。

使用函数可以简化程序结构。复杂的编程问题可以分解成若干简单的子问题，每个子问题用函数来解决。函数像积木块一样通过灵活组合构建起复杂程序。这种简化问题的方法体现了“自顶向下、模块化编程”的程序设计思想。

使用函数可以使程序更容易阅读。函数使程序结构层次分明，好的函数名可以直接体现程序的功能，易于阅读和理解。

使用函数还可以让程序更容易调试和测试。如果程序由一系列的函数组成，程序员可以编写程序调用每个函数，在各种测试条件下测试函数的运行状态。当所有函数都能正常运行时，程序的运行就基本没有问题。当程序运行出错时，也很容易定位到某个函数并找出错误。

5.1.2 函数的定义

Python 包含内建函数和用户自定义函数。内建函数是 Python 事先定义好的程序，用户可以直接使用，如 pow()、input()、print()等。用户也可以自定义函数，方法如下：

```
def 函数名 (参数 1,参数 2…):
    函数体
    return 返回值
```

其中，第一行是 def 语句，用关键字 def(def 是英文 define 的缩写)定义函数。def 与

函数名之间有一个空格。函数名通常体现函数功能，且必须符合变量命名的规则。函数名后面圆括号中的参数 1、参数 2 等称为“形式参数”，简称为形参。它的作用是实现调用程序与被调用函数之间的联系。通常把函数要处理的数据、影响函数功能的参数或者函数的运行结果作为形参。形参的个数可以是零个、一个或多个，当参数个数为零时，圆括号也要保留。def 语句以冒号结尾，表示后面的函数体与 def 语句之间有缩进关系（通常是 4 个空格）。函数体是函数每次被调用时执行的代码。return 语句是可选项，它可以在函数体内任何地方出现，表示函数执行到此结束，控制权返回给调用程序，同时返回处理结果。

在图 5-1 的程序中，第 1 行定义了一个名称为 Sum 的函数，形参为 a、b、c。第 2、3 行是函数体，其功能是计算参数 a、b、c 的和，return 语句返回求和结果。第 4 行 print 语句与第 1 行 def 语句没有缩进关系，因此不属于 Sum 函数范围，它的功能是调用 Sum 函数计算 10、2、3 的和并显示计算结果。在调用函数时，必须给函数传递具体的参数值，此时参数称为“实际参数”，简称实参。10、2、3 就是实参。实参可以是常量、变量、表达式、函数等，无论何种类型，在进行函数调用时都必须具有确定的值，才能让函数运行。

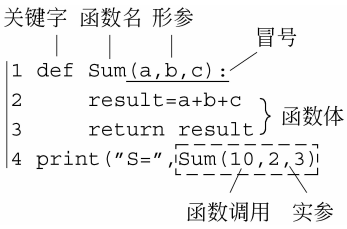

图 5-1 函数定义及调用示例

该程序的运行结果如下：

```
======RESTART: C:/Python36-32/图 5.1 程序.py======
S=15
```

如果将程序中的第 4 行放到第 1 行，结果如下：

```
======RESTART: C:/Python36-32/图 5.1 程序.py======
Traceback (most recent call last):
  File "C:/Python36-32/图 5.1 程序.py", line 1, in<module>
    print("S=",Sum(10,2,3))
NameError: name 'Sum' is not definedPython
```

Python 提示名字为“Sum”的函数未进行定义。这是因为在调用 Sum 函数前并没有定义该函数。因此调用函数时必须确保该函数之前已经定义。

再看两个函数定义的例子。

【例 5-1】 定义一个函数，将华氏温度转换为摄氏温度。

```
def fTemp (t):
    converTemp=(5/9) * (t-32)
    return converTemp
```

函数的参数是华氏温度，返回其对应的摄氏温度，例如：

```
>>>fahTemp=eval(input("请输入华氏温度："))
70
>>>celTemp=int(fTemp(fahTemp))
>>>print("转换后的摄氏温度是：", celTemp, "度")
转换后的摄氏温度是：21 度
```

