高等教育立体化精品教材

Python程序设计基础

金　伟　胡　悦　主　编
葛　娅　朱晓芸　副主编
李兴粉　颜家炼　参　编

汕頭大學出版社

图书在版编目（CIP）数据

Python 程序设计基础 / 金伟，胡悦主编．—汕头：汕头大学出版社，2021.8

ISBN 978-7-5658-4417-1

Ⅰ.①P… Ⅱ.①金… ②胡… Ⅲ.①软件工具—程序设计—高等学校—教材 Ⅳ.①TP311.561

中国版本图书馆 CIP 数据核字（2021）第 162929 号

Python 程序设计基础 Python CHENGXU SHEJI JICHU

主　　编： 金　伟　胡　悦
责任编辑： 胡开祥
责任技编： 黄东生
封面设计： 易　帅
出版发行： 汕头大学出版社
广东省汕头市大学路 243 号汕头大学校园内　邮政编码：515063
电　　话： 0754-82904613
印　　刷： 河北铄柠印刷有限责任公司
开　　本： 787mm×1092mm　1/16
印　　张： 11.5
字　　数： 245 千字
版　　次： 2021 年 8 月第 1 版
印　　次： 2021 年 8 月第 1 次印刷
定　　价： 32.50 元
ISBN 978-7-5658-4417-1

PREFACE 前言

Python 是一门优雅而健壮的编程语言，它继承了传统编译语言的强大性和通用性，同时借鉴了简单脚本和解释语言的易用性。随着人工智能时代的到来，Python 已经成为学习编程的首选语言。本书是 Python 编程的入门教材，适合零基础的读者学习 Python 的基本语法和编程技巧。

本书在内容组织上以实例为载体，注重描述 Python 编程的基本知识点和技能点，由浅入深，通俗易懂，能让读者快速入门和进阶。全书共分 13 章，具体内容如下。

第 1 章为 Python 概述，主要介绍 Python 的应用场景、搭建 Python 开发环境、创建绘画程序等方面的知识。

第 2 章为 Python 基础，主要介绍变量、字符串、数字、注释和运算符等方面的知识。

第 3 章为 Python 进阶，主要介绍条件语句、循环语句、列表、元组、字典和集合等方面的知识。

第 4 章为 Python 面向对象，主要介绍类和实例对象的创建、Python 对象销毁、类的继承、方法重写、类属性与方法等方面的知识。

第 5 章为 I/O 处理，主要介绍屏幕打印、键盘输入的读取、文件的打开和关闭、文件的读写、文件的重命名和删除、Python 目录等方面的知识。

第 6 章为异常处理，主要介绍 Python 标准异常、异常处理、异常的参数等方面的知识。

第 7 章为正则表达式，主要介绍 re. match 函数、re. search 函数、检索和替换、正则表达式的修饰符、正则表达式的模式等方面的知识。

第 8 章为 CGI 编程，主要介绍网页浏览、Web 服务器支持及配置、HTTP 头部、CGI 环境变量、GET 和 POST 方法等方面的知识。

第 9 章为 MySQL 数据库应用开发，主要介绍 MySQLdb 模块的安装、数据库的连接、数据库表的创建、数据库插入操作、数据库查询操作、数据库更新操作、事务的执行、错误处理等方面的知识。

第 10 章为多线程，主要介绍使用 Threading 模块创建线程、线程同步、线程优先级队列等方面的知识。

第 11 章为 Python XML 解析，主要介绍使用 SAX 解析 XML、使用 xml. dom 解析 XML 等方面的知识。

第 12 章为 Python 网络编程，主要介绍 SocketServer 模块、多连接和 Twisted 框架等方面的知识。

第 13 章为 Python 图形化界面设计，主要介绍窗体控件布局、tkinter 常见控件的特征属性和事件响应等方面的知识。

由于编者水平有限，书中难免存在不妥与疏漏之处，敬请广大读者批评指正。

编　者

CONTENTS 目录

第 1 章　Python 概述

1.1 初识 Python

严格来说，Python 是一门跨平台、开源、免费的解释型高级动态编程语言。除了解释执行，Python 还支持伪编译，可以将源代码转换为字节码，以优化程序、提高运行速度和对源代码进行保密，并且支持使用 py2exe、PyInstaller、cx_Freeze 或其他类似工具，将 Python 程序及其所有依赖库打包为扩展名为 .exe 的可执行程序，从而脱离 Python 解释器环境和相关依赖库，在 Windows 平台上独立运行；Python 支持命令式编程、函数式编程，完全支持面向对象程序设计，并且拥有大量的、几乎支持所有领域应用开发的成熟扩展库。

Python 又称“胶水语言”，因为它可以把多种编程语言编写的程序融合到一起，实现无缝拼接，更好地发挥不同编程语言和工具的优势，满足各应用领域的需求。

1.1.1 认识 Python 编程

Python 的主要优点是语法简单，可以用较少的代码实现复杂的系统。Python 的实际应用场景主要有以下几种。

1 Web 开发

虽然 PHP、JavaScript 依然是 Web 开发的主流语言，但是 Python 的 Web 开发框架已逐渐成熟，如 Django、Flask 等。

2 网络爬虫

网络爬虫是一种自动提取网页信息的程序。在 Python 中，requests 库用于抓取网页数据，Beautiful Soup 库用于解析网页。

3 计算与数据分析

Python 在人工智能领域中是主要的编程语言。其中，Google 公司的 TensorFlow 和 Facebook 公司的 PyTorch 都采用了 Python。在数据分析中，Python 在可视化方面有相当完善和优秀的库，如 NumPy、SciPy、Matplotlib、Pandas 等，这可以满足 Python 程序员编写科学计算程序的需要。

4 游戏开发

Python 和 Pygame 库是开发图形化计算机游戏的得力工具。Pygame 库使开发 2D 图形化程序变得很容易，而且它可以免费下载和使用。

1.1.2 如何学好 Python

编程语言的学习，最初是从模仿开始的，照着书上的例子一遍一遍地实现。入门之初，有时一个一个字符地敲，也不能正确输入简单的程序，能发现错误，就会有收获！学会查阅 Python 帮助文档，总结每次报错提示并不断实践，才能真正地理解 Python 编程。

1.2 搭建一个属于自己的开发环境

1.2.1 搭建 Windows 下的开发环境

从 Python 3.5 开始的 Python 版本已不支持 Windows XP 系统。下面介绍如何在 Windows 7 系统中搭建 Python 开发环境。

1 下载 Python 安装包

在浏览器中打开 Python 官方网站（www.python.org），单击 Downloads 选项，单击 Python 3.8.3 按钮，如图 1-1 所示。在打开的对话框中选择保存位置，浏览器会自动下载安装包。

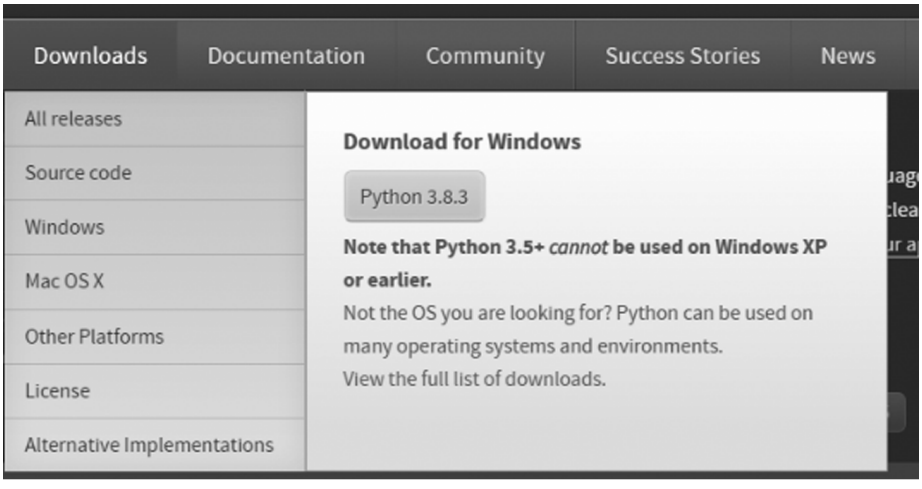

图 1-1 Python 安装包下载

提 示

目前，Python 有两个版本：Python 2.x 和 Python 3.x。这两个版本是不兼容的。由于 Python 2.x 不再更新维护，本书将以 Python 3.x 为基础进行讲解。

