高等院校信息技术规划教材

计算思维导论

——一种跨学科的方法

李　暾　编著

清华大学出版社

高等院校信息技术规划教材

计算思维导论

——一种跨学科的方法

李　暾　编著

清华大学出版社
北京

内 容 简 介

本书兼顾计算机科学基础知识和计算思维，以通俗易懂的方式介绍计算思维如何应用于各学科领域（含计算机科学）解决问题。本书以 Python 作为实践语言，展现利用计算思维解决问题方法的实现。通过这种跨学科应用问题求解的学习和实践，希望培养学生主动在各专业学习中利用计算思维的方法和技能，进行问题求解的能力和习惯，并能动手解决具有一定难度的实际问题。

本书适合作为高等院校计算机及相关专业的教材，也可以作为计算思维爱好者的读物。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目（CIP）数据

计算思维导论：一种跨学科的方法/李暾编著. --北京：清华大学出版社，2016
高等院校信息技术规划教材
ISBN 978-7-302-44225-7

Ⅰ. ①计… Ⅱ. ①李… Ⅲ. ①计算机科学－高等学校－教材 Ⅳ. ①TP3

中国版本图书馆 CIP 数据核字（2016）第 152473 号

责任编辑：白立军
封面设计：傅瑞学
责任校对：焦丽丽
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 刷 者：北京富博印刷有限公司
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：14.5　　**字　　数**：330 千字
版　　次：2016 年 9 月第 1 版　　**印　　次**：2016 年 9 月第1次印刷
印　　数：1～2000
定　　价：29.00 元

产品编号：069156-01

前言 foreword

从2008年开始，以**计算思维**的培养为主线开展计算科学通识教育，逐渐成为国内外计算机基础教育界的共识。2010年首届“钱学森创新拓展班”开始，作者就不断地在“大学计算机基础”课程中尝试计算思维基本概念、能力和技能的讲授以及能力培养。通过调整课时和授课内容比例，不断加大计算思维内容的比重。经过近5年的摸索，对计算思维的教学内容、授课方式、实践环节等有了很清楚的认识，形成了明确的思路，积累了大量的资料，才有了本书的成书。

编写本书的指导思想是兼顾计算机科学基础知识和计算思维，以计算思维授课内容为主，将原来的数据表示、计算机硬件、网络等知识穿插进来，选择**Python**作为实践语言，授课内容更偏重于计算思维如何应用于各领域解决问题，各领域包括计算机科学领域。实践内容将在授课内容的基础上进行拓展，并要求运用Python及相关的配套库进行问题求解练习。最终，我们希望通过这种跨学科应用求解的讲授和实践，培养学生在理解计算机系统的基础上，主动在各自专业学习中利用计算思维的方法和技能，进行问题求解的能力和习惯，能动手解决具有一定难度的实际问题。

考虑到对大部分学生来说，“大学计算机基础”课可能是大学期间的少数几门计算机科学相关课程之一，因此，本讲义更强调广度，涉及很多领域，使得学生在今后的学习、生活和工作中碰到问题时，可以考虑该问题是否能有**计算的**解决方法，并能借助计算思维和计算装置完成任务。本书在选择应用领域和案例时，着重在那些易于理解、不需要掌握算法和程序设计就能解决的问题上，因此，本书不会讲解算法，而是着重于如何利用计算思维理解和解决问题，展现计算思维在问题求解、系统构造、理解人类行为等方面发挥的重要作用。

本书的主要目标是帮助读者理解和掌握计算思维解决问题的基本方法和技能，并能较为熟练地应用这些方法和技能有效地解决其他问题。通过本课程的学习，将学会如何利用计算思维构造问题

求解框架、如何对问题进行抽象和建模、如何将数学或物理上的模型转换为能自动执行的模型等。总之，理解和掌握计算思维及计算机问题求解的艺术。

本书适用于计算机专业和非计算机专业一年级新生，不要求有计算机程序设计经验，并且也不是以程序设计为主要内容，而是要求学生专注于理解计算思维求解问题的方法和技能。一些 Python 语言基础知识的介绍是帮助学生阅读和理解讲义中给出的 Python 程序，并能在理解的基础上，对这些程序进行小修改就能实践自己的问题求解方法。

