

21 世纪高等学校计算机**基础**实用规划教材

算法与程序设计基础

(Python版)

吴 萍 主编

朱晴婷 蒲 鹏 副主编

刁庆霖 裘奋华 等 编著



清华大学出版社

内容简介

本书是“21世纪高等学校计算机基础实用规划教材”之一，旨在为高等院校计算机专业及相关专业的大一、大二学生提供一本基础教材。本书以Python语言为载体，系统地介绍了算法与程序设计的基本概念、方法和技巧。全书共分10章，主要内容包括：Python语言基础、数据类型、运算符、表达式、语句、函数、模块、文件操作、异常处理、面向对象编程等。本书注重理论与实践相结合，通过大量的例题和习题，帮助学生掌握编程技能，提高解决问题的能力。本书可作为高等院校计算机专业及相关专业的教材，也可供从事计算机工作的工程技术人员参考。

算法与程序设计基础

(Python版)

吴萍 主编

朱晴婷 蒲鹏 副主编

刁庆霖 裴奋华 等 编著

清华大学出版社
北京

内 容 简 介

理论、思维训练与实践相结合是本书的特色。本书共分为8章,通过将算法与程序设计的基本概念与某种程序设计语言(以Python为主)结合使学生理解计算思维的概念,了解算法与程序的关系,能够进行较为简单而经典的算法设计、评价算法的性能与效率,并能利用程序设计语言进行简单的程序开发;培养学生利用计算机解决与专业、科研、社会需要密切相关的实际问题的能力和基本创新精神,以适应信息化社会的要求、拓宽专业生存空间,使其在后续专业课程的学习和未来的工作中较长期地受益。

本书是高等院校非计算机专业的计算机技术应用课程的教材,适用于文理工科各专业学生,也可作为初学者的入门书籍。

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

算法与程序设计基础:Python版/吴萍主编.--北京:清华大学出版社,2015

21世纪高等学校计算机基础实用规划教材

ISBN 978-7-302-39021-3

I. ①算… II. ①吴… III. ①电子计算机—算法理论—高等学校—教材 ②程序设计—高等学校—教材 ③软件工具—程序设计—高等学校—教材 IV. ①TP301.6 ②TP311.1 ③TP311.56

中国版本图书馆CIP数据核字(2015)第017205号

责任编辑:付弘宇 薛 阳

封面设计:何凤霞

责任校对:李建庄

责任印制:王静怡

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 装 者:河北新华第一印刷有限责任公司

经 销:全国新华书店

开 本:185mm×260mm

印 张:18.5

字 数:460千字

版 次:2015年2月第1版

印 次:2015年2月第1次印刷

印 数:1~2000

定 价:39.00元

产品编号:061819-01

出版说明

随着我国改革开放的进一步深化,高等教育也得到了快速发展,各地高校紧密结合地方经济建设发展需要,科学运用市场调节机制,加大了使用信息科学等现代科学技术提升、改造传统学科专业的投入力度,通过教育改革合理调整和配置了教育资源,优化了传统学科专业,积极为地方经济建设输送人才,为我国经济社会的快速、健康和可持续发展以及高等教育自身的改革发展做出了巨大贡献。但是,高等教育质量还需要进一步提高以适应经济社会发展的需要,不少高校的专业设置和结构不尽合理,教师队伍整体素质亟待提高,人才培养模式、教学内容和教学方法需要进一步转变,学生的实践能力和创新精神亟待加强。

教育部一直十分重视高等教育质量工作。2007年1月,教育部下发了《关于实施高等学校本科教学质量与教学改革工程的意见》,计划实施“高等学校本科教学质量与教学改革工程(简称‘质量工程’)”,通过专业结构调整、课程教材建设、实践教学改革、教学团队建设等多项内容,进一步深化高等学校教学改革,提高人才培养的能力和水平,更好地满足经济社会发展对高素质人才的需要。在贯彻和落实教育部“质量工程”的过程中,各地高校发挥师资力量强、办学经验丰富、教学资源充裕等优势,对其特色专业及特色课程(群)加以规划、整理和总结,更新教学内容、改革课程体系,建设了一大批内容新、体系新、方法新、手段新的特色课程。在此基础上,经教育部相关教学指导委员会专家的指导和建议,清华大学出版社在多个领域精选各高校的特色课程,分别规划出版系列教材,以配合“质量工程”的实施,满足各高校教学质量和教学改革的需要。