2 安装 Python

（1）打开 Python 3.8.3 所在目录，双击下载的文件。

（2）在打开的对话框中勾选 Add Python 3.8 to PATH，然后单击 Customize installation 选项，如图 1-2 所示。

图 1-2 安装 Python

（3）保持默认设置不变，直接单击 Next 按钮，如图 1-3 所示。

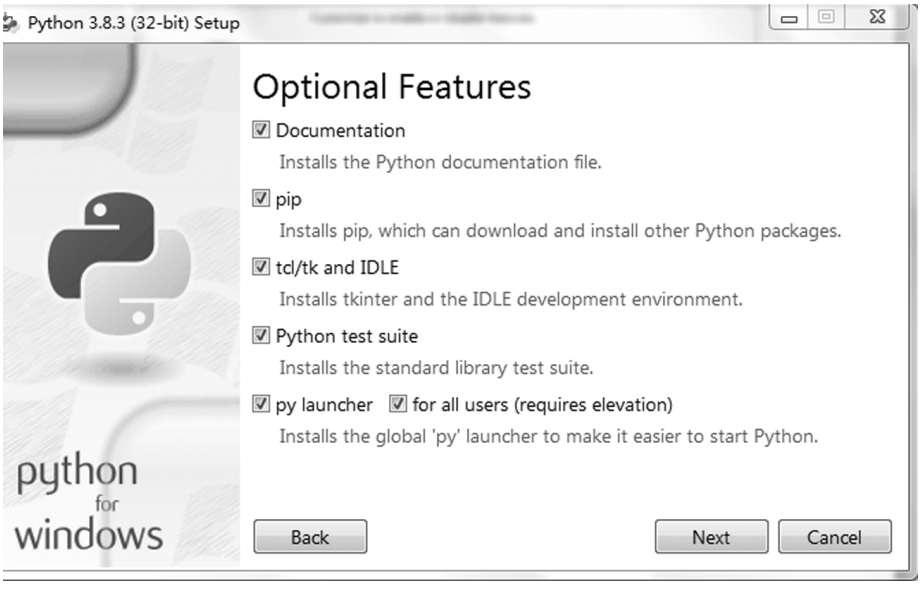

图 1-3 直接单击 Next 按钮

（4）单击 Browse 按钮，自定义安装路径；单击 Install 按钮，安装程序开始自动安装，如图 1-4 所示。

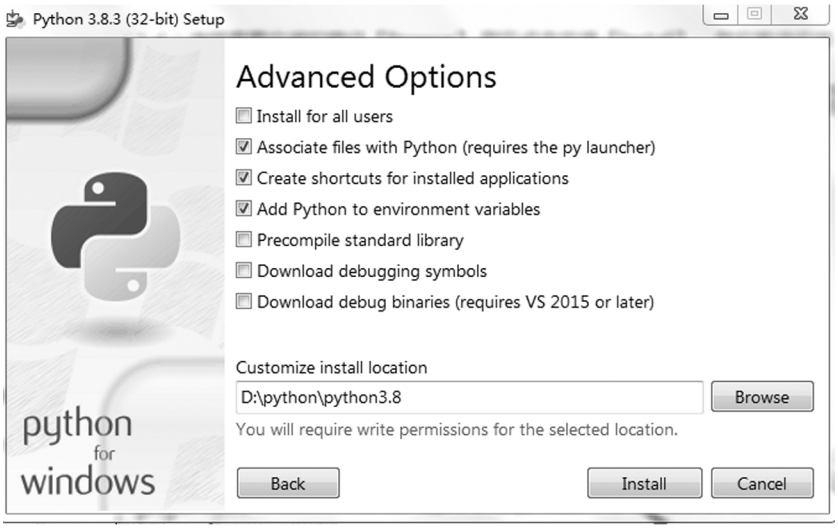

图 1-4　自定义安装路径

（5）安装成功后，会出现如图 1-5 所示的界面。

图 1-5　安装成功

3 运行 Python

在【开始】菜单的搜索框中输入“cmd”，单击 cmd.exe 选项，在打开的命令提示符窗口中输入“python”，并按 Enter 键。若 Python 安装成功，则窗口中将显示 Python 版本信息，如图 1-6 所示。Python 版本信息的结尾有符号“>>>”。“>>>”称为 Python 命令提示符，表示 Python 在等待用户输入代码。如果现在输入一行 Python 代码，Python 就会执行该代码。这种模式称为 Python 交互模式。

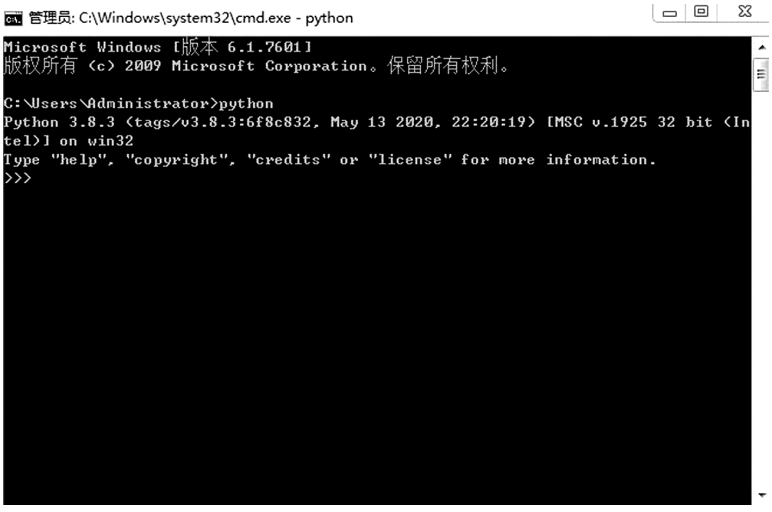

图 1-6 Python 版本信息

提 示

如果安装过程中出现 Setup failed，请按以下方法解决。

解决方法 1：在【开始】菜单的搜索框中输入“windows update”，下载安装 Windows 7 Service Pack 1。

解决方法 2：单击 Setup failed 界面中的 log file 选项，查看系统中缺失的文件，然后在微软公司官网中搜索该文件并下载安装。

1.2.2 搭建 Mac 下的开发环境

1 下载 Python 安装包

在浏览器中打开 Python 官方网站，单击 Downloads 选项，在 Mac OS X 选项下单击 Python 3.8.3 按钮，如图 1-7 所示。选择 python-3.8.3-macosx10.9.pkg，选择保存位置，浏览器会自动下载安装包。

