第1章 概 述

计算机的基本工作模式是运行预先存放在计算机内部的程序来控制计算机中各个部件协同工作，完成用户期望的任务。这种将预定任务用程序表现出来的过程称为程序设计。本章介绍计算机硬件的基本组成、实现程序设计的计算机语言的发展，以及 Python 语言的编程环境。

1.1 计算机基础知识

世界上第一台通用电子计算机是 1946 年由美国宾夕法尼亚大学莫尔电气工程学院研制成功的 ENIAC(electronic numerical integrator and computer)。当时的计算机主要使用穿孔卡片和穿孔带存储程序，这种机械式的输入远远跟不上电子运算的节奏，同时编程是一件非常复杂的难事。

在 ENIAC 的总体设计已经完成并进入硬件实现阶段时，冯·诺依曼加入了 ENIAC 团队，与团队的另外两位主要负责人——普雷斯伯·埃克特和约翰·莫奇利一起讨论对 ENIAC 的改进，而这三位负责人早在 1944 年就开始研制一款名为 EDVAC(electronic discrete variable automatic computer)的电子离散变量自动计算机。1945 年 6 月由冯·诺依曼起草书写了长达 101 页，影响现代计算机历史走向的《EDVAC 报告书的第一份草案》。该草案不仅详述了 EDVAC 的设计，还为现代计算机的发展指明了道路。其要点如下。

(1) 计算机中的指令和数据均以二进制形式存储，指令由操作码和地址码组成。

(2) 像存储数据一样存储程序。

(3) 指令的执行是顺序的，即一般按照指令在存储器中存放的顺序执行，程序分支由转移指令实现。

(4) 计算机由运算器、控制器、存储器、输入模块和输出模块五部分组成。

运用“存储程序”和“程序控制”相结合原理，将程序和数据存放在内存中，在程序的控制下自动完成操作。

这种结构一直延续至今，所以现在一般计算机被称为冯·诺依曼结构计算机，冯·诺依曼也被誉为“现代电子计算机之父”。

1.1.1 计算机组成

计算机作为 20 世纪最伟大的发明之一，其结构和工作原理相比传统的计算工具(如算盘、计算尺等)要复杂得多。一个完整的计算机系统可以分为硬件系统和软件系统两大模块。计算机硬件系统就是被称为冯·诺依曼结构的五大物理设备组件；而计算机软件系统通常包含系统软件和应用软件，是能完成一定功能的各种算法、数据的集合。计算机系统组成如图 1-1 所示。

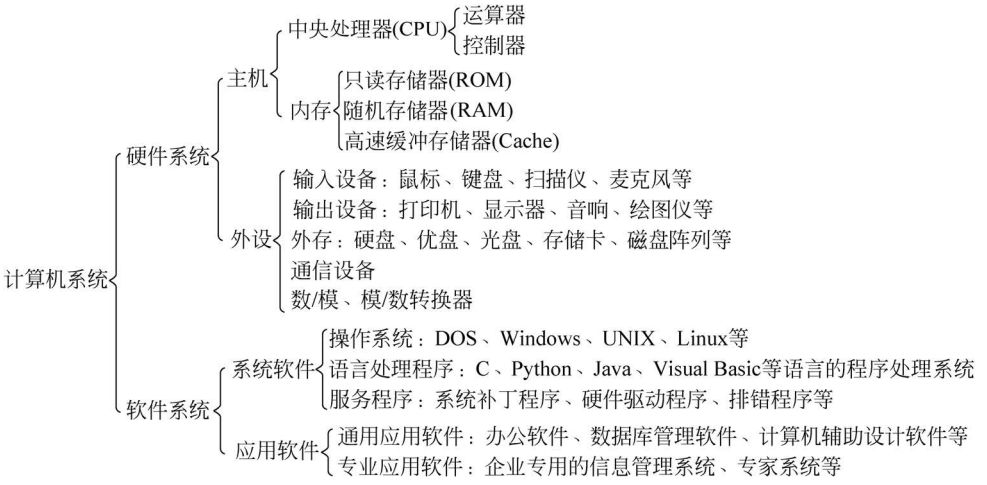

图 1-1　计算机系统组成

1.1.2　计算机语言

英文单词 computer 后缀 er 通常是指人，如 teacher、leader 等，computer 最早是指专门负责计算的人，后来演变为用于计算的设备，也就是计算机。