本系列教材立足于计算机公共课程领域,以公共基础课为主、专业基础课为辅,横向满足高校多层次教学的需要。在规划过程中体现了如下一些基本原则和特点。

(1) 面向多层次、多学科专业,强调计算机在各专业中的应用。教材内容坚持基本理论适度,反映各层次对基本理论和原理的需求,同时加强实践和应用环节。

(2) 反映教学需要,促进教学发展。教材要适应多样化的教学需要,正确把握教学内容和课程体系的改革方向,在选择教材内容和编写体系时注意体现素质教育、创新能力与实践能力的培养,为学生的知识、能力、素质协调发展创造条件。

(3) 实施精品战略,突出重点,保证质量。规划教材把重点放在公共基础课和专业基础课的教材建设上;特别注意选择并安排一部分原来基础比较好的优秀教材或讲义修订再版,逐步形成精品教材;提倡并鼓励编写体现教学质量和教学改革成果的教材。

(4) 主张一纲多本,合理配套。基础课和专业基础课教材配套,同一门课程可以有针对不同层次、面向不同专业的多本具有各自内容特点的教材。处理好教材统一性与多样化,基本教材与辅助教材、教学参考书,文字教材与软件教材的关系,实现教材系列资源配套。

(5) 依靠专家,择优选用。在制定教材规划时依靠各课程专家在调查研究本课程教材

建设现状的基础上提出规划选题。在落实主编人选时,要引入竞争机制,通过申报、评审确定主题。书稿完成后要认真实行审稿程序,确保出书质量。

繁荣教材出版事业,提高教材质量的关键是教师。建立一支高水平教材编写梯队才能保证教材的编写质量和建设力度,希望有志于教材建设的教师能够加入到我们的编写队伍中来。

21 世纪高等学校计算机基础实用规划教材

联系人:魏江江 weijj@tup.tsinghua.edu.cn

前言

为了适应信息和计算技术的发展,切实满足社会各个领域对计算机应用人才不断增长的需求。本书设计了“算法与程序设计基础”的通识课程方案,力求融入计算思维的思想,将多年来计算机学科所形成的解决问题的思维模式和方法渗透到各个学科。与传统的程序设计类教材不同,本书选择较容易上手的 Python 语言,着重介绍分析问题和解决问题的方法和思路,通过对不同解决方案的分析比较,让学生掌握选取优化方案并予以实现的理论方法和实际应用能力。

本教材编写具有以下特点。

1. 难点和重点安排合理

教材中的内容编排凝聚了作者多年的教学经验与体会,并在章节的篇幅和安排上为教师提供了讲解内容和时间安排上的灵活性。扩展部分使有能力的读者可以更上一层楼,并把本书当作一个有价值的参考资源。

2. 讲解深刻

对一些重难点知识,学生不仅要知其然,还需要知其所以然。因此教材中会为教师和学生剖析其本质,让学生能够从根本上理解、掌握并灵活运用这些知识。

3. 实用性强

教材中提供了大量针对性的实例,同时编程中要注意什么? 如何阅读出错提示? 出现问题如何解决? 教材中都会一一讲解,带领学生迅速掌握编程的全过程。

4. 涵盖了算法和程序设计的较为核心的内容

选择了经典和应用广泛的各类算法,并结合程序设计的思想和方法,让学生能够通过循序渐进的程序设计过程了解计算的魔力,掌握求解问题的方法,进而融入后续的学习和今后的生活和工作中。

