第5章 chapter 5

函　数

5.1 函数入门

在前面的章节中，使用过 Python 的内置函数和 Python 标准库中定义的函数，对使用函数名传入函数参数的函数调用方式已经不再陌生。这些函数实际上是系统已经定义好的、能够完成特定功能的代码段，只要知道函数的功能、函数名、函数的输入/输出方式，就可以在程序需要的位置正确地调用函数并利用其功能。但是我们并不知道也不需要知道函数内部实现的具体代码，所以函数是一种功能的抽象。

用户自定义函数和内置函数的本质是一样的，它们都是函数。本章主要讨论 Python 自定义函数。

5.1.1 函数的概念

函数(Function)是一个相对独立的实体，是对完成一定功能的代码段的封装，是一种功能的抽象。

定义一个函数后，可以在一个程序需要的位置多次调用该函数，调用时给出不同的参数就可以实现对不同数据的处理，也可以在不同的多个程序中调用，所以函数可以实现代码复用，这种代码复用可以减少程序代码量。当需要修改功能时，只要在函数中修改代码，所有调用位置的功能就会被同时更新，所以函数可以降低代码的维护难度。

【例 5.1】 分别对 3 和 5、6 和 8、12 和 28 求最大值。

当不使用函数时，程序代码如下。

```
1   x, y = 3, 5
2   if x > y:
3       result = x
4   else:
5       result = y
6   print("3 和 5 的最大值为{}".format(result))
7
8   x, y = 6, 8
9   if x > y:
10      result = x
```

```
11  else:
12      result = y
13  print("6 和 8 的最大值为{}".format(result))
14
15  x, y = 12, 28
16  if x > y:
17      result = x
18  else:
19      result = y
20  print("12 和 28 的最大值为{}".format(result))
```

此程序中，求最大值代码重复执行了 3 次，而除了初始时 x 和 y 的赋值不同，其余代码均相同。若引入求最大值函数，则代码可以精简为如下。

```
1   def calMax(x,y):
2       if x > y:
3           return x
4       else:
5           return y
6
7   print("3 和 5 的最大值为{}".format(calMax(3,5)))
8   print("6 和 8 的最大值为{}".format(calMax(6,8)))
9   print("12 和 28 的最大值为{}".format(calMax(12,28)))
```

可见，引入函数可以使程序更加简洁，有利于缩减代码量，提高重用性。要善于使用函数，以减少重复编写程序段的工作量。

此外，当编程实现一个较复杂的任务时，为了简化程序设计、便于组织和规划，一般会将大任务分解成一系列简单的小任务，每个小任务用一段相对独立的功能函数实现。引入函数后，只需要在主程序中合理地调用各个函数，就可以实现整个程序的功能。所以函数实现了问题的简化，降低了编程的难度。

例如，2019 级金融学一班有 50 位学生，已知其“程序设计”课程的考试成绩，要求统计该班的平均分、最高分、最低分、中位分，并按照考试成绩由高到低排序。由于该程序实现的功能较多、代码较长，因此在编程时可以根据其实现的不同功能将代码划分为不同的函数，如图 5-1 所示。

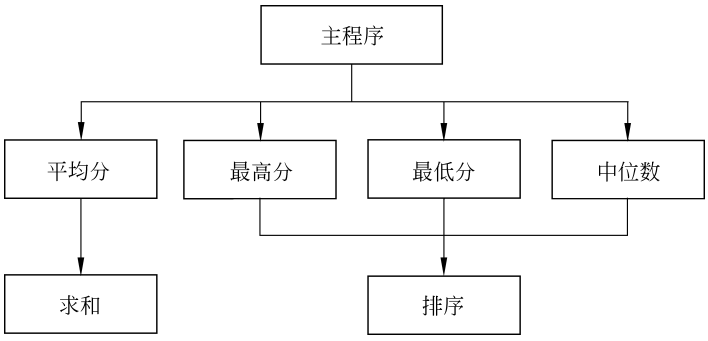

图 5-1　函数关系示意

图 5-1 中，程序的具体功能由各个不同的函数实现，主程序一般负责调用不同的函数，用来统筹全局。各个函数既可以被主程序直接调用，也可以被其他函数调用。例如要计算平均分，可以先调用求和函数得到总分后再计算平均分；要获取最高分、最低分和中位数，可以先把数据进行排序，这样就可以很好地简化编程。当然，如果主程序要输出总分和排序后的数据，则可以直接调用求和函数和排序函数。