图 1-7 下载 Python 安装包

2 安装 Python

（1）安装过程共有 7 个步骤。从【介绍】界面（见图 1-8）到【目的宗卷】界面，单击【继续】按钮。

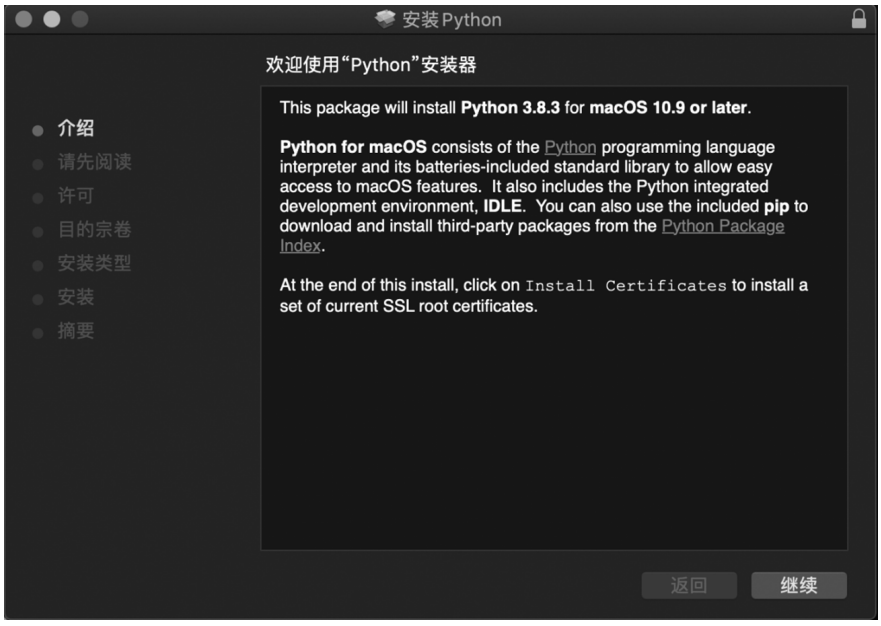

图 1-8　安装 Python

（2）在【安装类型】界面中单击【安装】按钮，输入密码，允许此次操作，安装程序开始自动安装，如图 1-9 所示。

图 1-9　安装过程

（3）图 1-10 所示为安装成功界面。安装成功之后，可以在应用程序中找到 Python 图标。

图 1-10 安装成功

1.2.3 安装 Python IDE

在 Python 交互模式中，输入的代码不会被保存。当输入较长的代码时，这个方法就会非常不方便。此时，用户可以考虑使用集成开发环境（IDE）来编写程序。

1 什么是 Jupyter Notebook

Jupyter Notebook 是一个开源的 Web 应用程序，可以创建和共享包含实时代码、方程式、可视化和说明文本的文档。简而言之，Jupyter Notebook 以浏览器打开，可以在网页页面中编写和运行代码。

2 在 Windows 系统中安装 Jupyter Notebook

在命令提示符窗口中输入命令“pip install jupyter notebook”，系统会自动开始安装。

安装成功之后，在该窗口中输入命令“jupyter notebook”，系统会自动打开浏览器，如图 1-11 所示。

图 1-11 Jupyter Notebook 界面

3 在 Windows 系统中修改 Jupyter Notebook

（1）在命令提示符窗口中输入命令“jupyter notebook --generate-config”，得到 jupyter_notebook_config.py 文件的路径，即保存在 C 盘 Users 目录下的 Administrator 文件夹中，如

图 1-12 所示。

```
C:\Users\Administrator>jupyter notebook --generate-config
Writing default config to: C:\Users\Administrator\.jupyter\jupyter_notebook_conf
ig.py
```

图 1-12　生成 config 文件

（2）找到 jupyter_notebook_config.py 文件并右击，在打开的快捷菜单中选择 Edit with IDE 命令，输入以下 Python 代码，如图 1-13 所示。

```
import webbrowser
webbrowser.register("chrome",None,webbrowser.GenericBrowser(u"C:\\Program Files(x86)\\Google\\Chrome\\Application\\chrome.exe"))
c.NotebookApp.browser ='chrome'
```

图 1-13　编辑 config 文件

这里是以 Google Chrome 浏览器为例，如果需要用到其他浏览器，可以找到该浏览器的 .exe 文件的路径，然后输入到 GenericBrowser 后面的括号中，并在 c.NotebookApp.browser 的“=”后输入该浏览器的名称。

4 在 Mac 系统中安装 Jupyter Notebook

在【终端】中输入命令“pip3 install jupyter notebook”，安装完成之后，在【终端】中输入命令“jupyter notebook”，即可打开 Jupyter Notobook。

1.3 第一个 Python 项目——一分钟创建一个绘画程序

turtle 模块是 Python 中一个用于绘制图形的函数库，它像一个小乌龟，在一个横轴为 x、纵轴为 y 的坐标系中，从(0,0)位置开始，通过一组函数的控制，在这个平面坐标系中移动，从而在其爬行的路径上绘制出图形。它是 Python 自带的，无须再次安装。

1.3.1 认识 turtle 模块

1 画布（canvas）：画图区域

用户可以自定义画布，代码如下。

```
turtle.screensize(canvwidth=None,canvheight=None,bg=None)
```

其中，参数分别为画布的宽度（单位为像素）、高度和背景颜色。例如：

```
turtle.screensize(canvwidth=800,canvheight=700,bg='green')
```

此时，画布的宽度是 800 像素，高度是 700 像素，背景颜色为绿色。

2 绘图命令

绘图命令主要分为两种：画笔运动命令和画笔控制命令。

（1）常用的画笔运动命令如表 1-1 所示。

表 1-1 常用的画笔运动命令

命 令	说 明
turtle. forward(distance)	向当前画笔方向移动 distance 像素长度
turtle. backward(distance)	向当前画笔相反方向移动 distance 像素长度
turtle. right(degree)	顺时针旋转 degree 度
turtle. left(degree)	逆时针旋转 degree 度
turtle. pendown()	移动时绘制图形，默认时也为绘制
turtle. goto(x,y)	将画笔移动到坐标为(x,y)的位置
turtle. penup()	移动时不绘制图形，用于另起一个地方绘制
turtle. speed(speed)	设置画笔的移动速度，取值为[0,10]中的整数
turtle. circle()	画圆，半径为正（负）数，表示圆心在画笔的左边（右边）

（2）常用的画笔控制命令如表 1-2 所示。

表 1-2 常用的画笔控制命令

命 令	说 明
turtle. pensize(width)	绘制图形时的宽度
turtle. pencolor()	设置画笔颜色
turtle. fillcolor(colorstring)	绘制图形的填充颜色
turtle. color(color1,color2)	同时设置 pencolor＝color1 和 fillcolor＝color2
turtle. filling()	返回当前是否处于填充状态
turtle. begin_fill()	准备开始填充图形
turtle. end_fill()	填充完成
turtle. hideturtle()	隐藏箭头的显示
turtle. showturtle()	显示箭头

1.3.2 运行项目

使用如图 1-14 所示的代码，可以完成太阳花的绘制，样式如图 1-15 所示。

```
In [1]: import turtle
        turtle.color("red", "yellow")
        turtle.speed(10)
        turtle.begin_fill()
        for i in range(50):
            turtle.forward(200)
            turtle.left(170)
        turtle.end_fill()
```

图 1-14 相关 Python 代码

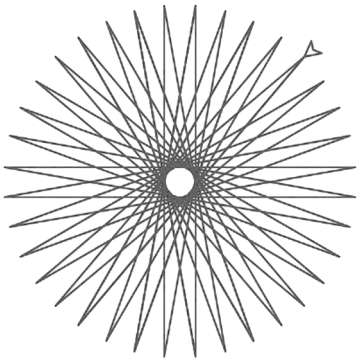

图 1-15 绘图效果

第 2 章　Python 基础

2.1 变量、字符串与数字

2.1.1 变量

1 理解 Python 中的变量

在 Python 中，变量严格意义上应该称为“名字”，也可以理解为标签。如果将值“学会 Python 还可以飞”赋给 str，那么 str 就是变量。在大多数编程语言中，人们都把这一过程称为“把值存储在变量中”，意思是存储在计算机内存中的某个位置。字符串“学会 Python 还可以飞”已经存在，用户不需要准确地知道它到底在哪里，只要告诉 Python 这个字符串的名字是 str，就可以通过这个名字来引用该字符串。这个过程就像快递员取快递一样，内存像一个巨大的货物架，在 Python 中定义变量就如同给快递盒贴标签，如图 2-1 所示。

