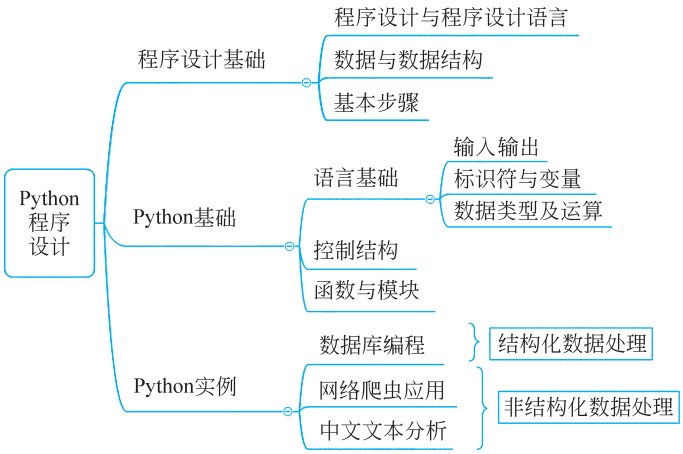
第5章

Python 程序设计

第 4 章介绍了算法，而程序是算法在计算机上的具体实现，是用计算机语言描述的某个问题的解决步骤。用计算机程序解决问题时，需要将算法用某种计算机程序设计语言精确描述（也称编写程序），并在计算机上调试运行直至正确，才能最终解决问题。

在计算机科学中，常见的程序设计语言有 Python、C++、Java、Visual Basic 等。同一个算法可以用不同的程序设计语言实现。尽管不同的程序设计语言特点不同，语法规则也可能不同，但是程序设计方法基本相同。

本章介绍了 Python 程序设计的基础知识，并给出了大量应用实例。通过本章的学习，读者可以使用 Python 语言编写程序，实现第 4 章的部分算法，从而更加深入地了解计算思维。



5.1 程序设计概述

5.1.1 程序

程序是计算机为完成某个任务遵循一定规则和算法思想组织起来并执行的一系列代码（也称指令序列）。通常，计算机程序需要描述两部分内容：一是描述问题涉及的每个对象及它们之间的关系；二是描述处理这些对象的规则。其中，前

者涉及数据结构的内容，而后者则是求解问题的算法。因此，对于程序的描述，可以用经典的公式来表示：

程序＝算法＋数据结构

一个设计合理的数据结构往往可以简化算法，而好的算法又使程序具有更高的执行效率，并使程序更加容易阅读和维护。

5.1.2　程序设计与程序设计语言

程序设计就是根据计算机要完成的任务，先设计解决问题的数据结构和算法，再编写相应的程序代码，并测试该代码运行的正确性，直到能够得到正确的运行结果为止。程序设计应遵循一定的方法和原则，良好的程序设计风格是程序具备可靠性、可读性、可维护性的基本保证。

在编写程序代码时，程序员必须遵循一定的规范描述问题的求解方法和解决步骤，这种规范就是程序设计语言。计算机程序设计语言具有一些基本原则，即固定的语法格式、特定的语义和使用环境，这些基本原则比日常语言更加严格，必须避免语言的二义性。

程序设计语言是编写程序所使用的计算机语言。随着计算机技术的发展，程序设计语言经历了从机器语言、汇编语言到高级语言的发展历程。

机器语言由二进制的 0、1 代码指令构成，能被计算机直接识别。但理解和记忆机器语言非常困难，并且容易出错，编程效率极低。

汇编语言是符号化的机器语言，采用英文助记符代替机器指令，比机器语言容易识别和记忆，从而提高了程序的可读性。但是汇编语言仍然是面向机器的语言，是为特定的计算机系统设计的，它要求软件工程师对相应的机器硬件非常熟悉，因而汇编语言属于低级语言。

高级语言更接近自然语言，并不特指某种语言，也不依赖于特定的计算机系统，因而更容易掌握和使用，通用性也更好。比较流行的高级语言有 Java、C/C++ 以及本书使用的 Python 等。用高级语言编写的程序可读性更强，也便于修改、维护。