计算机是能够根据一组指令操作数据的机器，它具有两个特性：功能性和可编程性。功能性是指计算机可以进行数据计算，可编程性是指计算机可以根据一组指令来执行操作。无论哪种特性，都需要给出能够完成这些特性的所有细节，描述计算机执行操作的所有动作和执行顺序。为了描述这些动作，需要一个系统的描述方式，这就形成了计算机语言。

计算机语言是人与计算机之间交换信息的工具，是用于指挥或控制计算机工作的“符号系统”。为使计算机能按照人的意图工作，能够接受人向它发出的命令和信息就必须使用计算机语言，把待解决的问题按处理步骤写成一条条计算机能够识别和执行的语句。所有语句的集合称为程序。因而计算机语言也被称为程序设计语言或编程语言。

人与人之间的交流采用人类自然语言，不同的国家使用不同的语言，如汉语、英语、法语等；人与计算机之间的交流采用计算机语言，同样，计算机语言也有很多类型，如 Python 语言、C 语言、Java 语言等。

无论从语言设计还是语言使用上，计算机语言与人类自然语言有着极其相似的语言模型，图 1-2 所示为计算机语言模型图。

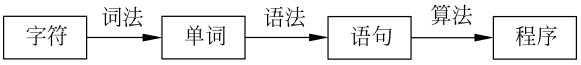

图 1-2　计算机语言模型图

提示：程序设计语言跟人类自然语言一样，都需要经历字符、单词、造句、写作的一系列学习过程，大众意识中的 Python 语言很简单，但同样需要在掌握了词法、语法、算法之后，在应用的时候相比其他程序设计语言才会更简单，前期的基础知识学习对所有语言都是一样的。

1. 机器语言

计算机究其实质是一台电子设备，之所以能完成数据计算、过程控制等功能，是因为它可以将数据和指令转换成相应的电信号，并由物理元器件完成各种信号处理。物理元器件的工作由电信号控制，典型的电信号为高电平和低电平，用数字信号 1 和 0 描述。也就是说计算机能够工作，核心指令就是代表 1 和 0 的高、低电平，在计算机内部被表示为二进制数字形式，这就是机器语言。机器语言也可以被看作控制计算机执行操作的最直接的程序设计语言。

机器语言由 1 和 0 二进制代码按一定规则组成，是能够被计算机直接理解和执行指令的集合。机器语言的每一条指令都是一条二进制形式的指令代码，指令代码中包含操作码和操作数。操作码指出执行何种操作，操作数指定被操作的数值或某数值在内存中的地址。

例如，执行数据计算 1＋1，采用机器语言的程序如下。

```
10111000 00000001 00000000          ;将被加数 1 放入累加器 AX 中
00000101 00000001 00000000          ;将加数 1 与累加器中数据相加,结果仍放入 AX 中
```

不难看出，用这样的机器语言编写程序，其工作量大，难学、难记、难调试，只能由专业人员使用。计算机指令系统不同，机器语言也不同，因此通用性差，其唯一的优势就是这是控制机器工作的最直接指令，执行速度最快，占用存储空间最少。

2. 汇编语言

汇编语言的诞生源于机器语言因难理解、难编程、难记忆而阻碍了计算机行业的发展。汇编语言采用符号和数字代替二进制指令码，每一条指令采用英文助记符，又称为符号语言，将难以记忆的机器指令转换成助记符，方便记忆、理解和调试，极大地提高了工作效率。同样是完成 1＋1 的数据计算，采用汇编语言的程序如下。

```
MOV AX, 1                           ;将被加数 1 放入(move)累加器 AX 中
ADD AX, 1                           ;将加数 1 与累加器中数据相加(add),结果仍放入 AX 中
```

相比于机器语言，汇编语言在一定程度上改善了机器语言存在的难读、难记等问题，同时汇编语言是仅次于机器语言的执行速度最快、占用存储空间最少的语言。但是汇编语言要求用户详细了解所用的计算机硬件性能、指令系统、寻址方式以及其他相关知识，对机器硬件的依赖性很大。汇编程序的通用性、可移植性都较差。

机器语言和汇编语言都被称为低级语言。由于机器语言晦涩难懂，普通程序开发人员一般接触不到；汇编语言虽然可以依赖助记符编写程序，但除了与硬件相关的程序开发人员外，大多数开发者也不会使用这种语言，更多的程序开发者使用高级语言来编写程序。