图 2-1　货物架中贴着标签的快递

快递存放在货物架上，上面贴着写有客户名字的标签。当客户来取快递时，并不需要知道它们在货物架上的具体位置，只需要提供自己的名字，快递员就会把快递交给客户。变量也一样，用户只需要记住存储数据时所用的名字，再调用这个名字就可以了。

2 变量的定义与使用

在 Python 中，不需要先声明变量名及其类型，直接赋值即可创建各种类型的变量。需要注意的是，变量的命名并不是任意的，应遵循以下规则。

（1）变量名必须是一个有效的标识符。

（2）变量名不能使用 Python 中的关键字。

（3）慎用小写字母 l 和大写字母 O。

（4）应选择有意义的单词作为变量名。

为变量赋值可以通过等号（=）来实现。其语法如下。

```
变量名=value
```

例如，创建一个整型变量，并为其赋值 1024，可以使用下面的语句。

```
number=1024                    #创建变量 number 并赋值为 1024
```

这样创建的变量是数值型变量。如果直接为变量赋值一个字符串，那么该变量是字符串类型的。例如，下面的语句：

```
nickname="碧海苍梧"          #字符串类型的变量
```

另外，Python 是一门动态类型的语言。也就是说，变量的类型可以随时变化。例如，在 IDLE 中创建变量 nickname，并赋值为字符串“碧海苍梧”，然后输出该变量的类型，可以看到该变量为字符串类型；也可以将该变量赋值为数值 1024，并输出该变量的类型，可以看到该变量为整型。执行过程如下。

```
>>> nickname="碧海苍梧"                #字符串类型的变量
>>> print(type(nickname))
<class 'str'>
>>> nickname=1024                      #整型变量
>>> print(type(nickname))
<class 'int'>
```

说明：在 Python 中，使用内置函数 type()可以返回变量的类型。

在 Python 中，允许多个变量指向同一个值。例如，将两个变量都赋值为数值 2048，再分别使用内置函数 id()获取变量的内存地址，将得到相同的结果。执行过程如下。

```
>>> no=number=2048
>>> id(no)
49364880
>>> id(number)
49364880
```

说明：在 Python 中，使用内置函数 id()可以返回变量指向的内存地址。

提 示

常量是程序运行过程中，值不能改变的量，如身份证号、π 值等。这些都是不会发生改变的，它们都可以定义为常量。在 Python 中，并没有提供定义常量的关键字。不过，在 PEP 8 规范中规定了常量由大写字母和下画线组成，但是在实际项目中，常量首次赋值后，还是可以被其他代码修改的。

2.1.2 字符串

字符串就是连续的字符序列，可以是计算机所能表示的一切字符的集合。在 Python 中，字符串属于不可变序列，通常使用单引号“' '”、双引号“" "”、三引号“''' '''”或“""" """”括

起来。这 3 种引号在语义上没有区别，只是在形式上有些差别。其中，单引号和双引号括起来的字符序列必须分布在一行，而三引号括起来的字符序列可以分布在连续的多行。例如，定义 3 个字符串类型的变量，并且使用 print()函数输出，代码如下。

```
title='我喜欢的名言警句'          #使用单引号,字符串内容必须在一行
#使用双引号,字符串内容必须在一行
mot_cn="命运给予我们的不是失望之酒,而是机会之杯。"
#使用三引号,字符串内容可以分布在连续的多行
mot_en='''Our destiny offers not the cup of despair,
but the chance of opportunity.'''
print(title)
print(mot_cn)
print(mot_en)
```

执行结果如图 2-2 所示。

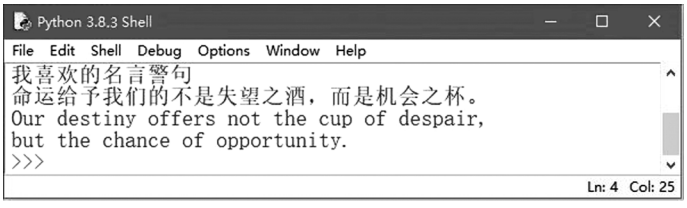

图 2-2 使用 3 种形式定义字符串

需要注意的是，字符串开始和结尾使用的引号形式必须一致。另外，当需要表示复杂的字符串时，还可以嵌套使用引号。例如，下面的字符串都是合法的。

```
'在 Python 中,也可以使用双引号定义字符串'
"'(・・)nnn'也是字符串"
"""'___'"_"* * * """
```

实例 01 输出 007 号坦克

在 IDLE 中创建一个名为 tank.py 的文件，然后在该文件中输出一个表示字符画的字符串。因为该字符画有多行，所以需要使用三引号作为字符串的定界符。具体代码如图 2-3 所示。

```
print('''
                                学编程，你不是一个人在战斗~~
                                |
                             __\--__|_
II=======00000[/ ★007___|
                 _____\_____|/-----.
               /___mingrisoft.com___|
                \◎◎◎◎◎◎◎◎◎/
                  ~~~~~~~~~~~~~~~~
''')
```

图 2-3 代码演示

执行结果如图 2-4 所示。

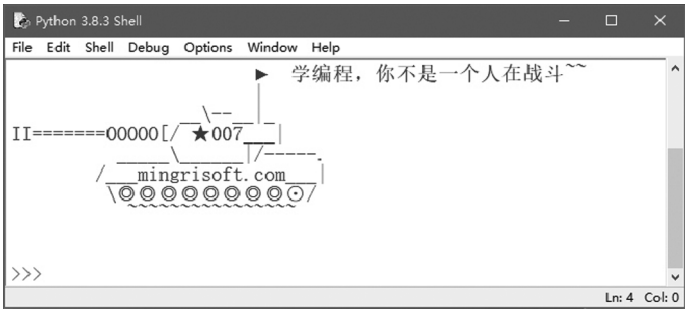

图 2-4　输出 007 号坦克

Python 中的字符串还支持转义字符。所谓转义字符，就是使用反斜线“\”对一些特殊字符进行转义。常用的转义字符如表 2-1 所示。

表 2-1　常用的转义字符

转义字符	说　明
\	续行符
\n	换行符
\o	空
\t	水平制表符，用于横向跳到下一个制表位
\"	双引号
\'	单引号
\	一个反斜线
\f	换页
\0dd	八进制数，dd 代表字符，如“\012”代表换行
\xhh	十六进制数，hh 代表字符，如“\x0a”代表换行

提 示

在字符串定界符（引号）的前面加上字母 r（或 R），字符串将原样输出，其中的转义字符将不进行转义。例如，输出字符串“"失望之酒\x0a 机会之杯"”将输出转义字符换行，而输出字符串“r"失望之酒\x0a 机会之杯"”会原样输出，执行结果如图 2-5 所示。

```
>>> print("失望之酒\x0a机会之杯")
失望之酒
机会之杯
>>> print(r"失望之酒\x0a机会之杯")
失望之酒\x0a机会之杯
>>>
```

图 2-5　转义和原样输出的对比

2.1.3 数字

在生活中，人们经常使用数字记录比赛得分、公司销售数据和网站访问量等信息。Python 提供了数字类型，用于保存这些数值，并且它们是不可改变的数据类型。如果需要修改数字类型变量的值，那么会先把该值存放到内存中，然后修改变量，让其指向新的内存地址。