高级语言的出现为计算机的应用开辟了广阔的前景。目前，很多人都在使用高级语言编写程序。高级语言有很多种，虽然它们的特点各不相同，但编程解决问题的过程，以及一些基本的程序设计规则和方法却是相似的。因此在学习某种语言后，应该具有将其中共性的思想和方法迁移到其他语言环境中进行问题求解的能力。

5.1.3　数据与数据结构

程序中的数据结构描述了程序中被处理的数据间的组织形式和结构关系。一般而言，数据结构与算法密不可分、相辅相成，一个良好的数据结构，将使算法简单化；而明确了问题的算法，才能更好地设计数据结构。

对于计算机程序设计，程序在实现算法的同时，还必须完整地体现作为算法操作对象的数据结构。对于复杂问题的求解，常常会发现由于数据的表示方式和数据结构的差异，对该问题的抽象求解算法也会完全不同。当然，对于同一个问题的求解，允许有不同的算法，也允许定义不同的数据结构。而依据不同算法编写的程序，其执行效率也会不一样。

5.1.4 程序设计的基本步骤

对于初学者,往往把程序设计简单地理解为只是编写一个程序,这是不全面的。程序设计反映了利用计算机解决问题的全过程,包含多方面的内容,而编写程序只是其中的一个方面。使用计算机解决实际问题,通常首先要对问题进行需求分析并建立数学模型,其次考虑数据的组织方式和算法,并用某种程序设计语言编写程序,最后上机调试程序,使其运行后产生预期的结果。具体要经过以下4个基本步骤。

(1) **需求分析**。要求计算机解决实际问题,先要认真分析问题的要求,明确问题的核心任务,需要解决的问题,达到的目标,输入输出等;再从已知条件出发,分析经过哪些处理才能解决问题。正确分析需求,可以为算法设计和编写程序打下良好的基础。

(2) **算法设计**。对于给定的问题,利用抽象和分解等计算思维,得出其求解方法,并且用详细的工具或步骤(流程图或伪代码等)描述求解过程。可以说,算法设计是程序设计过程中,也是解题过程中最关键的一步。它不仅是指计算的方法,而且包含从何处着手、解题步骤以及结果处理的全过程。

(3) **编写程序**。用程序设计语言实现算法的代码化,通俗地说就是编写程序。在计算机上,使用某种程序设计语言,把算法转换为相应的程序,然后提交给计算机执行。

(4) **上机调试**。编写程序后,还要对程序进行上机的测试和调试,发现并纠正程序中的各种错误,以保证它能正确运行。即使经过调试的程序,在使用一段时间后,仍然可能会被发现存在错误或不足之处,这就需要对程序做进一步的修改,使之更加完善。

5.2 Python语言基础


Python入门学习

Python语言是一种解释型、面向对象、动态数据类型的高级程序设计语言。从20世纪90年代初诞生至今,已逐渐成为最受欢迎的程序设计语言之一。Python简单易学,具有强大的数据处理能力,并且是一种通用的程序设计语言,既适合作为程序设计的入门语言,也适合作为解决数据分析等各类问题的通用工具。

以下从一个程序入手,开始动手编写程序,从而使读者对Python程序有一个基本认识。

5.2.1 引例

例5.1 设计一个成绩计算程序,对输入的考试成绩与平时成绩按7∶3的比例计算,并输出总评成绩。

1) 问题分析


例5.1

总评成绩和平时成绩与考试成绩的计算公式为

$$总评成绩=平时成绩\times30\%+考试成绩\times70\%$$

用程序解决这个问题的过程:首先接收输入的平时成绩和考试成绩,其次使用公式计算出相应的总评成绩,最后输出总评成绩。

本问题涉及3个数据:已知数据,即平时成绩、考试成绩;所计算的结果数据,即总评成绩。

代码如下:

```
a=float(input("平时成绩: "))          #输入平时成绩
b=float(input("考试成绩: "))          #输入考试成绩
c=a*0.3+b*0.7                         #计算总评成绩
print("总评成绩: ",round(c,2))        #输出总评成绩
```