本书由吴萍负责全书的统稿。第 1 章由朱敏、陈志云、蒲鹏执笔,第 2 章、第 6 章由周力执笔,第 3 章由朱晴婷执笔,第 4 章由蒲鹏执笔,第 5 章由朱晴婷、裘奋华执笔,第 7 章由吴萍执笔,第 8 章由刁庆霖执笔。附录 A 由各章编写者提供,附录 B 由郑凯选编。

由于时间仓促和作者水平有限,书中难免有不妥之处,恳请广大读者批评指正。

本书的配套课件等资源可以从清华大学出版社网站 www.tup.com.cn 下载,下载或使用中的相关问题,请联系 fuhy@tup.tsinghua.edu.cn。

编 者

2015 年 1 月

于华东师范大学

目 录

第 1 章 程序设计与计算思维	1
1.1 程序设计与计算机语言	1
1.1.1 程序设计	1
1.1.2 设计步骤	1
1.1.3 程序设计分类	2
1.1.4 基本规范	2
1.1.5 计算机语言	2
1.1.6 语言分类	2
1.2 计算机语言与计算思维的关系	3
1.2.1 思维与计算思维	3
1.2.2 计算思维与计算科学的关系	4
1.2.3 计算思维与程序设计语言的关系	5
1.3 初识 Python 语言	6
1.3.1 Python 语言概述	6
1.3.2 Python 语言的应用	8
1.3.3 编辑与运行环境	9
1.4 Python 的帮助系统	14
1.4.1 关于 Python 帮助系统	14
1.4.2 使用 Python 帮助系统	14
1.5 本章小结	15
1.6 习题与思考	16
1.7 实验 Python 的安装和运行环境	16
1.7.1 实验目标	16
1.7.2 实验范例	16
1.7.3 实验内容	17
第 2 章 算法概述	18
2.1 计算机程序与算法	18
2.1.1 计算机求解问题的过程	18
2.1.2 算法的定义及其发展历史	19

2.1.3	算法的基本性质	20
2.1.4	算法的评价	20
2.2	算法的描述	20
2.2.1	用自然语言或伪代码描述算法	21
2.2.2	用流程图描述算法	21
2.2.3	使用电脑软件绘制流程图	23
2.3	常用算法简介	25
2.3.1	枚举算法	25
2.3.2	迭代算法	28
2.3.3	贪心算法	31
2.4	本章小结	34
2.5	习题与思考	35
2.6	实验 算法描述和绘制流程图	36
2.6.1	实验目标	36
2.6.2	实验范例	36
2.6.3	实验内容	39
第3章	数据表示和计算	40
3.1	数据和数据类型的概念	40
3.1.1	数据的表示	40
3.1.2	数据类型的概念	41
3.1.3	Python 的内置类型	42
3.1.4	常量和变量	42
3.1.5	Python 的动态类型	46
3.2	数值数据的表示与计算	47
3.2.1	数值数据的常量表示	47
3.2.2	数值数据的计算	48
3.2.3	系统函数	54
3.3	文本数据的表示和操作	56
3.3.1	文本的表示	56
3.3.2	字符串类型数据的基本计算	58
3.3.3	str 对象的方法	59
3.4	批量数据表示与操作	61
3.4.1	批量数据的构造	61
3.4.2	元组和列表	62
3.4.3	集合和字典	67
3.5	本章小结	76
3.6	习题与思考	78
3.7	实验 数据的表示和操作	79