本讲义的内容分为三部分，强调系统化的问题求解和计算思维两个 A(Abstraction & Automation)的威力。

(1) 第一部分是计算导论，介绍计算思维的基本概念和基本技能、计算机问题求解的方法和本质，以及 Python 语言简介。

(2) 第二部分结合计算机科学相关的知识，探讨计算思维在这些问题的解决上的体现，以及一些基本的问题求解策略。

(3) 第三部分结合一些实际的应用背景和热点话题，介绍计算思维在解决实际问题上的体现。

通过本书的学习，希望读者最后将能：

(1) 列出计算思维的基本概念，较为熟练地利用本书所讲计算思维技术进行问题求解。

(2) 说出计算思维和计算机问题求解的本质。

(3) 能用程序设计语言，如 Python，表达计算。

(4) 能利用系统化的问题求解方法，完成从规划问题求解步骤，到用程序正确地表达计算整个完整的问题求解过程。

(5) 掌握一些常用的计算方法和计算工具，如随机方法、图、模拟等。

(6) 列出一些计算思维在各领域的应用案例，以及计算思维在其中发挥的重要作用。

由于本书编写时间仓促，加之作者水平有限，书中难免出现谬误，恳请读者不吝赐教。

编　者

2016 年 5 月

此为试读，需要完整PDF请访问：www.ertongbook.com

目录 contents

第一部分　计算思维导论

第 1 章　计算概论 …… 3

1.1　计算 …… 3
1.2　小结 …… 9
习题 …… 9

第 2 章　Python 简介 …… 10

2.1　Python 基本元素 …… 10
2.1.1　对象、表达式和数值类型 …… 11
2.1.2　变量和赋值 …… 12
2.2　分支语句 …… 14
2.3　str 类型与输入 …… 15
2.4　循环 …… 16
2.5　内置数据结构 …… 18
2.5.1　列表 …… 18
2.5.2　元组 …… 20
2.5.3　字典 …… 22
2.6　函数 …… 22
2.7　文件 …… 25
2.8　小结 …… 26
习题 …… 26

第 3 章　计算思维与计算机问题求解 …… 29

3.1　计算思维 …… 29
3.2　计算机问题求解 …… 32

3.3 算法复杂度 …… 36
3.4 计算机问题求解的核心方法 …… 38
3.5 小结 …… 42
习题 …… 42

第二部分 计算机科学篇

第 4 章 递归 …… 49

4.1 定义及应用 …… 49
4.2 递归与数学归纳法 …… 53
4.2.1 最大子集问题 …… 53
4.2.2 排序 …… 56
4.3 动态编程 …… 58
4.4 小结 …… 60
习题 …… 60

第 5 章 信息、信息表示及处理 …… 62

5.1 信息论基础 …… 62
5.2 信息的数字化 …… 64
5.2.1 数值的数字化 …… 65
5.2.2 字符的数字化 …… 67
5.2.3 声音的数字化 …… 69
5.2.4 图像的数字化 …… 70
5.3 数据压缩 …… 71
5.3.1 Huffman 编码 …… 72
5.3.2 Python 实现 …… 75
5.4 信息加解密 …… 78
5.5 小结 …… 87
习题 …… 87

第 6 章 面向对象程序设计 …… 89

6.1 Python 面向对象基础 …… 90
6.2 一个实际的例子：按揭贷款 …… 94
6.3 数据的图形化 …… 97
6.4 小结 …… 102
习题 …… 102

第 7 章 计算机系统 …… 103

7.1 概述 …… 103
7.2 数字电路 …… 105
7.2.1 逻辑门的建模与模拟 …… 106
7.2.2 加法器 …… 110
7.2.3 存储电路 …… 113
7.3 计算机硬件系统 …… 115
7.4 小结 …… 121
习题 …… 121