5.1.2 定义函数

函数定义使用 def 保留字，包括函数头和函数体两部分，其语法格式如下。

```
def <函数名>(形参列表):
    <函数体>
```

第 1 行是函数头，以 def 开头，函数名是合法的 Python 标识符。一对圆括号中的形参列表可以包含零个、一个或者多个形式参数。当包含零个形式参数时，圆括号不能省略。形式参数就是函数头的圆括号中定义的参数，相当于函数的自变量，之所以称为形式参数，是因为函数定义时形式参数是使用只有名字、不占内存空间的虚拟变量实现的，等待函数调用时为其传值。

函数体是实现函数功能的语句的集合，是函数定义的主体部分。函数体的最后一条语句一般是 return 语句，用于将函数的返回值带回主调函数，并将程序流程从被调函数转向主调函数。

return 语句的语法格式如下。

```
return [<返回值列表>]
```

若函数没有返回值，则可以不带＜返回值列表＞，也可以直接缺省 return 语句，此时函数会有一个默认返回值 None。Python 允许函数有多个返回值，以逗号分隔。无论函数体有无 return 语句，函数执行结束后都会将控制权交还给调用者。

以求两个数的最大值函数为例，其函数定义如图 5-2 所示。

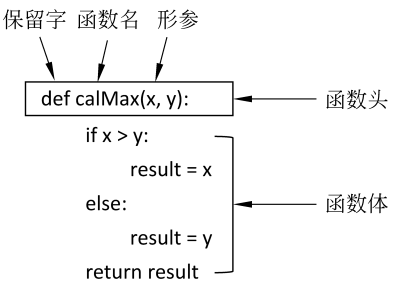

图 5-2 函数定义示例

图 5-2 中，calMax 是自定义的函数名，x、y 是两个形参，函数体内的 if-else 语句用于求 x、y 两个数的最大值，最后用 return 语句返回最大值 result。

5.1.3 调用函数

1. 函数调用和返回值

定义函数后，其函数体的代码并不会自动执行。只有当函数被调用时，才会执行该

函数的代码。调用函数的一般形式如下。

```
函数名([实参列表])
```

实参即实际参数。当发生函数调用时,实参会给形参传递值。如果调用的是无参函数,则实参列表可以省略,但圆括号不能省略。

函数调用通常有以下两种方式。

(1) 函数调用作为一条语句的一部分

该方式适用于有返回值的函数,调用者利用的是函数的返回值。例如 calMax()的函数调用可写成:y=calMax(5,8),将其函数返回值赋值给变量 y。

也可以用"print("Max={}".format(calMax(5,18)))"语句将函数返回值直接输出到屏幕上。

(2) 函数调用单独作为一条语句

这种方式通常适用于无返回值的函数,调用者利用的是函数代码执行的功能流程。

【例 5.2】 编写程序,分别为 Mary、John 和 Tom 致欢迎词。

程序代码如下。

```
1  def printHello(s):
2      print("Hello " + s + ",")
3      print("welcome to China!")
4
5  printHello("Mary")
6  printHello("John")
7  printHello("Tom")
```

该程序前 3 行代码为 printHello()的函数定义,无函数返回值(实际函数返回值为 None)。第 5～7 行是函数调用语句,即调用 printHello()函数 3 次。

执行结果如下。

```
Hello Mary,
welcome to China!
Hello John,
welcome to China!
Hello Tom,
welcome to China!
```

【例 5.3】 编程求任意两个数中的最大值并分析其执行流程。

程序的完整代码如下。

```
1  def calMax(x, y):
2      if x > y:
3          result = x
4      else:
5          result = y
```

```
6       return result
7
8   a, b = eval(input("请输入两个数:"))
9   m = calMax(a, b)
10  print("Max={}".format(m))
```