3.7.1	实验目标	79
3.7.2	实验范例	80
3.7.3	实验内容	87
第 4 章	基本控制结构的程序设计	89
4.1	用 Python 实现顺序结构程序	90
4.2	用 Python 实现分支结构程序	90
4.2.1	Python 简单分支	91
4.2.2	Python 双分支	91
4.2.3	Python 分支嵌套	92
4.2.4	Python 多分支结构	93
4.3	用 Python 实现循环结构程序	95
4.3.1	Python 的 for 循环语句	96
4.3.2	Python 的 range() 函数	98
4.3.3	Python 的 while 循环结构	100
4.3.4	Python 的 break、continue 和 pass 语句	100
4.3.5	循环结构应用	102
4.4	字符串数据操作	105
4.4.1	字符串和 List 数据的相互转换	105
4.4.2	字符查找	106
4.4.3	字符串遍历	107
4.4.4	字符串截取	108
4.5	本章小结	109
4.6	习题与思考	109
4.7	实验 基本控制结构	110
4.7.1	实验目标	110
4.7.2	实验范例	110
4.7.3	实验内容	114
第 5 章	数据的输入和输出	116
5.1	人-机交互的意义、方法	116
5.1.1	标准输入输出	116
5.1.2	文件输入输出	117
5.2	标准输入输出程序	118
5.2.1	标准输入函数	118
5.2.2	标准输出函数	121
5.2.3	输入输出重定向	123
5.3	文件输入输出程序	125
5.3.1	文件的基本操作	125

5.3.2	文件输入输出程序的实现	127
5.4	异常	136
5.4.1	简介	136
5.4.2	异常处理	137
5.5	本章小结	147
5.6	习题与思考	148
5.7	实验	149
5.7.1	标准输入输出	149
5.7.2	文件输入输出	153
5.7.3	异常处理	161
第 6 章	函数与模块	169
6.1	函数的基本概念	169
6.2	在 Python 语言中定义和使用函数	170
6.2.1	函数定义和调用	170
6.2.2	函数间的数据联系	174
6.3	函数应用	178
6.4	模块和 Python 标准库	183
6.4.1	模块	183
6.4.2	Python 标准库	186
6.5	本章小结	193
6.6	习题与思考	194
6.7	实验 函数和模块的使用	195
6.7.1	实验目标	195
6.7.2	实验范例	195
6.7.3	实验内容	201
第 7 章	算法分析与设计	203
7.1	算法性能分析	203
7.1.1	性能分析的重要性	203
7.1.2	举例说明算法的时间性能分析与量度	203
7.1.3	计算时间的渐近估计表示	204
7.2	查找问题	206
7.2.1	查找最大数最小数	206
7.2.2	查找特定数	207
7.3	排序问题	210
7.3.1	冒泡排序	210
7.3.2	选择排序	212
7.3.3	插入排序	213

7.3.4	快排-引入递归和分治概念	214
7.4	递归和分治的思想	216
7.4.1	递归概念	216
7.4.2	递归调用方法与实现	218
7.4.3	分治概念	219
7.5	本章小结	220
7.6	习题与思考	221
7.7	实验 算法实现与性能分析	222
7.7.1	实验目标	222
7.7.2	实验范例	222
7.7.3	实验内容	227
第8章	面向对象思想简介	230
8.1	面向对象思想	230
8.1.1	面向对象思想概述	230
8.1.2	面向对象中的基本概念	231
8.2	Python 中的类和对象	232
8.2.1	类的定义和对象的创建	233
8.2.2	类的继承	235
8.3	面向对象思想应用——图形界面编程	238
8.3.1	图形用户界面	238
8.3.2	Python 图形框架	239
8.4	本章小结	246
8.5	习题与思考	246
8.6	实验	247
8.6.1	Python 面向对象编程初步	247
8.6.2	Python 图形界面编程初步	252
附录A	习题解答	256
附录B	Python 习题选编	266
参考文献		283

第1章

程序设计与计算思维

计算机虽然是人类制造出来的,但是它还是依托电子器件而存在的,离不开它本身固有的电子属性;所以在使用程序设计指导计算机工作的时候,就必须建立一种特殊的能适合计算机思考的思维方法。

学习程序设计语言,不仅要掌握其语法规则,更重要的是从思维训练入手,学会分析问题,从实际问题中抽象求解方法的能力。

1.1 程序设计与计算机语言

1.1.1 程序设计

程序设计(Programming)是给出解决特定问题程序的过程,是设计、编制、调试程序的方法和过程。它是目标明确的智力活动,是软件构造活动中的重要组成部分。它以某种程序设计语言为工具,给出这种语言下的程序。程序设计通常分为问题分析、算法设计、程序编写、程序运行、结果分析和文档编写等阶段。专业的程序设计人员常被称为程序员。