3. 高级语言

从 1954 年由 IBM 公司推出的第一门高级语言 FORTRAN 至今，已经出现过 COBOL、BASIC、LISP、Pascal、C、C++、FoxPro、Java、Visual Basic、Python、C＃等多种高级语言。高级语言之所以称为高级，是因为这类语言更贴近人类自然语言和数学语言，更方便人的理解和认知。比如上面执行的数学运算，可以直接输入 1＋1 就能完成计算操作。但是计算机实质是一台电子设备，要控制其工作，依然需要 1 和 0 表示的高、低电平控制信号。也就是说，所有的高级语言都需要翻译成机器语言才能被计算机识别并执行，这个翻译过程称为编译(compile)或解释(interpret)。

用高级语言编写的程序被称为源文件(source file)或源代码(source code)。编译是指将源代码一次性翻译成机器语言格式的目标文件(object file)的过程。此时的代码还不能运行起来。因为目标文件还需要和操作系统提供的组件(如标准库)结合起来,而这些组件都是程序运行所必需的。例如,要在屏幕中输出字符,这类操作必须调用系统提供的库才能够实现,这就是连接。经过连接才会生成可执行程序(如 Windows 平台上的.exe 文件),这个可执行文件可以直接被加载运行。编译程序把一个源代码翻译成目标程序的工作过程分为五个阶段:词法分析、语法分析、中间代码生成、代码优化、目标程序生成。虽然过程相对烦琐,但是只要源代码不发生修改,编译过程只需要一次即可。C 语言属于编译型语言。

解释是指在程序运行时对源代码进行逐条语句的翻译并运行。它的执行过程是解释器翻译一句就执行一句,并且不会生成目标程序。因此解释型语言的主要缺点是速度慢,但是解释型语言提供了极佳的调试支持,并且具有高安全性和平台独立性。一般情况下,动态语言都属于解释型语言,Python 语言和 Java 语言都属于解释型语言。

采用高级语言编写程序,程序开发者不需要过多关注计算机硬件,不需要过多了解计算机的指令系统,这样开发者可以将更多的精力集中在解决问题本身上,而不必受机器制约,可以极大地提高编程效率。也正因为高级语言与具体的计算机硬件关系不大,因而所写出来的程序可移植性好,复用率高。当然因为所有的高级语言都要经过编译或解释才能控制计算机工作,其执行速度相对较慢,占用的资源多。

1.2 Python 语言简介

1.2.1 Python 语言的发展史

Python 语言是一种面向对象、解释型的计算机程序设计语言,其起源是圭多·范·罗苏姆(Guido van Rossum)于 1989 年开发的一个新的脚本解释程序。Python 的英文意思是蟒蛇,其实这个语言跟蟒蛇没有什么关系,只是因为开发者 Guido van Rossum 是英国 BBC 电视剧——蒙提·派森的飞行马戏团(Monty Python's Flying Circus)的爱好者。Python 的第一个公开发行版本发行于 1991 年。1999 年,Guido van Rossum 向美国国防部高级研究计划局 DARPA(Defense Advanced Research Projects Agency)提交了名为 Computer Programming For Everybody 的资金申请,并在后面说明了他对 Python 语言的想法。

(1) 一门简单直观的语言并与主要竞争者一样强大。

(2) 开源,以便任何人都可以为它作贡献。

(3) 代码像纯英语那样容易理解。

(4) 适用于短期开发的日常任务。

这些想法现在基本都已经成为现实,Python 已经成为一门流行的程序设计语言。

本书使用的 Python 3.x 最初版本发行于 2008 年 12 月。Python 3.x 增加了对 Unicode 的支持,解决了处理字符编码的问题,但是由于语句与动态连接库之间的变更,破坏了其向后兼容性,导致许多基于 Python 2.x 的程序无法直接在 Python 3.x 的环境中运行。

全国计算机等级考试二级从 2018 年 3 月开始新增科目"Python 语言程序设计",采用的版本是 Python 3.5.2,本书采用更新于 2019 年 12 月的 Python 3.7.6 版本。

1.2.2　Python 语言的特点和应用

在 Python 的世界里有一句很经典的话："人生苦短，我用 Python。"这句看似戏言的话实际上恰恰反映了 Python 的语言特性与其在开发者心里的分量。同样一个问题，采用不同的程序设计语言解决，代码量差距太大了，一般情况下 Python 是 Java 的 1/5。