执行结果如下。

```
请输入两个数:5,8
Max=8
```

该程序代码的第 1～6 行是 calMax()的函数定义，第 8～10 行是主程序代码。当该程序运行时，由于前 6 行是函数定义，因此会从第 8 条语句开始执行。程序的执行流程如图 5-3 所示，图中的箭头方向表示程序的执行流向。

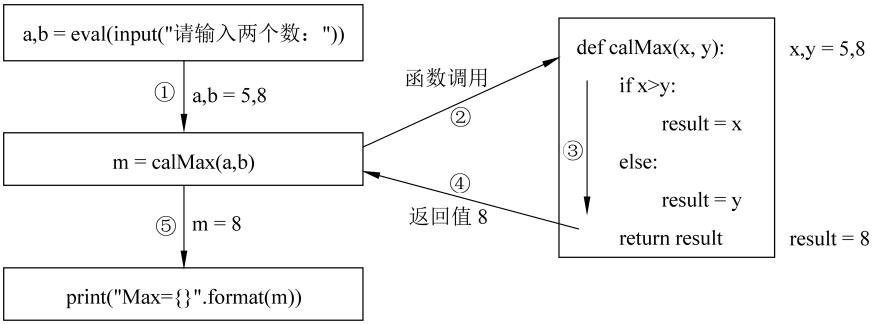

图 5-3 带函数调用的程序执行流程

执行流程分析如下。

① 执行第 8 条语句，即从键盘接收两个数并赋值给变量 a 和 b(假设为 5、8)；

② 执行第 9 条语句，由于发生了函数调用，这时主程序会暂停执行，将控制权转移给 calMax()函数。

③ 函数调用过程如下。

- 形参 x、y 分别接收实参 a、b 的值(x，y=5，8)。
- 执行 calMax(a,b)的函数体代码，求得最大值 result 为 8。
- 执行 return 语句，结束函数调用并返回主程序，其返回值为 8。

④ 执行第 9 条语句，将函数返回值 8 赋值给变量 m。

⑤ 执行第 10 条语句，输出“Max=8”，程序执行结束。

2. 函数的嵌套调用

Python 允许嵌套调用函数，即在调用一个函数的过程中又调用另一个函数。如图 5-4 所示，两层嵌套调用的执行流程如下。

① 执行主程序开始部分。

② 执行调用 f1()函数的语句，流程转向 f1()函数。

③ 执行 f1()函数的开始部分。

④ 执行调用 f2()函数的语句,流程转向 f2()函数。

⑤ 执行 f2()函数,直到 f2()函数结束。

⑥ 返回到 f1()函数中调用 f2()函数的位置。

⑦ 继续执行 f1()函数的后续代码,直到 f1()函数结束。

⑧ 返回到主程序中调用 f1()函数的位置。

⑨ 继续执行主程序的后续代码直到结束。

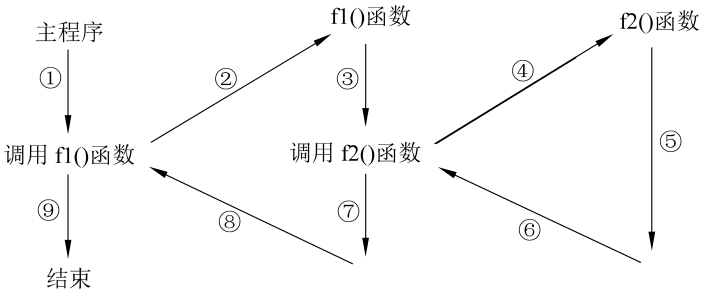

图 5-4 嵌套调用函数的执行流程

【例 5.4】 从键盘接收一个十进制整数,将其转换为十六进制数并输出。

分析:十进制数转换为十六进制数,可采用除以 16 取余数法。如十进制数 110,其转换步骤如下。

① 计算 110÷16,商为 6,余数为 14(即十六进制的 E)。

② 计算 6÷16,商为 0,余数为 6。

一旦商为 0,则转换结束。此时,110 对应的十六进制数为 6E。

程序代码如下。

```
1  def toHexStr(hexInt):
2      if 0 <=hexInt <=9:
3          return str(hexInt)
4      else:
5          return chr(ord('A') + hexInt - 10)
6
7  def decToHex(decInt):
8      hexStr = ""
9      while decInt !=0:
10         hexInt = decInt %16
11         hexStr = toHexStr(hexInt) + hexStr
12         decInt = decInt // 16
13     return hexStr
14
15 decInt = eval(input("请输入一个十进制整数:"))
16 hexStr = decToHex(decInt)
17 print("十进制数{}转换为十六进制数为:{}".format(decInt,hexStr))
```