在 Python 中，数字类型主要包括整数、浮点数和复数等。

1 整数

整数用来表示整数数值，即没有小数部分的数值。在 Python 中，整数包括正整数、负整数和 0，并且它的位数是任意的（当超过计算机自身的计算功能时，会自动转用高精度计算）。如果要指定一个非常大的整数，只需要写出其所有位数即可。

整数类型包括十进制整数、八进制整数、十六进制整数和二进制整数等。

（1）十进制整数：由 0～9 组成，进位规则为“逢十进一”。例如，下面的数值都是有效的十进制整数。

```
3141592653589793238462 6
6666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666666
-2018
0
```

在 IDLE 中的执行结果如图 2-6 所示。

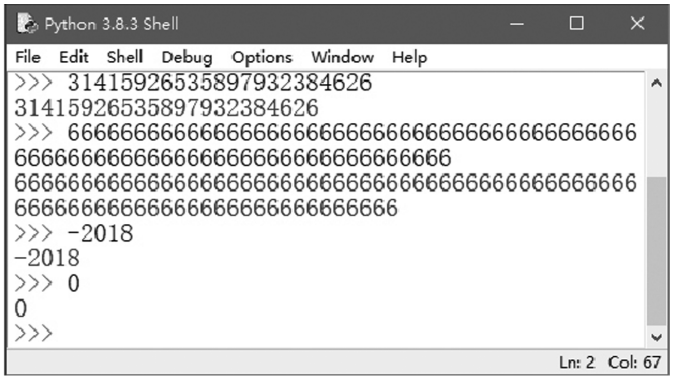

图 2-6 有效的十进制整数

说明：在 Python 2.x 中，如果输入的数比较大，Python 会自动在其后面加上字母 L（也可能是小写字母 l）。例如，在 Python 2.7.14 中，输入“3141592653589793238462 6”后，其输出结果如图 2-7 所示。

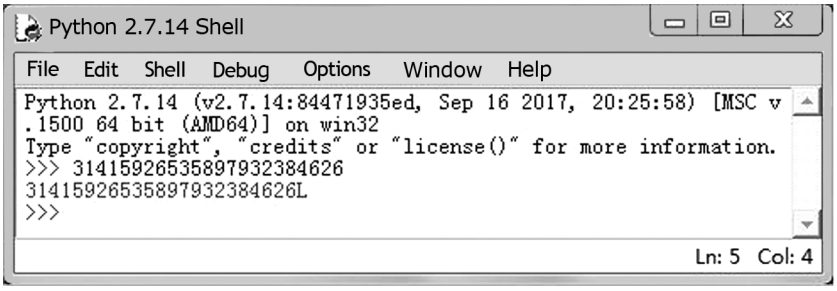

图 2-7　在 Python 2. x 中输入较大数时的输出效果

提 示

不能以 0 作为十进制整数的开头（0 除外）。

（2）八进制整数：由 0～7 组成，进位规则为“逢八进一”，日常书写时一般以 0o/0O 开头，如 0o123（转换成十进制整数为 83）、－0o123（转换成十进制整数为－83）。

提 示

在 Python 3. x 中，八进制整数必须以 0o/0O 开头。但在 Python 2. x 中，八进制整数可以以 0 开头。

（3）十六进制整数：由 0～9、A～F/a～f 组成，进位规则为“逢十六进一”，日常书写时一般以 0x/0X 开头，如 0x25（转换成十进制整数为 37）、0Xb1e（转换成十进制整数为 2846）。

提 示

在 Python 3. x 中，十六进制整数必须以 0x 或 0X 开头。

（4）二进制整数：由 0 和 1 两个数组成，进位规则为“逢二进一”，如 101（转换成十进制整数为 5）、1010（转换成十进制整数为 10）。

2 浮点数

浮点数由整数部分和小数部分组成，主要用于处理包含小数的数，如 1. 414、0. 5、－1. 732、3. 141592653589793238462 6 等。浮点数也可以使用科学记数法表示，如 2. 7e2、－3. 14e5 和 6. 16e－2 等。

需要注意的是，使用浮点数进行计算时，可能会出现小数位数不确定的情况。例如，计算 0. 1 ＋ 0. 1 时，将得到想要的结果 0. 2，而计算 0. 1 ＋ 0. 2 时，将得到 0. 30000000000000004（想要的结果为 0. 3），执行过程如下。

```
>>> 0.1+0.1
0.2
>>> 0.1+0.2
0.30000000000000004
```

所有的编程语言都存在这个问题，暂时忽略多余的小数位数即可。

实例 02　根据身高、体重计算 BMI 指数

在 IDLE 中创建一个名为 bmiexponent.py 的文件，并在该文件中定义两个变量：一个用于记录身高（单位为米），另一个用于记录体重（单位为千克），然后根据公式“BMI＝体重/(身高×身高)”计算 BMI 指数。其代码如下。

```
height=1.70      #保存身高的变量,单位为米
print("您的身高:" + str(height))
weight=48.5      #保存体重的变量,单位为千克
print("您的体重:" + str(weight))
bmi=weight/(height*height)      #用于计算 BMI 指数,公式:BMI=体重/身高的平方
print("您的 BMI 指数为:"+str(bmi))      #输出 BMI 指数
#判断身材是否合理
if bmi<18.5:
    print("您的体重过轻 ~@_@~")
if bmi>=18.5 and bmi<24.9:
    print("正常范围,注意保持(-_-)")
if bmi>=24.9 and bmi<29.9:
    print("您的体重过重 ~@_@~")
if bmi>=29.9:
    print("肥胖 ^@_@^")
```

说明：str()函数用于将数值转换为字符串，if 语句用于进行条件判断。如果需要了解更多关于函数和条件判断的知识，请阅读后面的章节。

执行结果如图 2-8 所示。

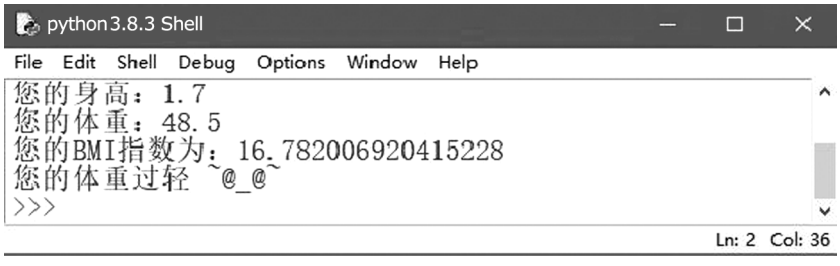

图 2-8　根据身高、体重计算 BMI 指数

3 复数

Python 中的复数与数学中复数的形式完全一致，都由实部和虚部组成，并且使用 j 或 J 表示虚部。当表示一个复数时，可以将其实部和虚部相加。例如，一个复数的实部为 3.14，虚部为 12.5j，则这个复数为 3.14＋12.5j。

2.2 注释

2.2.1 注释的含义

在程序中，注释就是对代码的解释和说明，可以让他人了解代码实现的功能，从而帮助他人更好地阅读代码。注释的内容会被 Python 解释器忽略，并不会在执行结果中体现出来。