代码中的 int()函数将摄氏温度浮点数转换为整数。

【例 5-2】 定义一个函数，提取中文姓名中的姓氏和名字。假设姓氏与名字之间有一个空格。

```
def getName(name):
    nameList=name.split()
    giveName=nameList[0]
    firstName=nameList[-1]
return giveName,firstName
```

该函数的参数是一个姓名字符串，姓氏与名字之间有一个空格。函数中 name. split()的功能是将字符串 name 按照空格分成不同元素保存到一个列表中，在本例中，列表有两个元素：姓氏和名字。函数返回姓氏和名字两个结果，在调用函数时应定义两个变量分别保存它们：

```
>>>Name=input("请输入姓名,姓氏和名字之间以一个空格分开：")
请输入姓名,姓氏和名字之间以一个空格分开：诸葛亮
>>>givename, firstname=getName(Name)
>>>print("姓氏：", givename, "名字：", firstname)
姓氏：诸葛 名字：亮
```

5.1.3 匿名函数

函数定义中有一类特殊函数，称为匿名函数或 lambda 函数。它使用 lambda 关键字，一般形式如下：

```
f=lambda 参数 1, 参数 2, …：表达式
```

这种函数省略了 def 声明函数的标准步骤，因此被称为匿名函数。其本质是一个表达式，能接收任何数量的参数并返回表达式的值。例如：

```
>>>f=lambda x, y, z: x+y+z
>>>f (1,2,3)
6
>>>g=lambda x: lambda y: x+y
```

```
>>>a=g(4)
>>>a(10)
14
>>>g(4)(10)
14
```

代码中 g 返回的是一个 lambda 函数 lambda y：x＋y。当 x＝4 时，a＝g(4)＝lambda y:4＋y。当 y＝10 时，a(10)＝4＋10＝14。两个表达式也可以简化为 g(4)(10)。

5.2 函数的参数传递

函数有三种方法将实参传递给形参，即按照位置传递参数、按照关键字传递参数、按照默认值传递参数。

5.2.1 按照位置传递参数

函数调用时默认采用该方法传递参数，其形式如下：

```
函数名 (参数 1,参数 2,…)
```

这种参数传递方法要求形参和实参的个数必须一致，并且一一对应，即相同位置的实参向相同位置的形参传递参数。当实参是一个表达式时，先要计算表达式的值，再将它传递给形参。例如图 5-1 的程序中 print 函数在调用 Sum 函数时就是按照位置传递参数：

```
Sum (10,2,3)
```

这里 a＝10，b＝2，c＝3。

5.2.2 按照关键字传递参数

当函数需要传递很多参数时，按照位置传递参数很容易出错，此时可以按照关键字传递实参，其形式如下：

```
函数名 (参数名 1=值 1,参数名 2=值 2,…)
```

这种参数传递方法在实参前添加了参数名，即关键字。关键字明确了每个参数的含义，这样即便参数顺序被打乱，参数的位置发生改变，也不会影响参数的传递。例如图 5-1 的程序如果需要对 10 个值求和，可以采用关键字传递实参。同时，关键字参数可以在函数中提供默认值。

```
Sum (a=10, b=2, c=3, d=…)
```

5.2.3 按照默认值传递参数

在定义函数时可以给形参指定默认值。在调用函数时如果形参提供了数值，则使用给定的参数值，如果没有提供数值，则会使用默认值。例如图 5-1 中的程序，如果函数定义如下：

```
def Sum (a,b,c=2)
    result=a+b+c
    return result
```

此时 Sum 函数有三个形参 a、b 和 c，其中 c 指定了默认值 3。如果调用函数如下：

```
>>>print ("S=", Sum (10,2,3))
S=15
```

结果为 15。因为调用函数时形参 c 提供了参数 3，c 等于 3，而不是默认值 2，求和结果是 10＋2＋3＝15。如果调用函数如下：

```
>>>print("S=", Sum(10,2))
S=14
```

结果为 14。因为此时实参 10、2 按照位置顺序分别传递给形参 a、b，而 c 没有提供实参，因此其值等于默认值 2，求和结果是 10＋2＋2＝14。要注意的是，如果函数定义时给某个形参指定了默认值，则该参数必须定义在无默认值的形参后面，因此 c 要定义在 a、b 之后。