运行结果如下。

```
请输入一个十进制整数:245
十进制数 245 转换为十六进制数为:F5
```

该程序采用了嵌套调用函数。程序中定义了两个函数,函数 decToHex()的功能是将十进制数转换为十六进制数。在转换过程中,需要将得到的余数转换为对应的十六进制字符。由于该功能相对独立,因此将其封装为函数 toHexStr(),并在 decToHex()函数中予以调用。当余数为 0～9 时,只需要用内置函数 str()直接将其转换为字符串类型即可;当余数为 10～15 时,则需要将其转换为对应的 A～F 字符。例如若余数为 14,则应转换为字符 E。

3. 多返回值函数

例 5.3 和例 5.4 中的函数只有一个返回值,实际上,函数也可以有多个返回值。下面通过例 5.5 介绍多返回值函数。

【例 5.5】 求两个整数的最大公约数和最小公倍数。

分析:由于最小公倍数为两个数的乘积除以最大公约数,因此该题的核心是求最大公约数。求最大公约数的方法有很多,这里采用辗转相除法(又称欧几里得算法),它是一种典型的迭代算法,具体步骤是:用较大数除以较小数,再用出现的余数(第一余数)除以除数,再用新出现的余数(第二余数)除以第一余数,如此反复,直到最后余数是 0 为止。最后得到的除数就是这两个数的最大公约数。

以 15 和 9 为例。

第 1 步计算 15%9,余数为 6。

第 2 步计算 9%6,余数为 3。

第 3 步计算 6%3,余数为 0。

算法结束,最大公约数为 3。辗转相除法的执行流程如图 5-5 所示。

程序代码如下。

```
1   def calGcdLcm(m, n):
2       a, b = m, n
3       if a <b:
4           a, b = b, a                 #a 存放大数
5       while b !=0:
6           temp = a %b
7           print("余数为:",temp)
8           a = b
9           b = temp
10      gcd = a                         #最大公约数
11      lcm = int(m * n / gcd)          #最小公倍数
12      return gcd, lcm                 #多返回值
13
14  x, y = eval(input("请输入两个整数:"))
```

```
15  z1, z2 = calGcdLcm(x, y)
16  print("最大公约数为:{}, 最小公倍数为:{}".format(z1,z2))
```

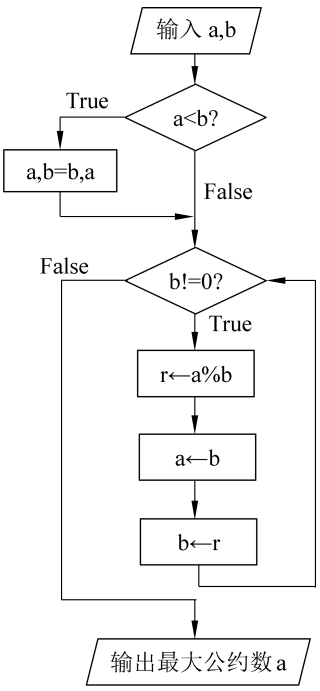

图 5-5　辗转相除法求最大公约数的流程图

由于 calGcdLcm()函数返回了最大公约数和最小公倍数两个值(第 12 行),因此在主程序中用 z1、z2 两个变量分别接收返回值(第 15 行)。

运行结果如下。

```
请输入两个整数:9,15
余数为: 6
余数为: 3
余数为: 0
最大公约数为:3, 最小公倍数为:45
```

第 15 和 16 行代码可以换成以下代码,输出效果完全相同。

```
15  z = calGcdLcm(x,y)
16  print("最大公约数为:{}, 最小公倍数为:{}".format(z[0], z[1]))
```

多返回值的函数返回的其实是一个元组,元组是一种由圆括号括起来的组合数据类型,例如上面的第 15 行代码返回的就是(3,45),赋值给 z 变量,z[0]和 z[1]分别为 3 和 45 这两个元素。关于元组,本书将在第 6 章详细介绍。