2）说明

（1）该程序首先要求输入数据，其次对输入的数据进行处理，最后将计算结果输出。实际上，任何一个程序都要包括 3 要素：输入原始数据、对得到的数据进行处理加工和输出计算结果。

（2）计算依赖于输入的数据，但 input()函数的返回值是字符型的，所以要用 float()、int()或 eval()函数将其转换为数值型。为使显示的结果为小数点后保留两位小数，使用了四舍五入函数 round()。

（3）用缩进表示语句块。Python 中代码缩进是一种语法规则，强制用缩进格式表示代码间的层次关系，相同缩进的语句构成一个语句块。通常，语句末尾的冒号表示语句块的开始。在包含代码嵌套时，应注意同级语句块的缩进量要保持相同。

（4）代码注释。Python 中的代码注释有单行注释和多行注释，在运行程序时会忽略被注释的内容。单行注释用＃表示注释开始，＃之后的内容不会被执行。单行注释可以单独占一行，也可以放在语句末尾。多行注释是用 3 个英文的单引号或双引号作为注释的开始和结束符号。

（5）语句续行。通常，Python 中的一条语句占一行。在遇到较长的语句时，可使用续行符号\，将一条语句写在多行中。注意在\符号后不能有任何其他符号，包括空格和注释。另一种续行方式是在使用圆括号时，圆括号中的内容可以分多行书写，圆括号中的空白和换行符都会忽略。

（6）语句分隔。Python 使用分号分隔语句，从而将多条语句写在一行。如果冒号之后的语句块只有一条语句，Python 允许将语句写在冒号之后。同时，冒号之后也可以是分号分隔的多条语句。

3）运行

使用 Python 语言编写程序时，需要严格遵守 Python 语言的语法规则，并选择合适的程序运行环境运行程序。编写 Python 程序比较方便的方式是使用集成开发环境(Integrated Development Environment，IDE)。运行 Python 程序有两种方式：交互式和文件式。交互式是指 Python 解释器即时响应用户输入的每条代码，给出输出结果。文件式是指用户将 Python 程序写在一个文件中，然后启动 Python 解释器批量执行文件中的代码。交互式一般用于调试少量代码，文件式是主要的编程模式。

打开 IDLE，会出现交互式解释器 Python Shell，如图 5.1 所示，可以通过它在 IDLE 内部执行 Python 命令，也可以在 Python Shell 的提示符>>>后输入任意的语句、表达式或者小段代码进行测试。

除此之外，IDLE 还带有一个 Python 编辑器，如图 5.2 所示，可以用来编辑 Python 程序。通过选择 Python Shell 菜单 File→New File 命令，打开编辑器，输入**例 5.1** 相应的 Python 程序，程序文件以 py 为扩展名保存。通过命令编辑器菜单 Run→Run Module 命令，运行程序。程序运行时，在解释器 Python Shell 的交互界面中输入相应数据，可得到如