除了拥有大多数主流编程语言的优点（面向对象、语法丰富）外，Python 的直观特点是简明优雅、易于开发，可以用更少的代码完成更多的工作。尽管 Python 是一种解释型语言，与传统的编译型语言相比机器执行效率较低，但是处理器的处理速度与环境速度（如网络环境）的差异在大多数场景中完全抵消了上述劣势。牺牲部分运行效率带来的好处则是提升了开发效率，在跨平台的时候无须移植和重新编译。所以 Python 的显著优点在于速成，对于时间短、变化快的需求而言尤为胜任。

Python 最强大的地方体现在它的两个称号上：一个是"内置电池"；另一个是"胶水语言"。前者的意思是，Python 官方本身提供了非常完善的标准模块库，包括针对网络编程、输入/输出、文件系统、图形处理、数据库、文本处理等领域。模块库相当于已经编写完成打包供开发者使用的代码集合，程序员只需通过加载、调用等操作手段即可实现对库中函数、功能模块的利用，从而省去了自己编写大量代码的过程，让编程工作看起来更像是在"搭积木"。除了内置库，开源社区和独立开发者长期为 Python 贡献了大量的第三方库，其数量远超其他主流编程语言，可见 Python 的语言生态已然相当壮大。

"胶水语言"是 Python 的另一个亮点。Python 本身具有可扩展性，它提供了丰富的 API 和工具，能够把用其他语言制作的各种模块（尤其是 C/C++）很轻松地结合在一起。就像使用胶水一样把用其他编程语言编写的模块黏合过来，让整个程序同时兼备其他语言的优点，起到了黏合剂的作用。常见的一种应用情形是，使用 Python 快速生成程序的原型（有时甚至是程序的最终界面），然后对其中有特别要求的部分，用更合适的语言改写。例如，3D 游戏中的图形渲染模块，由于对性能要求特别高，因此可以用 C/C++ 重写，而后封装为 Python 可以调用的扩展类库。正是这种多面手的角色让 Python 近几年在开发者世界中声名鹊起，因为互联网与移动互联时代的需求量急速倍增，大量开发者急需一种极速、敏捷的工具来助其处理与日俱增的工作，Python 发展至今的形态正好满足了他们的愿望。

Python 语言的特点如下。

（1）简单易学。Python 语言关键字少，结构简单，语法清晰，方便阅读。

（2）免费开源。任何一个开发者都能够自由地分发相关副本，阅读其源代码并把它运用到新的开源软件中，就好像所有的程序开发者都是 Python 源代码的 bug 检测员，一个功能会一直被优秀的程序员改进的语言一定是越来越优秀的。

（3）自动内存管理。Python 语言可以随意声明变量而无须提前申请内存，解释器承担了程序的内存管理工作，使程序员从内存事务中解脱出来，能够更集中精力到实际程序功能的开发中，从而提高开发效率。

（4）可移植。基于其开放源代码的特性，Python 可以被移植（也就是使其工作）到许多平台包括 Linux、Windows、FreeBSD、macOS、Solaris、OS/2、Amiga、AROS、AS/400、BeOS、OS/390、z/OS、Palm OS、QNX、VMS、Psion、Acorn RISC OS、VxWorks、PlayStation、Sharp Zaurus、Windows CE 等。

(5) 解释型语言。Python 程序不需要被编译成二进制代码即可直接运行。在内部，Python 将源代码转换成一种称为字节码的中间格式，然后将其翻译成计算机的机器语言。解释型语言的最大优点是不同系统之间的兼容性好。

(6) 丰富的类库。Python 标准库很大，并且是跨平台的，可以在 Windows、UNIX、macOS 之间兼容，能够帮助开发者完成许多工作，包括正则表达式、文档生成、单元测试、线程、数据库、网页浏览器、CGI(通用网关接口)、FTP(文件传送协议)、电子邮件、XML(可扩展置标语言)、XML-RPC(远程方法调用)、HTML(超文本置标语言)、WAV(音频格式)文件、加密、GUI(图形用户界面)以及其他系统相关的代码。

Python 的流行是与人工智能、大数据等领域的发展相关的。未来是 AI 的时代，Python 语言是擅长人工智能开发的语言。表 1-1 介绍了 Python 语言的主要应用领域及相关描述。