1.1.2 设计步骤

(1) 问题分析。

对于接受的任务进行认真的分析,研究给定的条件,分析最后应达到的目标,找出解决问题的规律,选择解决问题的方法,完成实际问题。

(2) 算法设计。

设计出解决问题的方法和具体步骤。

(3) 程序编写。

根据设计的算法,选择一种程序设计高级语言编写出源程序,并通过测试。

(4) 对源程序进行编辑、编译和连接。

(5) 运行程序,分析结果。

运行可执行程序,得到运行结果,并对结果进行分析,看它是否符合要求。如不符合要求,就需要进行修改、再测试、再运行,直至结果正确。

(6) 文档编写。

文档编写内容应包括程序名称、程序功能、运行环境、程序的装入和启动、程序的输入、输出数据,以及使用注意事项等。

1.1.3 程序设计分类

按照结构性质分类,程序设计有结构化程序设计与非结构化程序设计之分。结构化程序设计是指具有结构性的程序设计方法与过程,它具有由基本结构构成复杂结构的层次性,非结构化程序设计反之。

按照用户的要求分类,程序设计有过程式程序设计与非过程式程序设计之分。过程式程序设计是指使用过程式程序设计语言的程序设计,非过程式程序设计指非过程式程序设计语言的程序设计。

按照程序设计的成分性质分类,程序设计有顺序程序设计、并发程序设计、并行程序设计、分布式程序设计之分。

按照程序设计风格分类,程序设计有逻辑式程序设计、函数式程序设计、对象式程序设计之分。

1.1.4 基本规范

程序设计规范是进行程序设计的具体规定。程序设计是软件开发工作的重要部分,而软件开发是工程性工作,所以必须有规范,才能保证程序设计的质量。

1.1.5 计算机语言

语言分为自然语言与人工语言两大类。自然语言是人类在自身发展的过程中形成的语言,是人与人之间传递信息的媒介。人工语言指的是人们为了某种目的而自行设计的语言。计算机语言就是人工语言的一种。计算机语言是人与计算机之间传递信息的媒介。为了使电子计算机能进行各种工作,就需要有一套用以编写计算机程序的数字、字符和语法规则,由这些数字、字符和语法规则组成计算机的各种指令(或各种语句)就是计算机能接受的语言。

1.1.6 语言分类

计算机语言的种类很多,可以分成机器语言、汇编语言、高级语言三大类。

机器语言是表示成数码形式的机器基本指令集,或者是操作码经过符号化的基本指令集。汇编语言是机器语言中地址部分符号化的结果,或进一步包括宏构造。高级语言的表示方法更接近于待解问题的表示方法,其特点是在一定程度上与具体机器无关,易学、易用、易维护。

程序设计语言按照用户的要求可分为过程式语言和非过程式语言。过程式语言的主要特征是,用户可以指明一系列可顺序执行的运算,以表示相应的计算过程,如 FORTRAN、COBOL、PASCAL 等。

按照应用范围可分为通用语言与专用语言,如 FORTRAN、COLBAL、PASCAL、C 语言等都是通用语言。目标单一的语言称为专用语言,如 APT 等。

按照使用方式可分为交互式语言和非交互式语言。具有反映人机交互作用的语言称为交互式语言,如 BASIC 等。不反映人机交互作用的语言称为非交互式语言,如 FORTRAN、COBOL、ALGOL69、PASCAL、C 语言等都是非交互式语言。

按照成分性质可分为顺序语言、并发语言和分布语言。只含顺序成分的语言称为顺序语言,如 FORTRAN、C 语言等。含有并发成分的语言称为并发语言,如 PASCAL、Modula 和 Ada 等。