第 8 章 图灵机与图灵测试 …… 123

8.1 图灵机 …… 123
8.2 图灵测试 …… 129
8.2.1 正则表达式简介 …… 131
8.2.2 简单图灵测试程序 …… 133
8.3 小结 …… 138
习题 …… 138

第三部分 应 用 篇

第 9 章 模拟、概率与统计 …… 143

9.1 随机与概率 …… 143
9.2 数据分布 …… 152
9.3 正态分布与置信区间 …… 155
9.3.1 均匀分布 …… 156
9.3.2 指数分布 …… 156
9.3.3 几何分布 …… 156
9.3.4 Benford 分布 …… 158
9.4 随机数生成 …… 160
9.5 小结 …… 170
习题 …… 170

第 10 章 蒙特卡洛模拟方法 …… 172

10.1 概述 …… 172
10.2 初探——模拟赌局 …… 173

10.3 计算 π …… 177
10.4 游荡的醉汉 …… 179
10.5 高手赢面就大吗 …… 188
10.6 小结 …… 192
习题 …… 193

第 11 章 数据分析概览 …… 194

11.1 概述 …… 194
11.2 乳腺癌的诊断 …… 195
11.3 小结 …… 204
习题 …… 204

第 12 章 排队问题 …… 205

12.1 排队论基础 …… 205
12.2 SimPy 简介 …… 207
12.3 需要多少小便斗 …… 216
12.4 小结 …… 222
习题 …… 223

第一部分
计算思维导论

本部分是本书的基础部分，介绍与计算相关的基础知识，包括计算的概念、计算思维与计算机问题求解，以及 Python 语言。通过本章的学习，应能：

(1) 说出计算、计算的解、算法等基本概念；

(2) 列出计算思维的核心概念、计算机问题求解的基本步骤；

(3) 说出计算机问题求解的核心思想，并能用算法效率概念尝试改进算法；

(4) 能看懂 Python 程序，会编写简单的 Python 程序。

此为试读,需要完整PDF请访问: www.ertongbook.com

第1章 chapter 1

计算概论

从小学开始，“计算”这个词就不断出现在日常生活、数学作业中，如“苹果18元一斤，算一下买3斤苹果要多少钱”。那么，计算与计算机系统之间的关系是什么呢？本章将围绕计算展开相关基础知识的介绍。

1.1 计　　算

对计算买苹果要多少钱的问题，一般可用两种方法进行解答：一是3个18相加，二是18乘以3。对第一种方法，通常列出竖式，个位与个位对齐，十位与十位对齐。然后将个位上的3个8相加，得到24，直接在结果的个位写上4，2进位到十位，与3个1相加得到5，结果为54。对第二种方法，也可列竖式，首先将18的个位8与3相乘，得到24，将4写到结果的个位上，2进位到十位，十位的1与3相乘得3，与进位的2相加得5，结果也为54。当然，这样的问题也可直接用计算器求解，输入18×3就能得到结果。从这个例子，可以看出计算的一些特性，可将其定义为如下。

定义1-1　计算指的是根据已知条件，从某一个初始点开始，在完成一组良好定义的操作序列后，得到预期结果的过程。

对这个定义，有以下两点需要注意。

(1) 计算的过程可由**人**或**某种机器**执行。

(2) 同一个计算可由不同的**技术**实现。

在人类历史上，计算的作用受到了人脑运算速度和手工记录计算结果的制约，使得能通过计算解决的问题规模非常小。相对于制约计算的人的因素，计算机非常擅长于做(也只能做)两件事情：运算、记住运算的结果。随着**计算机**(Computer)的出现，以及计算机运算速度的不断提高，能通过计算解决的问题越来越多、问题规模也越来越大，即越来越多的问题被证明存在**计算的解**(Computational Solution)。所谓有计算的解，指的是对某个问题，能通过定义一组操作序列，按照该操作序列行为能得到该问题的解。

一般来说，知识可分为陈述性的知识或过程性的知识。**陈述性知识**(Declarative Knowledge)是对事实的描述。例如，“x的平方根是一个数y，使得$y\times y=x$”。但是，从平方根的描述，无法知道如何去求某数的平方根。而**过程性知识**(Imperative Knowledge)描述的是“如何做”，或演绎信息的动作序列。例如，古希腊亚历山大里亚的