表 1-1　Python 语言的主要应用领域

应用领域	相 关 描 述
科学计算	随着 Numpy(开源的数值计算扩展库，可用来存储和处理大型矩阵)、SciPy(专为科学和工程设计的 Python 工具包，包括统计、优化、整合、线性代数、傅里叶变换、信号和图像处理、常微分方程求解器等)、Matplotlib(二维绘图库，可以轻松绘制直方图、功率谱、条形图、错误图、散点图等)、Pandas(纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具)、Enthought librarys(包含了之前所说的众多科学计算库)等众多程序库的开发，Python 语言越来越适用于科学计算
图形处理	TKinter GUI 库、图像处理库的 PIL、PyQt、PySide、wxPython、PyGTK、PyGObject、PyGUI 等非常优秀的 Python 图形应用 GUI 开发框架，是快速开发桌面应用程序的利器
文本处理	Python 提供了 jieba、re 等功能强大的文本处理模块，还提供了大量的第三方库用于文本处理，如 Gensim 模块用于从非结构化的文本中无监督地学习到文本隐层的主题向量表达；支持包括 TF-IDF、LSA、LDA 和 word2vec 在内的多种 NPL 模型，如 NLTK、Python-docx、PyPDF2 等
网络爬虫	Python 语言有网络爬虫功能库 Requests、网络爬虫框架 Scrapy、Web 页面爬取系统 Pyspider 等大量开源爬虫网络库、爬虫框架、解析库
人工智能	Python 在人工智能领域内的机器学习、神经网络、深度学习等方面都是主流的编程语言，大量的 AI 算法都是基于 Python 语言的，目前应用较广泛的机器学习工具有 Scikit-learn、TensorFlow、MXNet 等
多媒体应用	Python 的 PyOpenGL 模块封装了 OpenGL 应用程序编程接口，能进行 2D 和 3D 图像处理。Pygame、Pyglet 以及 Cocos 2D 等框架可用于编写游戏软件。Python 可以直接调用 Open GL 实现 3D 绘制，这是高性能游戏引擎的技术基础
Web 开发	Python 拥有许多免费的 Web 模板系统和与 Web 服务器进行交互的库，Django、flask、TurboGears、web2py 等 Web 开发框架逐渐成熟，支持 HTML、XML 等置标语言，程序员可以更轻松地开发和管理复杂的 Web 应用

1.3　安装与运行开发环境

正如之前所说，计算机只能执行机器语言设定的指令代码，Python 语言是一种解释型高级语言，要通过解释器才能将写好的程序翻译成机器语言控制计算机执行相关操作。一

般在 Linux、UNIX 和 macOS 系统中都已经安装了 Python 解释器，不需要单独安装，只要用 Vim、Sublime 等编辑器编写好相应代码即可直接运行。Windows 系统没有这样的现成解释器，需要用户自行安装开发环境。开发环境包含解释器、库管理工具以及其他的辅助工具，可通过语法检查、代码格式自动生成等来提高编码效率。

经常有人问：我想学 Python，下载哪个开发环境好？这就好比美食家说，“我想做美食，用什么样的厨房好？西式？中式？集成厨房？农村三眼灶？”其实对于初学者而言，各种开发环境都可以，它们各有所长。但是不管用哪个，核心是“灶台”不能少，解释器就是核心，只要包含解释器，无论用哪种开发工具（常用的开发环境有 Python IDLE、Pycharm、Anaconda、Python Tutor 等）都能让计算机按指令操作。

本书采用的编程环境是 Python 官方网站（https://www.python.org）上可以直接下载的 Python IDLE（integrated development and learning environment）。下面介绍 Windows 操作系统、安装编程环境的具体步骤。

（1）进入官网下载中心 https://www.python.org/downloads/windows/，也可以在主页上单击 downloads 按钮，出现下拉菜单后单击 Windows 链接进入下载中心。通常首先出现的都是最新版本的开发环境，因为 Python 3.x 向后兼容，下载最新版本完全适用本书的学习内容。本书采用版本为 Python 3.7.6，建议安装此版本以后的任意版本。如果计算机为 64 位 Windows 系统，可以双击图 1-3 中方框所选项目，保存 EXE 安装包。