2.2.2 注释的 3 种类型

在 Python 中，通常包括 3 种类型的注释，分别是单行注释、多行注释和中文编码声明注释。这些注释在 IDLE 中的效果如图 2-9 所示。

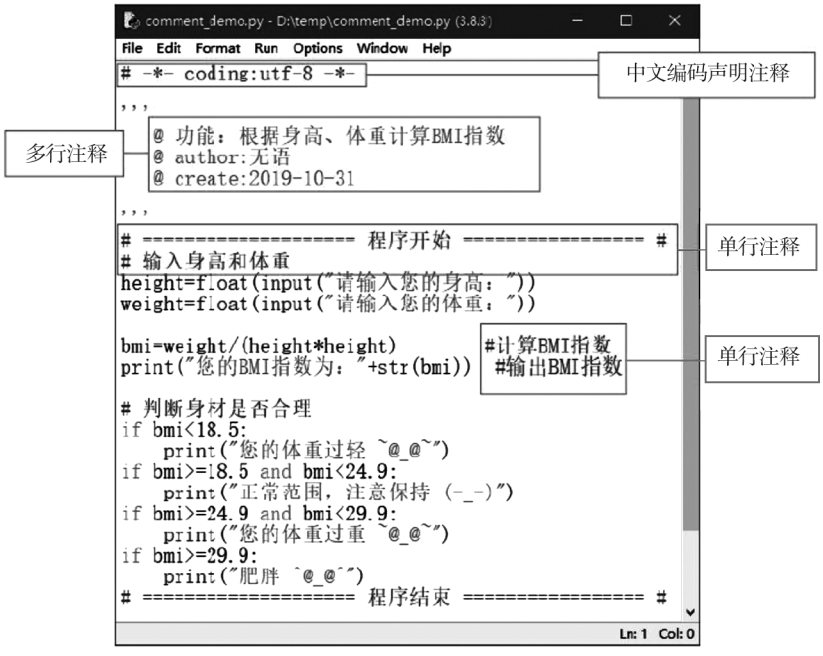

图 2-9 Python 中的注释

1 单行注释

在 Python 中，使用“#”作为单行注释的符号。从符号“#”开始，直到换行为止，“#”后面所有的内容都是注释的内容，并被 Python 解释器忽略。

其语法如下。

```
#注释内容
```

单行注释可以放在要注释代码的前一行，也可以放在要注释代码的右侧。例如，下面两种注释形式都是正确的。

第一种形式：

```
#要求输入身高,单位为米,如 1.70
height=float(input("请输入您的身高:"))
```

第二种形式：

```
height=float(input("请输入您的身高:"))    #要求输入身高,单位为米,如 1.70
```

上面两种形式的执行结果是相同的，如图 2-10 所示。

```
请输入您的身高：1.70
>>>
```

图 2-10 执行结果

说明：添加注释时，一定要有意义，即注释能充分解释代码的功能及用途。例如，图 2-11所示的注释就是冗余的注释。如果将其修改为如图 2-12 所示的注释，就能清楚地说明代码的用途了。

```
bmi=weight/(height*height)        #Magic，请勿改动
```

图 2-11 冗余的注释

```
bmi=weight/(height*height)        # 用于计算BMI指数，公式为“体重/身高的平方”
```

图 2-12 推荐的注释

需要注意的是，注释可以出现在代码的任意位置，但是不能分隔关键字和标识符。例如，下面的注释是错误的。

```
height=float(#要求输入身高 input("请输入您的身高:"))
```

多学两招：注释除了可以解释代码的功能及用途之外，还可以用于临时注释掉不想执行的代码。在 IDLE 中，通过选择主菜单中的 Format→Comment Out Region 命令（快捷键为<Alt+3>），可以将选中的代码注释掉；通过选择主菜单中的 Format→UnComment Region 命令（快捷键为<Alt+4>），可以取消注释掉的代码。

2 多行注释

在 Python 中，并没有一个单独的多行注释标记，而是将包含在一对三引号（'''……'''或"""……"""）之间，并且不属于任何语句的内容都视为注释，这样的代码将被 Python 解释器忽略。因为这样的注释是分为多行编写的，所以称为多行注释。

其语法如下。

```
'''
注释内容 1
注释内容 2
……
'''
```

或者

```
"""
注释内容 1
注释内容 2
……
"""
```

多行注释通常用来为 Python 文件、模块、类或者函数等添加版权、功能等信息。例如，下面的代码将使用多行注释为 demo.py 文件添加版权、功能及修改日志等信息。

```
'''
@版权所有:吉林省明日科技有限公司版权所有
@文件名:demo.py
@文件功能描述:根据身高、体重计算 BMI 指数
@创建日期:2017 年 10 月 31 日
@创建人:无语
@修改标识:2017 年 11 月 2 日
@修改标识:2017 年 11 月 2 日
@修改描述:增加根据 BMI 指数判断身材是否合理功能代码
@修改日期:2017 年 11 月 2 日
'''
```

提 示

在 Python 中，三引号是字符串定界符。如果三引号作为语句的一部分出现，就不是注释，而是字符串，这一点要注意区分。例如，图 2-13 所示代码为多行注释，图 2-14 所示代码为字符串。

```
'''
    @ 功能：根据身高、体重计算BMI指数
    @ author:无语
    @ create:2017-10-31
'''
```

图 2-13　三引号内的内容为多行注释

```
print('''根据身高、体重计算BMI指数''')
```

图 2-14　三引号内的内容为字符串

3 中文编码声明注释

在 Python 3.x 中，还提供了一种特殊的注释，即中文编码声明注释。该注释的出现主要是为了解决 Python 2.x 不支持直接写中文的问题。虽然在 Python 3.x 中，该问题已经不存在了，但是为了规范页面的编码，以及方便其他程序员及时了解文件所用的编码，建议在文件开头加上中文编码声明注释。

其语法如下。

```
#-*- coding:编码 -*-
```

或者

```
# coding=编码
```

在上面的语法中，编码为文件所使用的字符编码类型。如果采用 UTF-8 编码，则设置为 UTF-8；如果采用 GBK 编码，则设置为 GBK 或 CP936。

例如，指定编码为 UTF-8，可以使用下面的中文编码声明注释。

```
#-*- coding:UTF-8 -*-
```

说明：在以上代码中，“-*-”没有特殊的作用，只是为了美观才加上的。所以，以上代码也可以使用“#coding:UTF-8”代替。

另外，下面也是正确的中文编码声明注释。

```
# coding=UTF-8
```

2.3 运算符

运算符是一些特殊的符号，主要用于数学计算、比较大小和逻辑运算等。Python 的运算符主要包括算术运算符、赋值运算符、比较运算符、逻辑运算符和位运算符等。使用运算符将不同类型的数据按照一定的规则连接起来的式子称为表达式。例如，使用算术运算符连接起来的式子称为算术表达式，使用逻辑运算符连接起来的式子称为逻辑表达式。下面介绍常用的运算符。

2.3.1 算术运算符

算术运算符是处理四则运算的符号，在数字的处理中应用得最多。常用的算术运算符如表 2-2 所示。

表 2-2 常用的算术运算符

运算符	说 明	实 例	结 果
+	加	12.45+15	27.45
-	减	4.56-0.26	4.3
*	乘	5 * 3.6	18.0
/	除	7/2	3.5
%	求余，即返回除法的余数	7%2	1
//	取整除，即返回商的整数部分	7//2	3
* *	幂，即返回 x 的 y 次方	2 * * 4	16，即 2^4

说明：使用求余运算符时，如果除数（第二个操作数）是负数，那么取得的结果也是一个负数。

提 示

使用除法（/或//）运算符和求余运算符时，除数不能为 0，否则将会出现异常，如图 2-15 所示。

```
>>> 5//0
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    5//0
ZeroDivisionError: integer division or modulo by zero
>>> 5/0
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    5/0
ZeroDivisionError: division by zero
>>> 5%0
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    5%0
ZeroDivisionError: integer division or modulo by zero
```