5.2.4 可变数量的参数传递

某些情况下，在定义函数时无法确定参数个数，Python 允许函数设计可变数量的参数。此时函数定义形式如下：

```
函数名 (* 参数)
```

在参数名前面加一个“*”，表示参数是以形参名为标识符的元组，元组中的元素个数可以是零个、一个或多个。例如，定义函数：

```
def f(*a):
    print(a)
```

调用这个函数：

```
>>>f("Hello")
('Hello',)
```

结果表明参数 a 是一个以字符串“Hello”为元素的元组。又如下例：

```
>>>f(1,2,3)
(1, 2, 3)
```

返回结果是一个包含三个参数 1、2、3 的元组。如果不提供任何参数：

```
>>>f ()
()
```

则返回一个空元组。如果按照关键字传递参数，例如：

```
>>>f(a=1)
Traceback (most recent call last):
    File "<pyshell #257>", line 1, in<module>
        f(a=1)
TypeError: f() got an unexpected keyword argument 'a'
```

结果提示出错。此时应采用如下方式定义函数：

```
函数名 (**参数)
```

在参数名前加“**”，表示参数的数据类型是字典，其中关键字（参数名）为“键”，参数值为“值”。例如：

```
def f(**a):
    print(a)
```

此时再按关键字传递参数：

```
>>>f (x=1, y=2, z=3)
{'x': 1, 'y': 2, 'c': 3}
```

返回结果是一个字典参数，其中每个元素以关键字（参数名）为键、参数值为值。

综上所述，按照位置和按照关键字传递参数时，参数数量是固定不变的。如果参数数量不定或可变，需要在参数名前加“*”或“**”。加“*”表示参数是元组参数，加“**”表示参数是字典参数（键值对）。在定义函数时，可以混合使用多种参数传递方式，此时要遵循以下规则：

(1) 关键字参数应放在位置参数后面。

(2) 元组参数必须在关键字参数后面。

(3) 字典参数要放在元组参数后面。

在调用函数时,首先按位置顺序传递参数,其次按关键字传递参数。多余的非关键字参数传递给元组,多余的关键字参数传递给字典。

【例 5-3】 下列函数采用字典参数,观察函数调用结果。

```
def story(**info):
    print('人生苦短,{name}爱{language}'.format(**info))
```

调用函数,结果如下:

```
>>>story(name='Gudio', language='Python')
人生苦短,Gudio 爱 Python
>>>story(name='James', language='Java')
人生苦短,James 爱 Java
>>>params={'name':'Dennis','language':'C'}
>>>story(**params))
人生苦短,Dennis 爱 C
>>>del params['name']
>>>story(name='stroke of genius',**params)
人生苦短,stroke of genius 爱 C
```

【例 5-4】 函数定义如下:

```
def power (a, b, *c):
    if c:
        print('接收多余的参数: ',c)
    return a**b
```

调用函数,进一步理解元组参数的传递方法:

```
>>>power(2, 3)
8
>>>power(3, 2)
9
>>>power(b=3, a=2)
8
>>>arg1=(5, )*2
>>>power(*arg1)
3125
>>>arg2=(5,6)*2
>>>power(*arg2)
接收多余的参数: (5, 6)
```

```
15625
>>>power(3,3,'Hello,world')
接收多余的参数: ('Hello, world',)
27
```

函数 power 定义了三个形参,其中 a 和 b 是普通参数,c 是元组参数。if c 语句判断是否存在参数 c,存在的话给出提示并计算 a**b,否则直接计算 a**b。代码中 arg1=(5,)*2=(5,5),是一个元组。元组中的元素按照位置顺序赋值给形参,因此 power(*arg)等价于 power(5,5),没有参数 c,直接计算 5**5,返回 3125。arg2=(5,6)*2=(5,6,5,6),power(*arg2)等价于 power(5,6,5,6),这时参数 a=5,b=6,剩余的参数 5 和 6 组成元组传递给 c。因为参数 c 存在,所以会去执行 if 后面的语句,提示接收多余的参数 c,即(5,6),同时计算 5**6=15625。同样,power(3,3,'Hello,world')中参数 a 和 b 按顺序赋值为 3 和 3,第三个参数"Hello,world"传递给 c,也会执行 if 后面的语句,此时 c=('Hello,world',)。