程序设计语言还可分为面向对象语言和面向过程语言,面向对象的例如: C++/C#/Java 等,面向过程的例如: Free Pascal/C 语言等。

1.2 计算机语言与计算思维的关系

思维和语言是紧密相连的社会现象,思维与语言之间究竟是什么关系,历来是语言学家、哲学家、心理学家、人类学家、生物学家、逻辑学家都十分关心的问题。

1.2.1 思维与计算思维

思维是什么?哲学上的思维通常指人脑对客观事物间接和概括的反映,是认识的理性阶段。中国、俄罗斯和东欧各国学术界普遍认为,思维是人借助语言实现的对客观事物概括、间接的反映,是反映对象本质和规律的认识过程。

计算思维(Computational Thinking)是运用计算机科学的基础概念进行问题求解、系统设计,以及人类行为理解等涵盖计算机科学之广度的一系列思维活动。这个概念是由美国卡内基·梅隆(Carnegie Mellon)大学计算机科学教授周以真教授提出的,要理解它,首先得搞清楚什么是问题求解、系统设计和理解人类行为。

1. 问题求解

在每天的生活中,都会碰到很多问题,面对不同的问题,需要不断地进行思考和选择。而解决问题的方法也是多种多样的,不同的方法或许都能达到相同的目的,但过程和效率却可能大相径庭,最终如何解决问题,取决于对问题的分析和对所选方案的周密考虑。

【讨论】 开学了,作为学生会主席,你在考虑举行一场迎新晚会来迎接新同学的到来,那么,晚会该如何组织?如果申请的费用有限,如何在预算范围内,将晚会办得更好些?

这时,你也许会考虑这样一些问题:晚会的规模是多大?这取决于预算情况。会有多少老师和学生来参加?需要租借的场地也取决于此。还需要哪些资源?这些资源如何获取?把学生会干部们找来一起商量,根据大家的多个主意,权衡利弊,找到最佳解决方案。

问题求解的一般过程可以归纳为图 1-2-1。



图 1-2-1 问题求解的一般过程

确定问题:这是一个很重要的环节,如果把问题理解错了,后面的工作都是无效的。例如:要解决预算范围内的晚会规模,就不能只考虑将晚会办得华丽了。

分析问题:在这一环节,可以通过了解问题的相关背景知识,来进一步理解问题。如一

般开一个晚会碰到的问题有哪些? 场地、人员、节目、服装等, 目前已有了哪些资源, 还有哪些方面需要解决。预算费用如何在不同的需要方面进行划分, 如何在有限的费用前提下将晚会办得尽可能规模大些, 档次高些。

通过问题的分析提出方案, 在这个过程中, 有可能会有多种方案产生。可以在各种方案的基础上权衡利弊, 选择最适合的方案。

方案一旦确定后, 可以写出该方案的实施步骤, 步骤由简单到详细, 可以逐步细化, 直到容易实施。

在方案实施以后, 可以对方案的效果做出评价, 以便对今后类似问题的解决有所借鉴。

如果要用计算机来解决问题, 同样需要以上过程进行问题求解。利用计算手段求解问题的过程是: 首先要把实际的应用问题转换为数学问题, 然后建立模型、设计算法和编程实现, 最后在实际的计算机中运行并求解。前两步是抽象的过程, 后两步则是自动化的过程, 而计算思维的本质便是抽象与自动化, 这种思维方式是在计算科学的基础上逐渐建立起来的, 通过计算科学领域的学习, 可以使我们的计算思维得到提升, 并扩展到学习、生活的其他方面。

2. 系统设计

任何自然系统和社会系统都可视为一个动态演化系统, 演化伴随着物质、能量和信息的交换, 这种变换可以映射为符号变换, 使之能用计算机实现离散的符号处理。当动态演化系统抽象为离散符号系统后, 就可以采用形式化的规范来描述, 通过建立模型、设计算法和开发软件来揭示演化的规律, 实时控制系统的演化并自动执行。