图 2-15 除数为 0 时出现的错误提示

实例 03　计算学生成绩的分数差及平均分

某学生 3 门课程的成绩如表 2-3 所示。

表 2-3　3 门课程的成绩

课　程	分　数
Python	95
English	92
C 语言	89

编程实现以下功能。

（1）计算 Python 和 C 语言课程的分数之差。

（2）计算 3 门课程的平均分。

在 IDLE 中创建一个名为 score_handle.py 的文件，首先在该文件中定义 3 个变量，用于存储各门课程的分数，然后使用减法运算符计算分数差，再使用加法运算符和除法运算符计算平均成绩，最后输出计算结果。其代码如下。

```
01 python=95                    #定义变量,存储 Python 课程的分数
02 english=92                   #定义变量,存储 English 课程的分数
03 c=89                         #定义变量,存储 C 语言课程的分数
04 sub=python - c               #计算 Python 和 C 语言课程的分数差
05 avg=(python + english + c) / 3          #计算平均分
06 print("Python 课程和 C 语言课程的分数之差：" + str(sub) + " 分\n")
07 print("3 门课的平均分：" + str(avg) + "分")
```

执行结果如图 2-16 所示。

图 2-16　计算学生成绩的分数差及平均分

说明：在 Python 2.x 中，除法运算符（/）的执行结果与 Python 3.x 的不一样。在 Python 2.x 中，如果操作数为整数，则结果将截取为整数；而在 Python 3.x 中，计算结果为浮点数。例如，7/2 在 Python 2.x 中的结果为 3，而在 Python 3.x 中的结果为 3.5。

2.3.2　比较运算符

比较运算符又称关系运算符，用于对变量或表达式的结果进行比较。如果比较结果为“真”，则返回 True；如果比较结果为“假”，则返回 False。比较运算符通常用在条件语句中作为判断的依据。Python 中的比较运算符如表 2-4 所示。

表 2-4 Python 中的比较运算符

运算符	说 明	实 例	结 果
>	大于	'a'>'b'	False
<	小于	156<456	True
==	等于	'c'=='c'	True
!=	不等于	'y'!='t'	True
>=	大于或等于	479>=426	True
<=	小于或等于	62.45<=45.5	False

在 Python 中，当需要判断一个变量是否介于两个值之间时，可以采用“值 1 < 变量 < 值 2”的形式，如“0 < a < 100”。

实例 04 使用比较运算符比较大小关系

在 IDLE 中创建一个名为 comparison_operator.py 的文件，然后在该文件中定义 3 个变量，并分别使用 Python 中的各种比较运算符对它们的大小进行比较。其代码如下。

```
01 python=95        #定义变量,存储 Python 课程的分数
02 english=92       #定义变量,存储 English 课程的分数
03 c=89             #定义变量,存储 C 语言课程的分数
04 #输出 3 个变量的值
05 print("python=" + str(python) + " english=" +str(english) + " c=" +str(c) + "\n")
06 print("python < english 的结果:" + str(python < english))          #小于操作
07 print("python>english 的结果:" + str(python>english))             #大于操作
08 print("python == english 的结果:" + str(python == english))       #等于操作
09 print("python != english 的结果:" + str(python != english))       #不等于操作
10 print("python <= english 的结果:" + str(python <= english))       #小于或等于操作
11 print("english >= c 的结果:" + str(python >= c))                  #大于或等于操作
```

执行结果如图 2-17 所示。

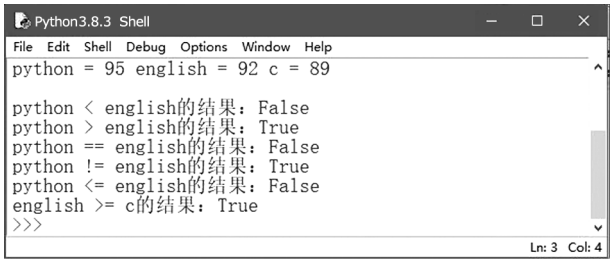

图 2-17 使用比较运算符比较大小

2.3.3 赋值运算符

赋值运算符主要用来为变量赋值。使用时，可以直接把基本赋值运算符“=”右边的值赋给左边的变量，也可以进行某些运算后再赋值给左边的变量。在 Python 中，常用的

赋值运算符如表 2-5 所示。

表 2-5　常用的赋值运算符

运算符	说　明	实　例	结　果
=	简单的赋值运算	x=y	x=y
+=	加赋值	x+=y	x=x+y
-=	减赋值	x-=y	x=x-y
**=	幂赋值	x**=y	x=x**y
//=	取整除赋值	x//=y	x=x//y

提　示

混淆=和==是编程中常见的错误之一。很多编程语言（不只是 Python）都使用了这两个符号，而且很多程序员经常会用错这两个符号。

2.3.4　逻辑运算符

某手机店在每周二的 10 点至 11 点和每周五的 14 点至 15 点，对华为 Mate 10 系列手机进行打折让利活动，那么想参加打折活动的顾客，就要在时间上满足两个条件：周二 10:00～11:00 或周五 14:00～15:00。这里用到了逻辑关系，Python 提供了逻辑运算符来进行逻辑运算。

逻辑运算符是对“真”和“假”两个布尔值进行运算，运算后的结果仍然是一个布尔值。Python 中的逻辑运算符主要包括 and（逻辑与）、or（逻辑或）、not（逻辑非）。表 2-6 所示为逻辑运算符的用法和说明。

表 2-6　逻辑运算符的用法和说明

运算符	含　义	用　法	结合方向
and	逻辑与	op1 and op2	从左到右
or	逻辑或	op1 and op2	从左到右
not	逻辑非	not op	从右到左

使用逻辑运算符进行逻辑运算时，其运算结果如表 2-7 所示。

表 2-7　使用逻辑运算符进行逻辑运算的结果

表达式 1	表达式 2	表达式 1 and 表达式 2	表达式 1 or 表达式 2	not 表达式 1
True	True	True	True	False
True	False	False	True	False
False	False	False	False	True
False	True	False	True	True

实例 05　参加手机店的打折活动

在 IDLE 中创建一个名为 sale.py 的文件，然后在该文件中使用代码实现本小节开头所描述的场景，代码如下。

```
01 print("\n手机店正在打折,活动进行中……")          #输出提示信息
02 strWeek=input("请输入中文星期(如星期一):")       #输入星期,如星期一
03 intTime=int(input("请输入时间中的小时(范围:0～23):"))      #输入时间
04 #判断是否满足活动参与条件(使用了 if 条件语句)
05 if (strWeek == "星期二" and (intTime >= 10 and intTime <= 11)) or (strWeek ==
"星期五" and (intTime >= 14 and intTime <= 15)):
06   print("恭喜您,获得了折扣活动参与资格,快快选购吧!")          #输出提示信息
07 else:
08   print("对不起,您来晚一步,期待下次活动……")                  #输出提示信息
```

代码注解：

（1）第 2 行代码中，input()函数用于接收用户输入的字符序列。

（2）第 3 行代码中，因为 input()函数返回的结果为字符串类型，所以需要进行类型转换。

（3）第 5 行和第 7 行代码使用了 if…else 语句，该语句主要用来判断程序是否满足某种条件。该语句将在第 3 章中详细讲解，这里只需要了解即可。

（4）第 5 行代码中对条件进行判断时，使用了逻辑运算符 and、or 和比较运算符==、>=、<=。

按 F5 键运行程序，首先输入星期为“星期五”，然后输入时间为 19，将显示如图 2-18 所示的结果；再次运行程序，输入星期为“星期二”，时间为 10，将显示如图 2-19 所示的结果。

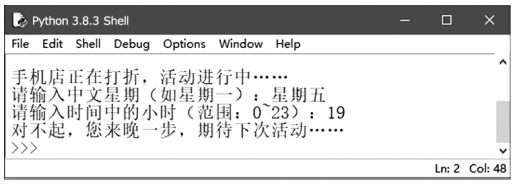

图 2-18　不符合条件的运行效果

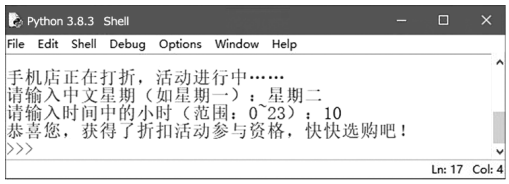

图 2-19　符合条件的运行效果

2.3.5　位运算符

位运算符是把数字当作二进制数来计算的，因此，需要先将要计算的数据转换为二进制形式，然后才能计算。Python 中的位运算符有位与（&）、位或（|）、位异或（^）、位取反（～）、左移位（<<）和右移位（>>）运算符。