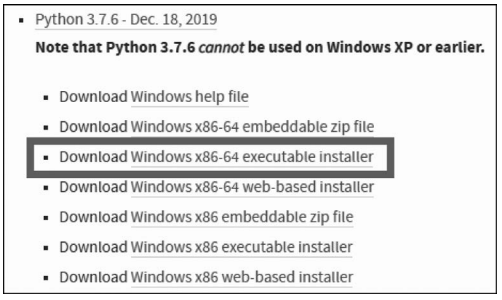

图 1-3　Python 3.7.6 安装包下载界面

（2）双击 EXE 安装包（默认文件名为 python-3.7.6-amd64.exe）进入安装界面，如图 1-4 所示。对于初学者，建议勾选 Add Python 3.7 to PATH，这样可以自动配置环境变量，确保 PATH 路径包含解释器 Python.exe。

（3）单击 Install Now，就可以自动安装 Python 3.7.6 开发环境了。

（4）安装结束后，可以在 Windows 桌面左下角“开始”菜单中找到 Python 3.7。双击 IDLE（Python 3.7 64-bit），打开的界面如图 1-5 所示，这就是 Python 集成开发环境 IDLE 窗口，也称为 Python Shell，这是一个交互式运行窗口。

在交互式运行窗口中，可以在命令提示符>>>后边输入一行代码，按 Enter 键能够立刻显示运行结果。但是命令提示符后面不能复制粘贴多行代码后再按 Enter 键运行，这是初学者很容易犯的错误。

Python 的集成开发环境支持两种运行方式：交互式和脚本式。交互式运行窗口主要功能是显示结果，或者测试一行代码的功能，不能作为程序的核心编写工具，因为这个窗口需要逐条编写并运行代码，且无法保存程序，也不方便调试。当需要编写较复杂的大段程序

图 1-4　Python 3.7.6 安装界面

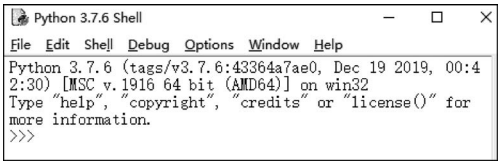

图 1-5　Python 3.7.6 集成开发环境 IDLE 窗口

时，可以采用脚本式运行方式。在 Python Shell 中（交互式运行窗口）选择 File→New File 命令，就可以打开一个文本编辑窗口，输入所有代码后执行 File→Save 或 Save As 命令，就能保存一个扩展名为.py 的脚本文件。如果不执行保存操作，直接选择 Run→Run Module 命令（也可以按 F5 键或 Fn+F5 组合键，不同计算机的热键可能不同），则首先弹出保存界面，然后运行程序，运行结果显示在交互式运行窗口中。

提示：对初学者而言，经常会出现输错代码、拼写错误、大小写错误、混用中英文标点、混用空格和 Tab 键等问题。平时写程序一定要靠自己一个字符一个字符地输入，不仅能快速提高编程能力，更能解决多行代码一次性复制粘贴在交互式运行窗口报错的问题。

图 1-6 所示是部分在交互式运行窗口中逐行执行的非常简单的代码。命令提示符>>>后边为输入的代码，#后面为注释内容，用于说明代码功能。注释只能放在代码后面，不会被解释器解释，因此不用担心写错。每一个命令提示符下面是当前代码的运行结果。

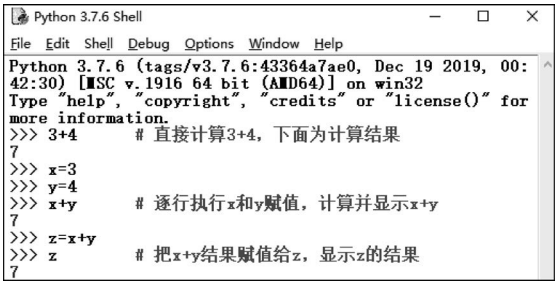

图 1-6　交互式运行窗口下逐行输入代码及运行结果

1.4　输入/输出函数

“巧妇难为无米之炊”，没有数据，就谈不上加工处理数据，更谈不上处理技巧，也就是算法。程序设计的本质就是“数据结构＋算法”。一般的程序结构都是基于 IPO（Input-Process-Output）的模式如图 1-7 所示。

图 1-7　基于 IPO 模式的程序结构

为了方便程序与用户的交互，本节首先介绍 Python 语言的内置输入/输出函数。

1.4.1　输入函数

在程序设计中，数据的输入有很多种方式，如用户直接输入、通过文件输入、通过网络输入、通过随机数据输入、通过程序内部参数输入等。本小节主要介绍采用 Python 内置函数实现用户直接输入的方法。input()函数的语法格式如下。

```
变量 = input( '提示信息: ')
```