例如: 计算机网络系统是一个复杂的系统, 为了实现该系统, 人们通过分层设计的方法, 将一个复杂的系统变成多个层次, 每个层次都有清晰的功能与上下层次之间的接口, 这样, 人们的精力可以放在对每个层次的设计上, 使复杂的问题变得简单。这种分层设计的方法经常会用在复杂系统的设计上, 通过不断分层, 直到将复杂问题简化为可以通过某种数学方法和模型将问题得到解决。这种思维方式不仅可以应用在计算科学领域, 也可推广到其他任何复杂系统的设计领域。

3. 人类行为

计算思维是基于可计算的手段, 以定量化的方式进行的思维过程。在人类的物理世界、精神世界和人工世界这三个世界中, 计算思维是建设人工世界所需要的主要思维方式。

利用计算手段来研究人类的行为, 可视为社会计算(Cyber-Society Computing), 即通过各种信息技术手段, 设计、实施和评估人与环境之间的交互。社会计算涉及人们的交互方式、社会群体的形态及其演化规律等问题。研究生命的起源与繁衍、理解人类的认知能力、了解人类与环境的交互以及国家的福利与安全等, 都属于社会计算的范畴。这些都与计算思维密切相关。

【讨论】 互联网的发展使用户数量激增, 大量用户的网络使用有什么规律可循吗? 怎样利用网络上的信息动态来把握商机? 通过讨论, 理解计算思维与人类行为的关系。

1.2.2 计算思维与计算科学的关系

计算思维虽然源自计算科学的发展, 拥有计算科学领域的许多特征, 看似与计算机相

关,但是计算思维本身并不是计算科学的专属,更不能认为只有与计算机相关的,才具有计算思维。实际上,即使没有计算机,计算思维也会逐步发展,甚至有些内容与计算机没有关联。但是,正是由于计算机的出现,给计算思维的研究和发展带来了根本性的变化。

由于计算机对信息和符号具有快速处理能力,使得许多原本只是理论上可以实现的过程变成了实际可以实现的过程。海量数据的处理、复杂系统的模拟和大型工程的组织,都可以借助计算机实现从想法到产品整个过程的自动化、精确化和可控化,大大拓展了人类认识世界和解决问题的能力,凸显了计算思维的重要性,推进了计算思维的形式、内容和表述的深入探索。在这样的背景下,作为人类思维活动中以形式化、程序化和机械化为特征的计算思维受到人们的重视,并且本身作为研究对象也被广泛和深入地研究着。

随着计算科学和技术的发展,社会进入了大数据时代,人们可以轻易地获得大量数据,并有能力在短时间内对完整的数据进行分析,从而获得新的信息。谁能利用好大数据,谁就获得了先机,计算思维的培养推动了大数据时代的创新。

什么是计算,什么是可计算,什么是并行计算,计算思维的这些性质得到了前所未有的彻底研究。由此不仅推进了计算机的发展,也推进了计算思维本身的发展。在这个过程中,一些属于计算思维的特点被逐步揭示出来,计算思维与理论思维、实验思维的差别越来越清晰化。计算思维的内容得到不断的丰富和发展,例如在对指令和数据的研究中,层次性、迭代表述、循环表述以及各种组织结构被明确提出来,这些研究成果也使计算思维的具体形式和表达方式更加清晰。从思维的角度看,计算科学主要研究计算思维的概念、方法和内容,并发展成为解决问题的一种思维方式,极大地推动了计算思维的发展。

1.2.3 计算思维与程序设计语言的关系

计算科学不是计算机编程。像计算机科学家那样去思维意味着远远不仅限于计算机编程,还要求能够在抽象的多个层面上思维。因此概念化的抽象思维不只是程序设计。

计算思维是人类求解问题的一条途径,但绝非要使人类像计算机那样去思考。计算机枯燥且沉闷,人类聪颖且富有想象力。是人类赋予了计算机激情,配置了计算机设备,人们就能用自己的智慧去解决那些计算机时代之前不敢尝试的问题,达到“只有想不到,没有做不到”的境界。计算机赋予人类强大的计算能力,人类应该更好地利用这种力量去解决各种需要大量计算的问题。