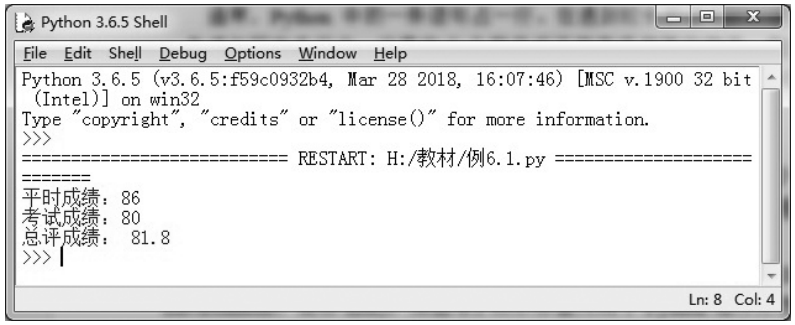


图 5.1 交互式解释器 Python Shell

下结果:

```
>>>
平时成绩: 86
考试成绩: 80
总评成绩: 81.8
```

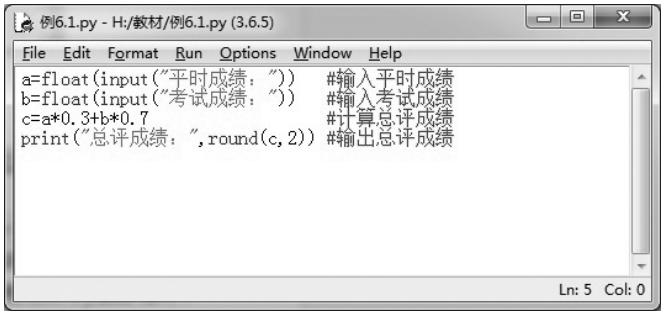


图 5.2 Python 编辑器

5.2.2 输入输出

1. 输入函数 input()

input()函数用于获得用户输入数据,其基本格式如下:

```
变量=input("提示字符串")
```

其中,变量和提示字符串均可省略。input()函数将用户输入以字符串返回。如果需要输入整数或小数,则需要使用 eval()、int()或 float()函数进行转换。

例 5.2 input()函数输出示例。

```
a=input("a:")
b-input("b:")
c=a+b
d=eval(a)+eval(b)
print(c,d)
```

2. 输出函数 print()

Python 中使用 print()函数完成基本输出操作。print()函数基本格式如下：

```
print([obj1,…][,sep=""][,end=""][,file=sys.stdout])
```

(1) 省略所有参数，输出一个空行。

(2) 输出一个或多个表达式，表达式之间用逗号分隔。

(3) 指定输出分隔符。print()函数默认分隔符是空格，可用 sep 参数指定特定符号作为输出对象的分隔符。

(4) 指定输出结尾符号。print()函数默认以回车符或换行符作为输出结尾符号，可以用 end 参数指定输出结尾符号。

(5) 输出到文件。print()函数默认输出到标准输出流，可用 file 参数指定输出到特定的文件。

例 5.3　print()函数输出示例。

```
print() #输出空行
print(1,2,3) #输出多个表达式
print(1,2,3,sep="*") #指定输出分隔符
print(1,2,3,end="*\n") #指定输出结尾符号
print(1,2,3, sep="*",end="*\n") #同时指定输出分隔符和结尾符号
```

5.2.3　标识符与变量

1. 标识符

在 Python 语言中，用来对变量、函数、类等数据对象命名的有效字符串序列统称为标识符。Python 语言规定标识符只能由字母、数字和下画线组成，且第一个字符必须为字母或下画线，不能以数字开头。标识符区分大小写，且其中不能出现标点符号或运算符。在 Python 中，可以用中文作为变量名，非 ASCII 标识符也是允许的。

在 Python 中，有一部分标识符是关键字，也称保留字，是 Python 语言本身的一部分，变量、函数、类等数据对象的命名不能与 Python 的关键字相同。Python 的标准库提供了一个 keyword 模块，可以输出当前版本 Python 的所有关键字，命令语句如下：

```
import keyword
print(keyword.kwlist)
```

输出：

```
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
'def','del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
'if','import','in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise',
'return', 'try','while', 'with', 'yield']
```

2. 变量

变量是指其值可以改变的量,每个变量都有一个变量名,对应计算机内存中具有特定属性的一个存储单元。该单元用来存储变量的值,在程序运行期间,这个单元中的值是可以改变的。变量通过变量名访问,变量的命名必须遵循标识符的命名规则。

Python是动态类型语言,变量不需要显式声明数据类型,对其直接进行赋值即可使用,Python语言的解释器会根据变量的赋值自动确定其数据类型。通过内置的type()函数,可以测试一个变量的数据类型。