数学家希罗第一次给出了一种计算平方根的方法，描述如下。

(1) 对给定的数 x，猜测其平方根为 g。

(2) 如果 $g\times g$ 足够逼近 x，停止，并报告 g 就是 x 的平方根。

(3) 否则，用 g 和 x/g 的平均值作为新的猜测。

(4) 该新的猜测，仍称其为 g，重复上述过程，直到 g 足够逼近 x。

例如，用上面的方法求 49 的平方根，计算过程如下。

(1) 猜测 49 的平方根为 6，即 g 为 6。

(2) $6\times6=36$ 不够逼近 49。

(3) 令 $g=(6+49/6)/2=7.0833$。

(4) $7.0833\times7.0833=50.17$，不够逼近 49。

(5) 令 $g=(7.0833+49/7.0833)/2=7.00049$。

(6) $7.00049\times7.00049=49.007$，已足够逼近 49。停止，并称 7.00049 足够近似于 49 的平方根。

希罗求平方根的方法是由一组简单动作的序列，以及规定每一个动作何时执行的控制构成的。这就是计算定义中所指的“一组良好定义的操作序列”，又称为**算法**(Algorithm)。

定义 1-2　算法是求解**问题类的**、**机械的**、**统一的**方法，它由有限个步骤组成，对于问题类中的每个给定的具体问题，**机械地**执行这些步骤就可以得到问题的解答。

可以用两数加法的运算方法来理解算法的概念。数的个数是无限的，因此可能要做的加法也是无限次的。但是无论做多少次加法，做加法的方法是不会变的。因此，做加法的方法是一种运用有限的规则应对无限可能情况的方法！算法正是这样一种方法，它是用来解决一类问题的。

与菜谱类似——按照这些步骤就能做出这道菜——可将算法理解为遵循这些步骤，就能解决你的问题。利用一组良好定义的序列来解决问题的思路可上溯到古希腊、波斯和中国古代，例如，古希腊数学家欧几里得在公元前三世纪，就提出了寻求两个正整数的最大公约数的“辗转相除”算法，该算法被人们认为是史上第一个算法。Algorithm 一词来源于波斯学者 Muhammand ibn Musa al-Khwarizmi 的名字，他定义了加减乘除等运算的过程，按上述定义，这些过程即为算法。

算法通常具有五大特征。

(1) 输入：一个算法必须有零个或零个以上输入量，用于描述要解决的问题。

(2) 输出：一个算法应有一个或一个以上输出量，输出量是算法计算的结果。

(3) 明确性：算法的每个步骤都必须精确地定义，拟执行的动作的每一步都必须严格地、无歧义地描述清楚，以保证算法的实际执行结果精确地符合要求或期望。

(4) 有限性：算法在有限个步骤内必须终止。

(5) 有效性：又称为可行性或能行性，是指算法的所有运算必须是充分基本的，因而原则上人们使用笔和纸可在有限时间内精确地完成它们。

算法描述了对数据进行加工处理的顺序和方法，从上面希罗算法的例子可以看出，动作难以严格按照所给顺序一个一个地进行，不可避免地会遇到需要进行**选择**或不断**重**