计算科学在本质上源自数学思维,因为像所有科学一样,其形式化基础是建筑在数学之上的。计算科学又从本质上源自工程思维,因为人们建造的是能够与实际世界互动的系统,基本计算机设备的限制迫使计算机科学家必须计算性地思考,而不是只是数学性地思考。构建虚拟世界的自由使人们能够超越物理世界的各种系统。数学和工程思维的互补与融合很好地体现在抽象、理论和设计三个学科形态上。

而程序设计语言(Programming Language)是用于书写计算机程序的语言,包含一组记号和一组规则,根据规则由记号构成的记号串的总体就是语言。程序设计语言有三个方面的因素,即语法、语义和语用。语法表示程序的结构或形式,亦即表示构成语言的各个记号之间的组合规律,但不涉及这些记号的特定含义,也不涉及使用者。语义表示程序的含义,亦即表示按照各种方法所表示的各个记号的特定含义,但不涉及使用者。语用表示程序与

使用者的关系。

学习了程序设计语言,掌握了其语法规则,但并不一定就能编写好的程序,这就像一个孩子,会用母语说话,但并不会用它来写小说。

实际上,程序设计语言是为人们解决问题服务的工具。真正要让计算机能工作,能为人们服务,要有好的软件,而软件的设计与制作虽然离不开程序设计语言,但更主要的是人们所提出的好的系统设计、好的问题解决方案,对方案的合理细化,并通过某种合适的程序设计语言的编程实现。

学习程序设计,虽然需要掌握程序设计语言的语法规则,但更重要的,是从训练思维入手,学会问题分析、思考解决方案,学会对各种现实问题的抽象,探寻计算机自动化的实现规律,这样才能在掌握程序设计语言语法的基础上,编制出好的程序,让计算机实现有效的功能,为人们提供服务。我们一定要把握好学习程序设计课程的机会,有意识地培养和提升自己的计算思维能力,为自己在专业领域的创新打下基础。

1.3 初识 Python 语言

1.3.1 Python 语言概述

1. 起源

为了让大家了解 Python 的起源,这里摘录一段 Leo Laporte(Twitter 网站创始人)和 Guido Van Rossum 的访谈录。Guido Van Rossum 从 2005 年开始在职于 Google。这个长相酷似埃尔顿·约翰逊的 Guido Van Rossum(图 1-3-1)虽然没有谱写出《风中之烛》那样的名曲,却在 1991 年的时候,给开源社区贡献了性能卓越的编程语言 Python。他也因此荣膺为 Python 之父。这个单词原意为蟒蛇,中文发音一般读作“派桑”。

Leo Laporte:

Google 是 Python 的一个大用户。首先,要问一个问题,是关于 Python 起源的,如果我说错了,请大家更正一下。据我所知,最初 Python 的设计是为了教学的目的,想创建一个用于学习和编程的语言,这么说对吗?

Guido Van Rossum:

Python 从一个叫 ABC 的语言继承了很多东西,而 ABC 这种语言在设计的时候就特别考虑到用于教学。在 20 世纪 80 年代末期,1989 年的时候,我觉得自己有必要创建一门新语言,我借鉴了 ABC 语言中所喜欢的特点,并将其中我不喜欢的东西用自己创新的或一些借鉴自别处的想法取而代之。我的目标是要建立一个为专业程序员使用的脚本语言,而这些专业程序员主要使用 C 语言和 Borne Shell 脚本语言作为他们的主要开发语言。Python 的位置大概是介于 C 和 Shell 语言之间的,所以我创建 Python 并没有明确的教学目的。因为我从 ABC 语言中借鉴了那么多,而 ABC 本身又有教学的目的在其中,所以我建立的语言也就很适合做教学语言。



图 1-3-1 Python 作者