Python允许同时为多个变量赋值,例如:

```
a=b=c=1
a,b,c=1,2,3
```

例5.4 变量赋值示例。

```
a,b,c=1,2.0,"3"
print(type(a),type(b),type(c))          #输出a、b、c的数据类型
d=a+b
e=str(a)+c                              #转换a为字符型
f=str(b)+c                              #转换b为字符型
g=a+int(c)                              #转换c为整型
h=b+float(c)                            #转换c为浮点型
print(d,e,f,g,h)
a,b=b,a                                 #交换a、b的值
print(a,b)
```

5.2.4 数据类型及运算

1. 数字

Python支持多种数字类型,包括布尔型、整型、浮点型、复数。

(1) 布尔型(boolean)只有两个值:True或False。对于值为零的任何数字或空集(空列表、空元组、空字典),在Python中的布尔值都是False。在数学运算中,True和False分别对应于1和0。bool()是布尔型的转换函数,可以将其他数据类型转换为布尔型。

(2) 整型(int)一般以十进制表示。Python也支持八进制、十六进制或二进制来表示整型。八进制整型以数字0o或0O开始,十六进制整型以0x或0X开始,二进制整型以0b或0B开始。int()是整型的转换函数,可以将其他数据类型转换为整型,其最为常见的用法是将包含整数的字符串转换为整数。

(3) 浮点型(float)也称小数,可以直接用十进制或科学记数法表示。浮点数通常都有一个小数点和一个可选的后缀e(大写或小写,表示科学记数法)。在e和指数之间可以用+或-表示正负,正数+可以省略。float()是浮点型的转换函数,可以将其他数据类型转换为浮点型。

(4) Python还支持复数,复数由实数部分和虚数部分构成,可以用a+bj或者complex(a,b)

表示，复数的实部 a 和虚部 b 都是浮点型。

在数字运算中，如果两个操作数类型不一致，Python 会进行数字类型的强制转换，即会强制将一个操作数转换为与另一个操作数相同的数据类型。类型转换的基本原则：整型转换为浮点型，非复数转换为复数。

具体规则如下。

（1）如果两个操作数是同一种数据类型，则无须进行类型转换。

（2）如果有一个操作数是复数，则另一个操作数被转换为复数。将非复数转换为复数，只需加个 0j 的虚数部分。

（3）如果有一个操作数是浮点型，则另一个操作数被转换为浮点型。将整型转换为浮点型，只要在后面加个.0 即可。

数字类型的数据可以进行多种运算。表 5.1～表 5.4 分别为算术运算符、比较运算符、逻辑运算符和数字运算函数。

表 5.1　算术运算符

运　算　符	功　能	运　算　符	功　能
expr1＋expr2	加	expr1//expr2	整除
expr1－expr2	减	expr1％expr2	取余
expr1 * expr2	乘	expr1 * * expr2	求幂
expr1/expr2	除		

表 5.2　比较运算符

运　算　符	功　能	运　算　符	功　能
expr1＜expr2	expr1 小于 expr2	expr1＞＝expr2	expr1 大于或等于 expr2
expr1＞expr2	expr1 大于 expr2	expr1＝＝expr2	expr1 等于 expr2
expr1＜＝expr2	expr1 小于或等于 expr2	expr1！＝expr2	expr1 不等于 expr2

表 5.3　逻辑运算符

运　算　符	功　能
not expr	expr 的逻辑非，优先级高于 and 和 or
expr1 and expr2	expr1 和 expr2 的逻辑与
expr1 or expr2	expr1 和 expr2 的逻辑或

表 5.4　数字运算函数

函　数	功　能
abs(num)	返回 num 的绝对值
max(num1，num2，…)	返回若干数字中的最大值

续表

函　　数	功　　能
min(num1,num2,…)	返回若干数字中的最小值
divmod(num1,num2)	返回一个元组(num1//num2,num1%num2)
pow(num1,num2,mod=1)	取 num1 的 num2 次方,再对 mod 取余
round(flt,ndig=0)	对浮点型 flt 进行四舍五入,保留 ndig 位小数