复的情况。通常使用**顺序结构**、**选择结构**和**循环结构** 3 种**控制结构**来组织算法中的动作。

(1) 顺序结构：算法的各个动作严格按它们的先后顺序依次执行，前一个动作执行完毕后，顺序执行紧跟在它后面的动作步骤。

(2) 选择结构：提供了一种根据判断的不同结果，分别执行不同的后续操作的控制机制。

(3) 循环结构：通常包括**循环控制条件**和**循环体**。循环控制条件描述了循环反复执行的条件，而循环体则描述了每次循环如何对数据进行处理的动作(序列)。

已经证明，任何算法都可用这 3 种结构描述，即这 3 种结构是组织算法动作的最小集合。

算法规定的动作序列可由人或机器来执行。以求平方根方法为例，当人来执行时，首先要能理解所描述的各动作的含义：第(1)步的“猜测”、第(2)步的“乘”和“足够逼近”、第(3)步的$(g+x/g)/2$等运算，以及第(4)步“重复”等的含义。其次，在理解这些含义的基础上能做相应的动作，即能完成“猜测某个数并用符号 g 表示”、“乘法运算”、“加法运算”、“除法运算”，以及“回到第(2)步开始执行”等动作。最后，要能够根据动作序列，自动化、机械地完成序列的执行，这包含两个方面的机制：一是记住了(在脑海中或在纸上)当前执行的动作，以及知道下一步该执行哪一步动作；二是记住了(在脑海中或在纸上)中间结果(如不同时刻 g 的值)。

与此类比，如果能设计一台机器，该机器能像人一样“理解”动作的含义、“执行”相应的动作(即能实现乘、除、加、比较和重复等操作)、能记住正在执行的和下一步要执行的动作序列，以及能记住相应的中间结果，那么，就能用这台机器代替人来进行求平方根的运算。通常称这样的机器为**固定程序计算机**(Fix-program computer)，即只能做特定事情的机器。事实上，很多早期的计算机都是这类计算机，如图灵的 Bombe 机器，只能破解德军 Enigma 密码，而不能做别的事。

那么，人除了能完成求平方根的计算，还能执行其他的计算序列吗？当然能，前提是列出的计算序列中的动作是能被人理解、并且人有能力做的动作。再结合“记住”的能力，人就能自动地、机械化地执行其他的计算了。

与固定程序计算机不同，第一台真正的现代计算机 Mark I 被称为**存储程序计算机**(Stored-program computer)。该计算机实现了**冯·诺依曼体系结构**，如图 1-1 所示。在该体系结构中，计算机由 5 部分组成：存储器、运算器、控制器、输入设备和输出设备。冯·诺依曼体系结构通常提供一组最基本的**指令**(“动作”)集合，称为**指令集**。能执行这些指令的执行机构称为**运算器**，运算器能进行的操作通常是算术运算和逻辑运算。**存储器**用于保存用指令集内指令编写的指令序列(即**程序**)，以及其操作的对象(即**数据**)。运行指令序列(程序)时，在**控制器**控制下，首先从存储器读入指令和数据，其次对读入的指令进行理解，最后控制运算器执行相应的动作以对数据进行操作。控制器还可控制从输入设备读入数据，以及向输出设备输出数据。

在实现上，通常将运算器和控制器集中在一起，构成**中央处理器**(Central Processing Unit，CPU)，它是现代计算机的“大脑”。CPU 中还包括各类**寄存器**，用于保存计算过程

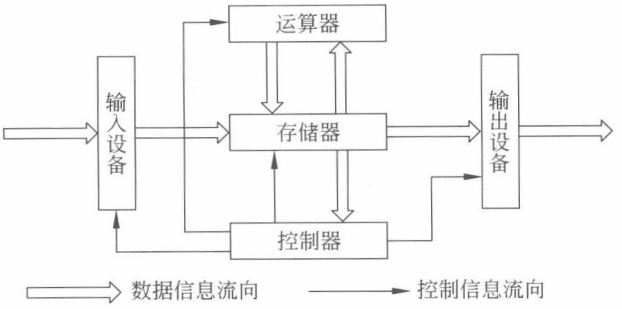

图 1-1　冯・诺依曼体系结构

的中间结果。存储器是分层次的——分为**主存**和**辅存**两层，CPU 只能直接访问保存在主存中的数据。主存的访问速度很快，但保存的数据是**易失性**的，即关闭计算机电源后，其保存的数据就丢失了。辅存的访问速度相对慢很多，但其存储的数据是**永久性**的，通常利用磁介质等保存数据，如个人电脑的硬盘、光盘、USB 盘等。**输入输出**设备是计算机与外界的联系通道，类型多种多样，如用于用户输入的鼠标和键盘，用于输出的显示器，以及用于长期存储数据和程序的磁盘等。除这些部件外，现代计算机通常利用总线连接计算机各部件，**总线**(Bus)是一组电子管线，它负责在各个部件之间传递信息。

冯・诺依曼体系结构的核心思想——**存储程序**——使得计算机变得通用和可编程，即能通过编写不同的程序，很容易地使计算机具备不同的功能。存储程序指的是将动作序列转换成用指令集内指令编写的程序，然后将程序及其操作的数据一起保存于存储器中(二进制形式)，利用 CPU 中的**程序计数器**(Program Counter)指向将要执行的指令在内存中的位置，用控制器从该位置取指令执行，然后自动更新程序计数器指向下一要执行指令的位置，以此实现程序的自动执行。将程序与数据以同样的形式存储于主存中的特点，对于计算机的自动化和通用性，起到至关重要的作用。甚至可以说：现代计算机是一个在可更换的程序控制下存储和处理信息的机器。在冯・诺依曼体系结构形成之前，人们将数据存储于主存中，而程序被看成是控制器的一部分，两者是区别对待和处理的，由此每台机器只能完成固定的任务，而不是通用和可编程的。

指令是计算机执行的最小单位，由**操作码**和**操作数**两部分构成，如图 1-2(a)所示。

操作码	操作数(参加运算的数据、结果数据或这些数据的地址)

(a) 指令格式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	0	1	0	0	0	0	1	1	0
	ADD				R6			R2						R6	

(b) 加运算的机器指令形式

ADD R6, R2, R6

(c) 加运算的汇编指令形式

图 1-2　指令

操作码表示指令的功能，即执行什么动作，操作数表示操作的对象是什么，例如，寄存器中保存的数据。计算机能识别的指令是由 0 和 1 构成的字串，称为**机器指令**。图 1-2(b)给出了某款 CPU 的加法指令的示意图。机器指令适合于机器理解，但是，不适合人使用。因此，在指令中引入了助记符表示操作码和操作数，以帮助人理解和使用指令。这样的指令称为**汇编指令**，如图 1-2(c)所示。

CPU 中的程序计数器是一个寄存器，保存内存中待执行程序某条指令的位置(称为**地址**)。开始执行程序时，程序计数器指向程序的第一条指令，在执行第一条指令的同时或之后，程序计数器会自动“加 1”，指向第二条指令，以此方式依次逐条执行程序中的指令。但是程序计数器并不总是“加 1”，有时会根据某条指令的运行结果，“跳转”指向程序中一条其他的指令，而不是指向顺序的下一条指令。此时，程序将从程序计数器所指向的其他指令处开始执行，这种跳转称为**控制流**，是利用简单的指令编写出复杂行为程序的必要机制。

综上可知，离开了各种各样的程序，计算机自身无法完成各种任务，但是在各种程序的指挥下，计算机几乎能干任何事情。即在人的指挥下，计算机几乎什么都能干！好的程序员能利用给定的机器指令集编写出各种各样有用的程序，这也是**程序设计**(Programming)的魅力所在。

机器指令和汇编指令又分别称为**机器语言**和**汇编语言**，它们非常贴近于计算机，不适合人用来编写程序。目前，人们常常用 C/C++ 、Java、Python 这类**高级语言**编写程序。此时，需要一个**编译器**或**解释器**将高级语言转化为计算机能理解的指令。

(1) 编译器的功能是将高级语言编写的程序翻译成等价的机器语言，使其能直接在计算机上运行。其运行模型如图 1-3 所示。

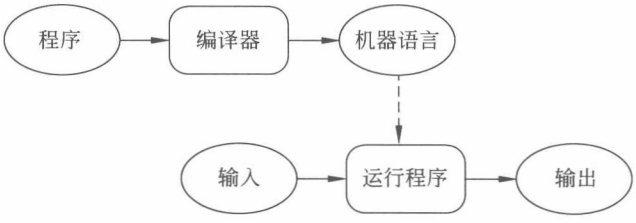

图 1-3 基于编译的程序执行

(2) 解释器模拟一台能理解某高级语言的计算机，并在这台模拟出来的计算机上，以逐条执行程序语句的方式，来运行程序。其运行模型如图 1-4 所示。

机器语言、汇编语言和高级语言统称为**程序设计语言**(Programming language)。本书采用 Python 语言作为示例和实践语言。

程序设计语言的定义由 3 个方面组成，即语法、语义和语用。**语法**定义了一个程序的组成结构或组织形式，即构成语言的各个部分之间的组合规则，但不涉及这些部分的特定含义，也不涉及使用者。如在英语中“Cat dog boy”是不满足语法的句子，因为英语语法中未定义形为“＜Noun＞＜noun＞＜noun＞”的句子。又例如，Python 表达式的 3.2+3.2 的语法是正确的，但是 3.2 3.2 的语法是不正确的。**语义**表示满足语法的程序

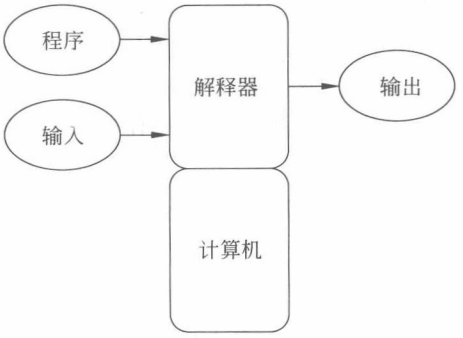

图 1-4　程序的解释执行模型

的含义，即各个语法单元的特定含义。如“I was born on the 30th February”，语法上是正确的，但是语义上是错误的，因为 2 月没有 30 号。又例如，Python 表达式 3.2/'abc'语法上是正确的，但是语义上是错误的，因为 Python 的语义定义不允许用一个数去除以一个字符串。

至此可进一步思考这样一些问题：世界上所有的问题是否都是**可计算的**（有计算的解）？程序设计语言的能力是否有强弱之分？1936 年，英国数学家阿兰·图灵在其论文“论可计算数以及在确定性问题上的应用”（*On computable numbers, with an application to the Entscheidung problem*）中，描述了一类计算装置——图灵机。图灵机用无限长的纸带表示无限的内存，在纸带上可以写上 1 和 0，以及其他的非常简单的基本指令控制对纸带的移动、读、写等操作。图灵机是一个通用的、抽象计算模型，它产生了计算的形式概念，即所谓**图灵可计算性**。**丘奇图灵论题**指出：**如果**一个函数是**可计算的**（即能用一组动作序列从输入得到预期结果），则能在图灵机上编程来计算它，即**一个函数是可计算的当且仅当它是图灵可计算的**。可见，图灵机的能力就是我们目前认识得到的“计算”的极限。丘奇图灵论题引出了**图灵完备性**概念。一种程序设计语言是图灵完备的，意味着它能被用来模拟一台通用图灵机，即意味着该语言的计算能力与一个通用图灵机相当。所有的通用编程语言和现代计算机的指令集都是图灵完备的，现代程序设计语言都提供了比通用图灵机的基本指令更丰富、更方便的指令。现存的几百种程序设计语言中，没有哪一门语言是最好的，因为用一门语言能编写的程序，也能用另一门语言编写，只是难易程度不同，而以计算能力衡量，所有语言在根本上是相等的。

不论采用何种语言、何种方法设计程序，程序设计的核心思想都是如何用给定的指令构造一个指令序列以正确地解决问题。程序设计能使计算机严格地做人们想让它做的事，因此能编写各种有趣而有用的程序，但是，当计算机没有按你的要求做事时，你也只能怪自己没有给计算机正确的命令。

注意，丘奇图灵论题中的“如果”非常重要，它意味着并不是所有的问题都是可计算的。例如：

（1）高考后，选择一所大学作为志愿：你相信计算机会帮你解决这个问题吗？

（2）邀请朋友一起看电影：机器能像人一样挑选朋友并协商好一切吗？

（3）下棋：棋子移动的规则非常简单，但是每次移动棋子时，可选的移动方案太多

此为试读，需要完整PDF请访问：www.ertongbook.com