说明：整型数据在内存中以二进制形式表示。例如，7 的 32 位二进制形式为

00000000 00000000 00000000 00000111

其中，左边最高位是符号位，最高位为 0 表示正数，最高位为 1 表示负数。负数采用补码表示。例如，－7 的 32 位二进制形式为

11111111 11111111 11111111 11111001

1 位与运算

位与运算符为“&”，位与运算的运算法则是：两个操作数的二进制表示，只有对应数位都是 1 时，结果数位才是 1，否则为 0。如果两个操作数的精度不同，则结果的精度与精度高的操作数相同，如图 2-20 所示。

```
  0000 0000 0000 1100
& 0000 0000 0000 1000
-------------------
  0000 0000 0000 1000
```

图 2-20　12&8 的运算过程

2 位或运算

位或运算符为“|”，位或运算的运算法则是：两个操作数的二进制表示，只有对应数位都是 0 时，结果数位才是 0，否则为 1。如果两个操作数的精度不同，则结果的精度与精度高的操作数相同，如图 2-21 所示。

```
  0000 0000 0000 0100
| 0000 0000 0000 1000
-------------------
  0000 0000 0000 1100
```

图 2-21　4 | 8 的运算过程

3 位异或运算

位异或运算符为“^”，位异或运算的运算法则是：当两个操作数的二进制表示相同（同时为 0 或同时为 1）时，结果为 0，否则为 1。若两个操作数的精度不同，则结果的精度与精度高的操作数相同，如图 2-22 所示。

```
  0000 0000 0001 1111
^ 0000 0000 0001 0110
-------------------
  0000 0000 0000 1001
```

图 2-22　31^22 的运算过程

4 位取反运算

位取反运算也称位非运算，运算符为“~”。位取反运算就是将操作数中的二进制数 1 修改为 0，0 修改为 1，如图 2-23 所示。

```
~ 0000 0000 0111 1011
-------------------
  1111 1111 1000 0100
```

图 2-23　~123 的运算过程

在 Python 中，使用 print()函数输出以上运算的结果，代码如下。

```
print("12&8="+str(12&8))      #位与计算整数的结果
print("4|8="+str(4|8))        #位或计算整数的结果
print("31^22="+str(31^22))    #位异或计算整数的结果
print("~123="+str(~123))      #位取反计算整数的结果
```

执行结果如图 2-24 所示。

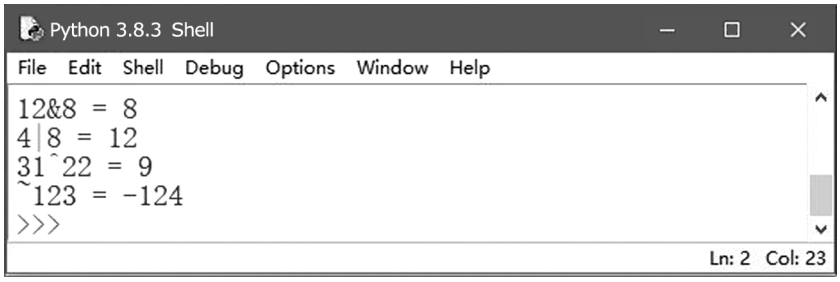

图 2-24 执行结果

5 左移位运算

左移位运算符“<<”是将一个二进制操作数向左移动指定的位数，左边（高位端）溢出的位被丢弃，右边（低位端）的空位用 0 补充。左移位运算相当于乘以 2 的 n 次幂。

例如，int 类型数据 48 对应的二进制数为 00110000，将其左移 1 位，根据左移位运算的运算规则，可以得出(00110000<<1)=01100000，转换为十进制数就是 96（48×2^1）；将其左移 2 位，根据左移位运算的运算规则，可以得出(00110000<<2)=11000000，转换为十进制数就是 192（48×2^2）。其运算过程如图 2-25 所示。

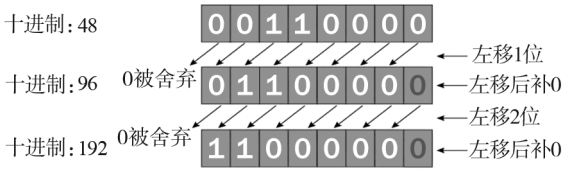

图 2-25 左移位运算

6 右移位运算

右移位运算符“>>”是将一个二进制操作数向右移动指定的位数，右边（低位端）溢出的位被丢弃，而在填充左边（高位端）的空位时，如果最高位是 0（正数），左侧空位填入 0，如果最高位是 1（负数），左侧空位填入 1。右移位运算相当于除以 2 的 n 次幂。

48 右移 1 位的运算过程如图 2-26 所示。

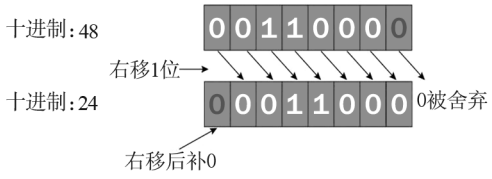

图 2-26 48 右移 1 位的运算过程

−80 右移 2 位的运算过程如图 2-27 所示。

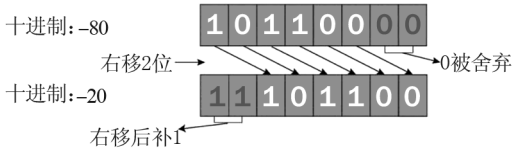

图 2-27 −80 右移 2 位的运算过程

2.4 成员运算符

成员运算符用于判定对象是否存在于字符串中，其功能和应用示例如表 2-8 所示。

表 2-8 成员运算符

运算符	描 述	示 例
in	若对象包含在字符串中，则返回 True，否则返回 False	“ * y * in "Python"” 返回 True
in not	若对象未包含在字符串中，则返回 True，否则返回 False	“"x" in not "Python"” 返回 True

2.5 身份运算符

身份运算符用于比较两个对象的内存地址是否相同，其功能和应用示例如表 2-9 所示。

表 2-9 身份运算符

运算符	描 述	示 例
is	若运算符两侧的变量指向同一个对象，则返回 True，否则返回 False	若 x＝1，y＝x，则“x is y”返回 True
is not	若运算符两侧的变量指向不同的对象，则返回 True，否则返回 False	若 x＝1，y＝2，则“x is not y”返回 True

2.6 运算符的优先级

所谓运算符的优先级，是指在运算中哪一个运算符先计算，哪一个后计算，与数学中四则运算遵循的“先乘除，后加减”是一个道理。

在 Python 中，优先级高的运算符先执行，优先级低的运算符后执行，同一优先级的运算符按照从左到右的顺序执行。用户也可以像四则运算那样使用括号来改变运算次序，括号内的运算符最先执行。表 2-10 所示为运算符的优先级。同一行中的运算符具有相同的优先级，它们的结合方向决定求值顺序。

表 2-10 运算符的优先级

运算符	说 明	优先级
* *	幂	高 ↓ 低
~、+、-	取反、正号和负号	
*、/、%、//	算术运算符	
+、-	算术运算符	
<<、>>	位运算符中的左移和右移	
&	位运算符中的位与	
^	位运算符中的位异或	
|	位运算符中的位或	
<、<=、>、>=、!=、==	比较运算符	

多学两招：在编写程序时尽量使用括号“()”来限定运算次序，避免运算次序错误。