例 5.5　数字运算示例。

```
a=5+4                  #加法,结果是整数 9
b=4.3-2                #减法,结果是浮点数 2.3
c=3*7                  #乘法,结果是整数 21
d=2/4                  #除法,结果是浮点数 0.5
e=2//4                 #整除,结果是整数 0
f=17%3                 #取余,结果是整数 2
g=2**5                 #乘方,结果是整数 32
print(a,b,c,d,e,f,g)
```

2. 字符串

字符串是字符的有序序列,字符串中的字符按顺序排列。在 Python 中,字符串可以用单引号、双引号或三引号括起,但必须配对,其中三引号既可以是三个单引号,也可以是三个双引号。

反斜线本身具有特殊含义,它是转义字符的开头,如果字符串本身就包含反斜线字符,则需要用\\表示。换行符是一种特殊的字符,无法用普通字符形式表示,而用\n(newline)表示,这种字符称为转义字符(见表 5.5),用反斜线开头。制表符\t(tab)也是一种常用的转义字符,其功能是在不使用表格的情况下在垂直方向按列对齐文本。

表 5.5　转义字符

转义字符	含　义	转义字符	含　义
\n	换行符	\"	双引号
\r	回车符	\'	单引号
\t	制表符	\\	反斜线

字符串类型支持比较运算,比较是按照字符编码值的大小进行的。除此之外,常用的字符串运算符还包括连接运算符、切片运算符、成员运算符,如表 5.6 所示。

表 5.6　字符串运算符

运　算　符	功　　能
+	连接运算符,字符串连接
*	连接运算符,重复输出字符串

续表

运　算　符	功　　能
[]	切片运算符，通过索引获取字符串中字符
[:]	切片运算符，截取字符串中的一部分，遵循左闭右开原则
in	成员运算符，如果字符串中包含给定的字符返回 True
not in	成员运算符，如果字符串中不包含给定的字符返回 True

(1) 连接运算符的作用是把一个序列和另一个相同类型的序列连接。对于字符串类型，就是把两个或更多个字符串连接成一个更长的字符串。连接运算符用加法运算符表示。对一个字符串做几次重复的连接，这种操作用乘法运算符表示，另一个操作数是整数，表示重复的次数。

字符串中的字符按顺序编号，最左边字符的序号为 0，最右边字符的序号比字符串的长度小 1。Python 还支持在字符串中使用负数从右向左进行编号，最右边字符(即倒数第 1 个字符)的序号为－1。字符在字符串中的序号也称下标或索引，可以通过索引获取字符串中的字符。

(2) 切片运算符的作用是通过指定下标或索引范围获得一个序列的一组元素。对于字符串类型，就是取出已有字符串中的一部分(子串)成为一个新的字符串。切片运算符的描述形式为 s[m:n:d]，得到在 s[m]到 s[n－1]的范围内按 d 的步长选出字符而形成的字符串。其中，s 是字符串，m、n、d 都是整数，切片描述中必须包含冒号，但 m、n、d 都可以省略。m 省略时默认为 0(从头开始)；n 省略时默认为字符串长度(直到末尾)；d 省略时默认为 1(按顺序选出字符)；如果都省略，则表示整个字符串。

(3) 成员运算符是用来判断一个元素是否属于一个序列的。对于字符串类型，就是判断一个字符(也可以是一个子串)是否出现在一个字符串中。成员运算符用 in 或 not in 表示，返回值是布尔值 True 或 False。

例 5.6　字符串运算示例。

```
str="Python"
print(str+str)                          #连接
print(str*3)                            #重复
print(str[1],str[-1])                   #截取单个字符
print(str[1:],str[:-1])                 #切片
print(str[1:-1],str[1:-1:2])            #切片
print(str[-1:1:-1],str[-1:1:-2])        #切片
print('P' in str,'P' not in str)        #成员判断
print('p' in str,'p' not in str)        #成员判断
```

Python 提供了一系列关于字符串的函数和方法，如表 5.7、表 5.8 所示。它们的区别在于，字符串函数的参数是字符串，而字符串方法是隶属于字符串这个类的功能，调用方法是点成员(例如，字符串.方法)的方式。