语法说明如下。

（1）input()函数引号中的提示信息可以省略不写，但是为了更好地跟用户交互，建议书写合适的提示信息，提示用户在这个信息后面输入相关内容。

（2）提示信息可以使用字符串常量（必须用引号）或字符串变量。

（3）input()函数执行后用户输入的内容可以赋值给一个变量，变量的概念在后续章节会详细介绍。

（4）利用 input()函数输入的数据类型必须是字符串。

下面为交互式运行窗口下，部分 input()函数的运行结果。

```
>>> x = input('请输入课程名称: ')                # 显示提示信息,在后面输入内容
请输入课程名称: Python
>>> x                                          # 显示 x 的结果
'Python'
>>> y = input()                                # 没有任何提示信息,用户体验会变差
Python
>>> y                                          # 显示 y 的结果
'Python'
>>> z = '请输入课程名称: '                      # 给变量 z 赋值为一个字符串
>>> k = input(z)                               # 提示信息采用变量,注意 z 的两边没有引号
请输入课程名称: Python
>>> k                                          # 显示 k 的结果
'Python'
```

思考：尝试在交互式运行窗口利用 input()函数给变量 x 和 y 输入两个数字，然后运行 x＋y，看看会发生什么？思考其原因。

1.4.2 输出函数

数据的输出也有很多方式，常见的数据输出模式有屏幕显示输出、文件输出、网络输出、操作系统内部变量输出等。本小节主要介绍利用 Python 的内置函数实现屏幕显示输出的方法。print()函数的语法格式如下。

```
print(value1, value2, ...[, sep = ''][, end = '\n'][, file = sys.stdout][, flush = Truel
False)]
```

语法说明如下。

(1) 在语法格式中出现的中括号若不做特殊说明表示括号内的参数可以缺省，缺省参数的值为系统默认值。

(2) print()函数是一个操作，因此不需要赋值给任何变量。

(3) value 表示输出的具体内容，省略号表示可以打印不止一个数据项，各个数据项之间用逗号分隔。

(4) sep 参数表示输出多个数据项时，用指定分隔符进行分隔，默认为一个空格。

(5) end 参数表示输出完 print()函数内的所有数据内容后光标的具体位置，默认为换行符'\n'，即换行到下一行等待后续输出。

(6) file 参数表示将文本输出到文件对象 file 中，file 对象可以是数据流、磁盘文件等，默认为 sys.stdout，即直接输出到控制台即交互式运行窗口。

(7) flush 参数用于内存管理，flush 为 True 会在 print()结束之后，立即将内存中的数据显示到屏幕上，并清空缓存，默认为 False。

为了更好地理解 Python 集成开发环境支持的两种运行方式，图 1-8 显示了文本编辑窗口中的程序代码及在交互式运行窗口中显示的运行结果(注意是两个窗口界面，上面是文本编辑窗口，下面是交互式运行窗口)。在文件编辑窗口的顶端会显示保存的文件名及相关路径；而在交互式运行窗口中则是在两个命令提示符之间显示当前运行结果来源，包含文件的路径、文件名及运行结果。

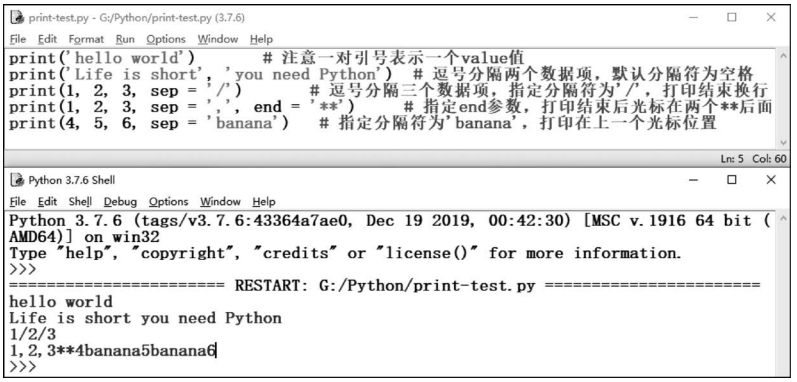

图 1-8　Python 支持两种运行方式

提示：如果将文本编辑窗口比喻为厨房，那么交互式窗口就是餐桌。可以在餐桌上简单地做一些凉拌菜，但是大菜还是需要厨房出品，而餐桌是用来放成品的。所以在开始学习