【例 5-5】 按默认值传递参数,函数定义如下:

```
def f (x, y=None, z=1):
    if y is None:
        x, y=0, x
    result=[]
    i=x
    while i<y:
        result. append(i)
        i+=z
    return result
```

调用函数,结果如下:

```
>>>f(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>f(1,5)
[1, 2, 3, 4]
>>>f(1,10,2)
[1,3, 5,7, 9]
>>>power(*f(3,7))
接收多余的参数: (5, 6)
81
```

其中函数 power 的参数是*f(3,7),因为 f(3,7)返回一个列表[3,4,5,6],函数调用等价于 power(3,4,5,6),根据 power 函数定义,a=3,b=4、5 和 6 组成元组传递给*c,最后返回参数提示和 3**4 的结果 81。

5.3 函数的返回值

5.3.1 返回布尔值和列表的函数

在函数定义中，return 语句是可选项，主要功能是返回函数运行结果。返回的数据类型除常见的数值、字符串外，还可以是布尔值或列表等。

【例 5-6】 下面程序判断用户输入的英文单词首字母是否是大写字母，并给出提示。

```
def isBigLetter(word):
    if ord('A')<=ord(word[0])<=ord('Z'):
        return True
    return False
    str=input('请输入一个英文单词：')
    if isBigLetter(str):
        print('单词首字母是大写字母!')
    else:
        print('单词首字母不是大写字母!')
```

其中函数 isBigLetter 用来判断单词首字母是否为大写字母，return 语句返回一个布尔值，如果是大写字母返回 True，否则返回 False。函数中 word[0]表示字符串 word 中的第一个字符，也就是单词首字母。ord(x)是 Python 内置函数，用于将字符串 x 转换为其所对应的 Unicode 编码。因为大写字母 A～Z 在计算机中用编码 65～90 表示，如果字符编码在这个范围，就表示该字符是大写字母。

请注意，函数中出现了两个 return 语句。如果 if 语句条件成立，会去执行“return True”语句，然后函数结束运行，返回调用程序，不再执行“return False”语句。如果 if 语句条件不成立，会执行“return False”语句，再返回调用程序。当函数中有多条 return 语句时，执行完第一条 return 语句就会退出函数，不再执行其他 return 语句。

程序运行结果如下：

```
=======RESTART: C:/Python36-32/例 5.6.py=======
请输入一个英文单词：Python
单词首字母是大写字母!
>>>
=======RESTART: C:/Python36-32/例 5.6.py=======
请输入一个英文单词：python
单词首字母不是大写字母!
```

return 还可以返回列表。例如，如果要求显示用户输入的英文单词中包含的元音字母，可以用下面的程序：

【例 5-7】 显示英文单词中的元音字母。

```
def find_Vowels(word):
    word=word.lower()
    vow=('a','e','i','o','u')
    findVowels=[]
    for i in vow:
        if (i in word) and (i not in findVowels):
            findVowels.append(i)
    return findVowels
    str=input("请输入一个单词: ")
    vowelsinStr=find_Vowels(str)
    print(vowelsinStr)
```

为方便处理,函数 find_Vowels 首先使用 str.lower()方法将单词的所有字母转换为小写,再依次判断 5 个元音字母(a,e,i,o,u)是否包含在单词中。如果包含在单词中,就将其添加到列表 findVowels 中,最后返回这个列表。调用函数时需要定义一个变量保存列表,程序运行结果如下:

```
=======RESTART: C:/Python36-32/例 5-7.py=======
请输入一个单词: Gudio
['i', 'o', 'u']
```

5.3.2 无返回值的函数

函数中如果没有 return 语句,会自动返回 None,表示没有返回值。它的数据类型是 NoneType。例如:

```
def f(x,y):
    s=x+y
```

调用这个函数:

```
>>>result=f(2, 3)
>>>print(result)
None
>>>type(result)
<class 'NoneType'>
```

如果函数有 return 语句,但没有任何参数,则运行结果与上例相同。例如: