同济大学本科教材出版基金资助

Python计算思维与问题求解

丛培盛　杨志强　朱仲良　龚沛曾　编著

同济大学出版社
TONGJI UNIVERSITY PRESS

同济大学本科教材基金资助项目

Python 计算思维与问题求解

丛培盛　杨志强　朱仲良　龚沛曾　编著

同济大学出版社
TONGJI UNIVERSITY PRESS

同济大学本科教材基金资助项目

Python 计算思维与问题求解

丛培盛　杨志强　朱仲良　龚沛曾　编著

同济大学出版社
TONGJI UNIVERSITY PRESS

内 容 提 要

本书面向非计算机专业的大学生，旨在提高学生知识交叉融合、求解问题的能力。全书分为基础篇和应用篇。基础篇以计算思维为导向，讲解计算求解的基本思维方式、Python 的基础语法及数据结构；应用篇则面向实际问题的求解，介绍常见的计算问题及 Python 中的应对方法。本书内容布局简洁合理、深入浅出，力求达到基础学科知识应用融会贯通的目的。

为便于教学，本书提供了高质量的 PPT，上机实验。线上教学配套资源丰富，包括重点章节的微课、程序代码、实例数据，通过二维码扫描即可获取。

本书既可作为高等学校工科交叉类课程的教材，也可作为各专业计算类课程的参考教材，也适合于自学者使用。

程序代码和数据文件下载

图书在版编目(CIP)数据

Python 计算思维与问题求解/丛培盛等编著. --上海：同济大学出版社，2020.7
ISBN 978-7-5608-9311-2

Ⅰ.①P… Ⅱ.①丛… 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(2020)第 107701 号

Python 计算思维与问题求解

丛培盛 杨志强 朱仲良 龚沛曾 **编著**

责任编辑 赵 黎 **助理编辑** 朱涧超 **责任校对** 徐逢乔 **封面设计** 陈益平

出版发行 同济大学出版社 www.tongjipress.com.cn
(地址：上海市四平路 1239 号 邮编：200092 电话：021-65985622)
经 销 全国各地新华书店
排 版 南京月叶图文制作有限公司
印 刷 江苏凤凰数码印务有限公司
开 本 787mm×1092mm 1/16
印 张 11.25
字 数 281 000
版 次 2020 年 7 月第 1 版 2020 年 7 月第 1 次印刷
书 号 ISBN 978-7-5608-9311-2
定 价 58.00元

目　录

第一章

计 算 思 维

计算思维是2006年以来，在全国计算机基础教学中被广泛强调培养的思维方式，是随着计算机在各领域应用面的拓展及深入，计算机基础教育工作者逐步总结归纳出来的一套用计算机的基础概念表达问题、求解问题的思维方式、基本思路。培养计算思维并使其成为习惯，对理工商专业学生培养而言，可以提供另外的思考问题角度，对促进学科交叉，提高中国大学生利用计算机求解问题的能力，有积极的促进意义。

1.1 计算思维基本概念及方法

1.1.1 计算思维基本概念

提到思维一词，一般情况下，人们会自然联想到数学中建立在公理及推导上的逻辑思维和物理学中建立在观察和实验上的实证思维，却很少听说计算思维。这或许与中小学教育期间，学校重点培养学生对数学与物理知识的理解，而计算机教育相对欠缺有关系。一般而言，人类的思维活动与其从事的工作密切相关。1972年图灵奖得主 Edsger Dijkstra 曾说过："我们所使用的工具影响着我们的思维方式和思维习惯，从而也将深刻地影响着我们的思维能力。"从这个意义上讲，计算机作为一种工具，其应用的熟练程度，在中国的教育中还需要加强。程序设计者在用计算机解决某些问题时，会根据自己的专业进行有别于其他工种的思维方式，这可以被认为是计算思维。大学计算机编程语言类课程中，会涉及编码前程序流程框图设计的内容，即用图形或语言的形式，表达出程序的实现步骤及每个步骤应该做什么、完成该步骤应该注意什么问题。这其实也是程序设计阶段的一种思维方法，属于用计算机进行问题求解时的思维范畴。

1. 计算思维的提出

计算思维（computational thinking）一词由美国卡内基梅隆大学周以真（Jeannette Wing）教授于2006年提出。她认为：计算思维是运用计算机科学的基础概念进行问题求解、系统设计、对人类行为进行理解等涵盖计算机科学之广度的一系列思维活动。其本质内涵是人们用计算机进行问题求解时，对问题的表达、提出解决方案、程序的设计与编码等步骤中采用的思维方式和方法。

视频讲解：计算思维的概念及方法

计算思维的本质是抽象和自动化，抽象是指在用计算机求解问题的过程中，把不同类别事物的本质共性分析出来，抽象可以有效地提取问题求解的关键点，抛弃琐碎环节；自动化则指熟练使用计算机的计算特长，设计出精巧的解决方案。

计算思维的培养，需要学习者在不断的实践过程中，对方法运用进行逐步体会。我们将从 1.1.2 节开始，逐步以实例的方式对计算思维的概念加以讲解，期待学习者从此对计算思维有根本认识上的转变。

2. 计算思维的特性

计算机科学本质上源自数学思维，程序化本身就存在着大量的数学逻辑实现。但因计算机科学是利用计算机解决各类问题的，其本身又受到一些限制：如存储、计算速度、精度等，不能只是数学性地思考，而必须从工程思维上加以补充，所以可以这么认为，计算机科学是逻辑和工程思维的互补与融合。

例如，梯形求积分问题，是数学上的极限问题。如果仅按数学理论，用计算机计算是无法实现的，因为分割的梯形小了还可更小，计算机如何表达无限小？因此，此类问题的解决必须考虑工程可实现问题。

3. 开展计算思维教学

计算机基础教育者们预测，计算思维将是 21 世纪中叶人类的一种基本技能，这种技能就像今天人们普遍掌握的阅读、写作和算术（Reading，wRiting，and aRithmetic——3R）技能一样，我们每个人将习惯于在不同种类问题的思考中注入计算的元素。在关于大学生计算思维能力培养的问题上，教育界已经达成了共识：计算机科学的教授应当为大学新生开一门称为“怎么像计算机科学家一样思维”的课程，面向所有专业，而不仅仅是计算机科学专业的学生。要让学生明白，计算问题无处不在，计算思维无往不利的道理。美国科学家 Stephen Wolfram 在其著作 *A New Kind of Science* 中提到，传统科学建立在数学上，新的科学建立在计算机程序上。这充分体现了计算机思维培养的重要性。

视频讲解：计算思维案例

1.1.2 人类行为理解

人类能够掌控世界，最大的优势在于其学习和知识归纳提高的能力。人类在处理不同类别的问题时，如果问题以前遇到过、或者可采取日常生活中的类比，则往往会加速对问题的理解，从而提出较可靠的解决方案，用计算机科学的基础概念对这些人类行为进行理解，属于计算思维的范畴。

计算思维案例 PPT

例如，在将一组数据排序时，联想到上体育课时按身高排队的过程，人类行为往往采用如下实现过程：

step1：从所有人中挑选最高者。

step2：将最高者放在一个位置 p_0。

step3：从剩余学生中再找最高者。

step4：将其排在 p_0 旁边的位置 p_1。依此类推，将继续挑选出来的人选放到 p_2，…，

p_n的位置，就可以完成排序过程。

可以进一步设想，如果在 step1 挑选第一个学生时就认为是从“剩余的学生中挑选最高者”，则可以将上述过程变通为反复重复 step3，step4 即可。对计算机而言，这种反复就是循环做某件事情，也是自动化的基础。

可以进一步给上述的排序过程增加一点限制，即在不增加额外空间存储需求的情况下完成排序。解决该限制可以通过记录新挑选出来的学生应该存放的位置 k 和该学生目前所在的位置 i，并将这两个位置的学生进行交换来进行。因此用计算机来完成排序的过程就可以描述为：

开始：将所有学生身高数据保存到表 L 中。

Step1：当前应存放位置 $k=0$。

Step2：进入循环。

Step3：从剩余学生中找最高者的位置 i。

Step4：交换 L 中位置 k 和 i 的学生。

Step5：位置 k 增加 1。

Step6：如果 k 等于学生总数，结束，否则转 step3。

上述过程，从一个生活中的例子，去理解问题的计算机实现，及实现自动化时应重点考虑的问题。

再如一组数据去重复项的过程，数据已存储在一个表中，因为删除重复数据还涉及数据项之间的前后关系及数据移动的问题，因而逻辑比较复杂。此时，如果类比日常生活中的例子：统计一个装有不同颜色小球的袋子里的小球共有几种颜色，它也属于去重问题。可以设想如下的步骤：

Step1：从袋子中取出一个小球。

Step2：比对这种颜色是否没记录，如果记录了，则丢弃，否则将该颜色加到色彩记录集中。

Step3：如果袋子空了，结束；否则转向 Step1。

采用上述过程，去重过程就变得很简单。该案例也很好地体现了计算思维中提到的在算法与存储之间寻求平衡的方法：牺牲了部分存储空间而使逻辑简化。在数据处理的程序设计中，尽量类比日常生活中的例子，可以有效提高对问题的理解深度，从而达到简化逻辑的目的。

1.1.3 抽象和分解

人们在处理一些复杂系统时，经常采用的手段是将问题分解，即将问题分解为互相独立或相互间关联较少的子系统，达到逐步精化求解的目的。在计算求解的领域，这种方法被称为自顶向下(Top-Down)的分析方法，很明显，成功的关键是将问题有效地分割抽象。

例如设计一个校园车牌自动识别系统，其要求是：自动识别进出校园的车牌，对于在校园内注册的车牌(教职工的私家车)，识别确认后抬杆放行，对于非注册车辆，经门卫确认后

放行并计时收费。

该系统的分析并不复杂，可将系统分为如下 3 个模块：

(1) 教工车辆校内注册模块，负责车辆的校内注册和取消注册。

(2) 车牌识别模块与查询模块，判别是否为校内注册车辆并确定收费。

(3) 后台模块，统计收费、车辆在校时间等管理需要的信息。

这三个子系统中，最难实现的是汽车牌照的自动识别，它涉及图像采集、数字分割、模式识别算法。在项目开发过程中，可以通过预留接口、采取封装技术，将汽车牌照的计算机识别过程独立出来、甚至委托第三方开发。

1.1.4 约简与结构化设计

计算思维是通过约简、嵌入、转化和仿真等方法，把一个困难的问题阐释成如何求解它的思维方法。结构化程序设计中的模块化就是一种典型的约简过程。程序员设计系统时，经常使用各种函数如：正弦函数 sin，求平方根函数 sqrt 等，可以从中体会到这些函数给我们带来的便利。如果我们正在处理的问题，有某个算法需要经常被调用，就应该学习如何将其优化固定，使其成为系统搭建的一个构件，随时在该出现的地方被使用，使程序的逻辑变得简单。

结构化程序设计带来的好处是容易将一个系统中被多个模块共同使用的模块分离出来。以 1.1.3 节中提到的车牌自动识别系统为例，系统中提到的三个模块都需要操作数据库，因此可以将负责数据库链接的模块独立出来，供各模块调用，其体系构架如图 1.1.1 所示，这种方式提高了系统的可维护性。

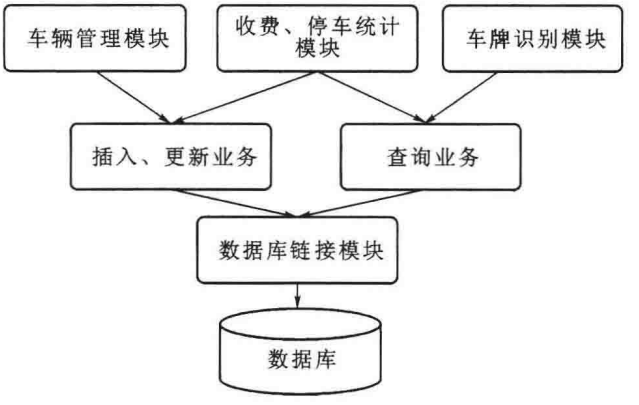

图 1.1.1 学校车辆管理系统的体系构架

现代计算机程序设计语言都强调程序代码的结构化，不只体现在函数模块的规划抽象上，也体现在程序代码的编写上，例如在 C++中，任何一个循环，都强调循环体语句组向右缩进，以表达一种从属关系，增加可阅读性。虽然书写时，循环体语句不向右缩进也不违反语法，但在软件设计行业中，不遵循这样的书写格式，则意味着你不是一个合格的程序员。而 Python 语言则要求更严格，如果不遵循这样的结构化格式，系统即认为编码非法。

1.1.5 面向过程与面向对象

1. 面向过程的程序设计

在软件发展的初期，人们习惯于面向过程的程序编写方式，其特点是将问题看作一系列

需要完成的任务，将完成任务的要素抽象归纳为多个函数。这种程序设计方式将数据与处理数据的程序分开，这符合人们刚刚进入编写程序领域、处理简单问题的习惯。这符合冯·诺依曼计算机顺序化处理逻辑，易被初学程序设计者接受。

然而，面向过程的程序设计方法在处理较大规模的系统时，很难控制系统的复杂度。在多函数程序中，由于采用了数据与函数分离的方式，整个程序流程中，许多重要的数据被放置在全局数据区，由程序员控制数据在不同函数间传递，这也被称为数据的封装性不够。由于每个函数都可以有它们自己的局部数据，所以完全存在函数内部对全局数据进行无意修改的可能，导致程序的正确性不易保证。

2. 面向对象的程序设计

1967 年挪威计算中心的 Kisten Nygaard 和 Ole Johan Dahl 开发了 Simula67 语言，在这种语言中引入了数据抽象和类的概念，提供了比函数更高一级的抽象和封装。它被认为是第一个面向对象的程序设计语言。之后，美国 Palo Alto 研究中心的 Alan Kay 所在的研究小组开发出的 Smalltalk-80，被认为是最纯正的面向对象语言，它对后来出现的面向对象语言，如 Object-C、C++等，都产生了深远的影响。

面向对象程序设计，将世界万物看作对象，并对同种对象的共同性质进行抽象，形成一种类别。在程序的实际设计中，将对象作为程序的基本要素，而对象则是行为和属性的结合体。用计算机程序领域中的术语来说，是将数据（对象的属性）和代码（对象的行为）结合为一个整体。这种包装带来的巨大优势是：代码很容易被继承和复用；每个类都可以被当作一个坚固的构件，用于不同系统的搭建，因而非常符合工程建造领域快速搭建的思维方式，系统的复杂度得以很好地分散控制，适用于大型体系的建设。

面向对象程序设计的主要特征包括：

（1）封装性

封装使数据和处理数据的程序组装为一个整体，从而可以实现独立性很强的模块。通过封装技术，很容易将类的使用者和设计者分开，实现信息的隐藏，使得使用者只能见到对象的外部特性，不必知晓行为实现的细节，只须用设计者提供的方法来访问该对象。

（2）继承性

继承性指子类自动拥有父类的数据和方法的机制。一个类可以直接继承其父类的部分或全部特征，同时可进行修改和扩充。其带来的巨大利益是，在支持系统可重复性的基础上，进一步提升了系统的可扩充性。

（3）多态性

多态，是指父类中定义的方法，被不同的子类继承之后，每个子类通过扩充，拥有不同的行为。多态性的实现受到继承性的支持，通常的设计方法是：利用类继承的层次关系，把具有通用功能的特性存放在类层次中尽可能高的地方，而将实现这一功能的不同方法置于较低层次。这样，在这些低层生成的对象就能给通用的消息以不同的响应。

1.2 计算思维案例

为加强对计算思维的理解，本节以大学基础课程教学中的一些案例的计算机求解为例，希望将计算思维中抽象的概念通过实例进一步阐述清晰，以易于理解。

1.2.1 梯形积分逼近

程序：梯形积分 e-x

高等数学中求解函数积分时，通过将积分区间分割成无限小的梯形来求解，如图1.2.1所示：

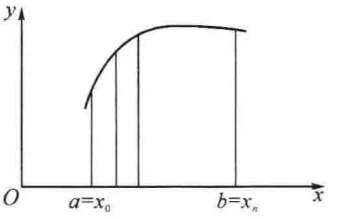

图 1.2.1 梯形法求积分

在梯形无限小时，所有小梯形的面积之和，就是函数在区间[a,b]中的积分。在用计算机求解该类问题时，摆在我们面前的一个问题是：小梯形区间小到什么时候才算无限小？因为用计算机计算时，不可能像高等数学理论中讲的那样无限小。如果这样，这个问题就算不完了。那如何使这样的问题变得可编程计算？此时可以利用容错概念，即将梯形的无限小定义为当梯形小到一定程度，再小下去时，求得的积分变化几乎可以忽略不计。此时，虽然求得的积分值不是精确解，但对计算求解而言，已经可以满足要求的精度。

在这样的指导原则下，梯形求解积分就变得可计算。由此，可以设计这样的思路：

先将积分区间分成 n 份，求得一个积分值 S_1。

将积分区间份数扩大为原来的两倍，即 $2n$ 份，再求一个积分值 S_2。

如果 S_1-S_2 的绝对值足够小，则求解过程结束，否则再将区间分成 $4n$ 份，如此反复增加积分区间的份数，达到求解的目的。

上述思考得到的求解方法，由于事先难以得知划分小区间的最佳份数，所以，需要一个自动化的循环迭代过程来求解。通过循环，不断将原来的小区间份数变成原来的 2 倍，并比较前后 2 次所求积分的差值。因此使用循环的求解过程可以描述如下：

将区间看作一个大梯形，区间计算面积记录到 T_2，小区间个数 n=1
令精度为 eps=10^{-6}
循环开始
令 $T_1=T_2$
令 n=2n
计算新的面积 T_2
如果 T_1-T_2 的绝对值小于 eps，计算结束
否则转到循环开始位置

这种循环迭代配合精度的策略，在计算求解的过程中经常使用。例如求解 e^{-x} 的近似

值，sin(x)的近似值等问题，将它们用级数展开后，可以分别得到如下的公式：

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

$$\sin(x) = \frac{x}{1} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + (-1)^{n-1}\frac{x^{2n-1}}{(2n-1)!}$$

该类问题，通过分析其计算公式，在给定精度的条件下，都可以通过循环的方式实现求解。

1.2.2 插值与离散积分

科学试验中，往往难以明确待测量与其响应间的函数关系。但通过试验测得一组试验点，却是不难做到的事，如图 1.2.2 所示。

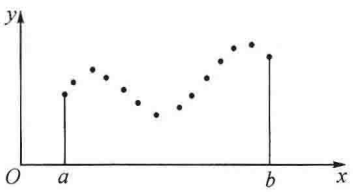

图 1.2.2 离散试验点的积分

此时再提出积分的任务，没有函数形式，该如何实现？

简单思考后，我们可以选择 3 种方案：

(1) 直接用试验点形成的小梯形进行积分；

(2) 通过曲线拟合等手段，得到所有试验点的拟合函数，然后再用 1.2.1 节提供的策略求解；

(3) 通过将试验点逐段拟合成不同的函数，对于落在 2 个测量点之间的小梯形，通过该段的拟合函数求解。

在实际应用中，采用 3 点拉格朗日分段模拟插值的方法应用很普遍，可以很好地解决离散试验的积分问题，又被称为离散积分。这种问题转化手段，也是问题求解的计算思维方式。

1.2.3 计算模拟

科学研究中经常遇到一些用试验和理论思维难以解决的复杂问题，这其中有部分问题可以通过计算机模拟大大加快理论验证。例如材料领域晶体的随机生长、航天器飞行的空气动力学计算、核反应体系中子碰撞过程，等等。这类问题求解中的模拟手段也是计算思维的体现。

图 1.2.3 是利用计算机模拟得到的多晶材料晶粒生长情况，可为材料研究提供新的方法和重要依据。

下面我们以一个简单的求解 $f(x) = e^{-x}$ 函数在区间[0, 1]的积分为例，来谈谈随机模拟求解的实现方式。虽然该例用高等数学的理论很容易求解，但这里我们采用一种完全不同的思路求解，以扩展读者计算求解的思路。

先观察一下图 1.2.4，图中曲线即为 $f(x) = e^{-x}$。如果将此图看作一幅数字化的图片，则可以将积分看作是曲线下面的点占总点数的百分比。

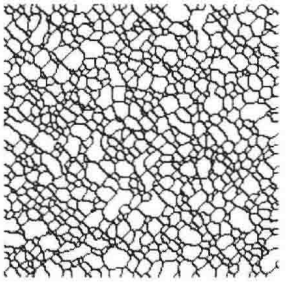

图 1.2.3　多晶材料晶粒生长的计算机模拟

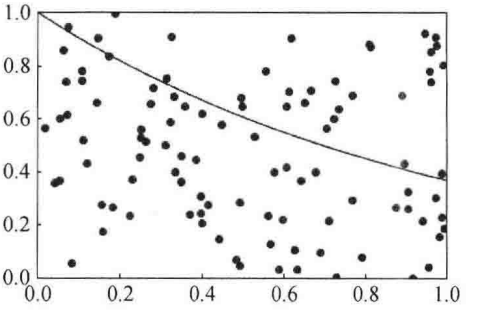

图 1.2.4　计算机模拟求解定积分

这是一个特殊设计的例子，函数及自变量所在的区间都在[0,1]范围内，这个矩形的面积为 1，因而曲线下面的面积(积分)，正好是曲线下面的所有点占矩形全部点的百分比。

为使问题可模拟求解，可设定如下的随机过程：不断在矩形($0 \leqslant x \leqslant 1$, $0 \leqslant y \leqslant 1$)中产生均匀分布的点，这产生了两个服从[0, 1]均匀分布的随机量 X、Y，且两者相互独立。定义事件

$A = \{Y \leqslant f(x)\}$，然后计算 A 事件发生的概率，当在图 1.2.4 中随机产生大量的点时，A 事件的概率就接近真值，此时，A 的概率就是问题的近似解。

故此，设计如下的求解伪码：

设定产生的所有的点数 N
产生 N 个随机小数，记录到 $X(x_1, x_2, \cdots, x_n)$中
再产生 N 个随机小数，记录到 $Y(y_1, y_2, \cdots, y_n)$中
曲线下面的点数 count 初值=0
令 i 从 1 到 N 循环
根据 x_i，计算函数值 $f(x) = e^{-xi}$
如果 $f(x) \leqslant y_i$ ，则 count=count+1
计算概率 count/N

对应的 Python 程序如下：

程序：模拟求积分

```
import random as R
import matplotlib.pyplot as plt
import math
points=100 #产生点数，
below=0
x2=[]
y2=[]
for i in range(points):
    x, y=R.random(), R.random()
```

```
        x2.append(x)
        y2.append(y)
        if math.exp(-x)>y:
            below +=1
    pi=below/points
    print(pi)
```

以下是用 Python 编写的程序在不同 N 值时的计算结果，见表 1.2.1：

表 1.2.1　蒙特卡洛模拟计算定积分

随机点数	10^3	10^4	10^5	10^6	真值
积分值	0.607	0.629 3	0.634 28	0.632 749	0.632 12

从计算结果可以看到，随着随机点数的增加，结果逐步逼近真值。

思考题

1. 请举例简单描述你对计算思维中的约简、嵌入、转化等方法的理解。

2. 请举例描述计算思维中如何通过抽象控制复杂度。

3. 通过梯形积分案例，谈谈你对计算思维迭代自动化、可计算求解的理解。

4. 通过搜索引擎查找通过随机过程求圆周率值的过程，谈谈计算机随机过程模拟的作用。

5. 已知 $\sin(x)=\frac{x}{1}-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\cdots+(-1)^{n-1}\frac{x^{2n-1}}{(2n-1)!}$，请从约简、容错及迭代等方面谈谈该问题应如何求解。