5.2 函数的参数

在5.1节中，我们初步了解了函数的实参和形参，本节将重点介绍实参和形参的各种形式，包括位置参数和关键字参数、参数默认值以及可变数量参数。

5.2.1 位置参数和关键字参数

图5-6是函数调用过程中的参数传递与返回值示意图。结合5.1节可以知道，函数定义中给出的是求x、y的最大值，但是此时x、y并无确定的值，只是形式上有这两个变量而已，因此x、y被称为形式参数(形参)。

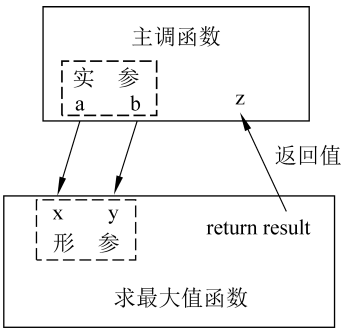

图5-6 函数的参数传递与返回值示意

当发生函数调用时，函数形参需要从主调函数那里获取实际的数据，以明确到底求的是哪两个数的最大值。此时主调函数给出的实际数据(放在变量a、b中)被称为实际参数(实参)。实参的值在函数调用时会传递给形参。

实参有两种类型：位置参数(positional argument)和关键字参数(keyword argument)。默认情况下采用的是位置参数，即按照实参的位置次序依次传值给对应的形参，图5-6中采用的就是位置参数，实参a、b按照次序依次传值给形参x、y，此时，实参必须与形参在顺序、个数和类型上相匹配。

当参数较多时，使用位置参数的可读性较差，使用起来很不方便。在这种情况下，可以采用关键字参数，如函数funStu()的定义如下。

```
>>>def funStu(name, gendor, age, score, city):
    print("name=",name)
    print("gendor=",gendor)
    print("age=", age)
    print("score=",score)
    print("city=",city)
```

若使用位置参数，则其正确的函数调用为

```
funStu("Mary","Female",18, 98, "Jinan")
```

但是由于参数较多，一般很难记住它们的顺序。此时就可以使用关键字参数进行函数调用，方法为

```
funStu(name="Mary", age=18, gendor="Female", city="Jinan", score=98)
```

由于在函数调用时指定了相应的关键字(即形参名称)，因此实参之间的顺序可以任

意调整，函数的使用会更加容易。

位置参数和关键字参数可以混合使用，但是需要注意：位置参数不能出现在关键字参数之后。例如，使用“funStu ("Mary","Female",18, city="Jinan", score=98)”进行函数调用是正确的，但是使用“funStu ("Mary", "Female", age=18, 98, "Jinan")”进行函数调用是错误的，原因就在于位置参数(98,"Jinan")不能在关键字参数(age=18)之后出现。

【例 5.6】 求两个二维坐标点之间的欧氏距离，要求实参使用关键字参数。

程序代码如下。

```
1  from math import sqrt
2
3  def f_dist(x1,y1,x2,y2):
4      return sqrt((x1 - x2) ** 2 + (y1 - y2) ** 2)
5
6  a,b,c,d = eval(input("请输入坐标值:"))
7  result = f_dist(x1 = a, x2 = b, y1 = c, y2 = d)
8  print("两点间的欧氏距离为:{:.2f}".format(result))
```

代码第 7 行调用 f_dist()函数时采用的就是关键字参数。

运行结果如下。

```
请输入坐标值:3,8,5,9
两点间的欧氏距离为:6.40
```

5.2.2 参数默认值

定义函数时可以为形参指定默认值。如函数 funStu(name, gendor, age, score, city)中的 city 参数表示学生所在的城市。若将 city 的默认值设定为 Beijing，那么在函数调用时，如果其对应的实参缺省，则形参会自动取其默认值。

修改 funStu()的函数头为

```
def funStu (name, gendor, age, score, city="Beijing"):
```

即为 city 指定了默认值 Beijing，其函数调用可以采用如下方式。

```
funStu ("Liming","Male",19, 87)
```

运行结果如下。

```
name=Liming
gendor=Male
age=19
score=87
city=Beijing
```