Preface

This book is an introduction to computer science. It is intended for beginning CS majors or students from other fields who want a general introduction to computer science and computer programming.

The main focus is on the "big ideas" in computing. Contrary to what most people expect, computer science is much more than just programming. Computer science students learn to write programs, but the goal is to use programming skills to explore fundamental concepts and computational approaches to solving problems.

The distinguishing feature of this book is a set of tutorial exercises included in each chapter. The idea is to use an interactive programming language to provide an environment where students can type expressions, view the results (both in the terminal window and through algorithm animation), and run experiments that help them learn the concepts.

The analogy I like to make when I use this material in my own classes is to compare the tutorials to lab projects in an introductory chemistry class. There the instructor gives detailed instructions on the materials and methods, and students are expected to follow the instructions as precisely as possible. Students learn by observing as the experiment unfolds and writing lab reports that explain what happened.

For the "computational experiments" in *Explorations in Computing* students are shown the statements that create pieces of data and call functions that implement algorithms. An example of this sort of experiment is the section on the insertion sort algorithm. In an interactive Python session, students type statements to create lists of random numbers, display the lists as a series of bars on a canvas, and call the function that implements the sort algorithm. As the algorithm runs the bars are shuffled on the canvas. Later students are given a chance to write their own implementations of the algorithm, and the end of the chapter has several programming exercises based on simple iterative algorithms similar to insertion sort.

This book is a revised and updated version of *Explorations in Computing: An Introduction to Computer Science*, an introductory textbook I wrote in 2011 (and also published by Chapman & Hall/CRC Press). The new book has two major differences from the previous one. The first, most obvious, difference is the switch from Ruby to Python. Python has been widely adopted as the language of choice for first-year (CS1) computer science courses. By revising the lab software to use Python the hope is that students and instructors will find it easier to make a seamless transition from the introductory projects in this book to the deeper studies in later courses.

The second difference is that this new edition is also an introduction to Python programming. The primary emphasis is still on "computational thinking" and important concepts in computing, but along the way readers are presented with sufficient Python programming skills that they can implement their own programs to explore the ideas.

A Note for Students: How to Use This Book

You have no doubt heard the adage, "What you get out of a course depends on what you put into it." That saying is especially true for learning about computation with this book. You could simply read the book and hope to absorb the material, but to truly learn about computing you need to experience first-hand how the computer solves problems.

Each chapter features a tutorial project that helps you explore a particular problem and ways of solving it computationally. You are strongly encouraged to have your computer open as you read a chapter, and at the end of each section type in the Python statements exactly as they are shown in the tutorial exercise.

The "computational experiments" described in this book typically start by having you run complete programs that have already been implemented in PythonLabs, a set of modules written specifically for this book. These initial experiments include animations that illustrate the basic steps of the computation featured in that chapter. The remaining sections go into details that show how the key parts of the Python programs were implemented. As you work on the tutorials for these sections you will be writing your own programs, using the code in the book as the starting point.

The tutorials are designed so that you should be able to complete them in about the same amount of time you would spend on a lab project in a chemistry class. You could run through the tutorials in less time—about as fast as you can type, or if you are reading the book online, as fast as you can copy and paste—but you should take the time to make sure you understand what your computer is doing as you carry out each step in the tutorial.

At the end of each chapter you will find a set of exercises. These are similar to the questions you would find in a more traditional textbook and are designed to test your understanding of the material in that chapter. If you have completed the tutorial and understood what happened at each step along the way you should be able to answer these questions.

tl;dr

This book is designed to be read one section at a time. Each section has very few pages.

The projects at the end of each section are "tutorial' style exercises that tell you exactly what to type. Do these exercises. They will reinforce what you read.

T0. Type this statement to see a message from Python:

```
>>> print("hello")
hello
```

After you complete a chapter try your hand at the programming projects at the end of the chapter. The only way to learn how to program is to write programs. These exercises will get you started.

Notes for Instructors

The book is organized as a set of projects that gives students an opportunity to explore important ideas in computer science. In most cases, the main concepts are algorithms, and the projects are examples of how algorithms provide computational solutions to important problems.

An interactive programming language like Python provides a “computational workbench” where students can experiment with algorithms by typing expressions and immediately seeing the results. The language sets up an environment where students can run computations and explore the effects of changing parameters or modifying operations performed at key steps of the computation.

The topics presented in the book are outlined below. The general pattern for each chapter will be to first introduce the concept presented in that chapter; this introductory section will essentially be an essay that tries to make the case that the idea is interesting and worth understanding in more detail. The main part of the chapter will be the development, through a series of projects, of one or more algorithms that illustrate the idea and provide the student with a chance to experiment.

1 Introduction

The book starts with a general introduction to computation, focusing on the idea that computer science is not just about computers and is not just programming.

2 The Python Workbench

The second chapter is a brief introduction to Python and how it can be used as a “computational workbench” to set up experiments with computations. The project takes the students through the construction of a few simple programs, for example a function that converts temperature from Celsius to Fahrenheit. This chapter introduces the ideas of objects, variables, functions, and conditional execution.

3 The Sieve of Eratosthenes

This chapter introduces the first real algorithm studied in the book. It also introduces a few more practical techniques used later in the book: making lists of numbers and iterating over a list. The project starts with simple expressions involving integers, shows how to make a list of numbers, then how to selectively remove composite numbers, and leads finally to an algorithm that creates a complete list of prime numbers.

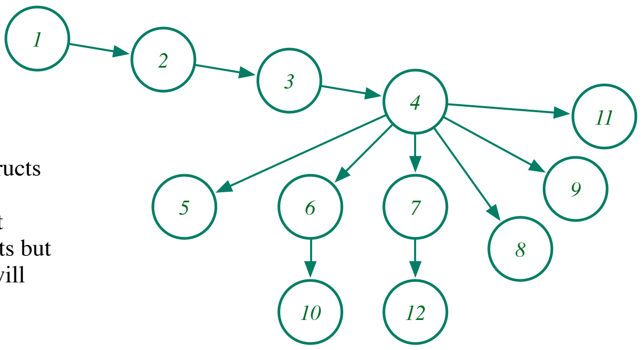
4 A Journey of a Thousand Miles

This chapter builds on the basic idea of iteration presented in the previous chapter. The project shows how iteration can be used to solve two common problems, searching and sorting, using linear search and insertion sort. An important idea in computing in this chapter is scalability, and students are introduced to $\mathcal{O}$ notation.

Chapter	CS Concepts	Python Programming
1. *Introduction*	Algorithms, computation	
2. *The Python Workbench*	Interactive computing	IDE, objects, functions, libraries, import, variables, assignment, numbers, strings, Boolean values, if statements
3. *The Sieve of Eratosthenes*	Iteration, containers, algorithm animation	Lists, list indices, for loops, incremental development
4. *A Journey of a Thousand Miles*	Linear search, insertion sort, scalability	While loops, Boolean operators
5. *Divide and Conquer*	Binary search, merge sort, exponential growth, $\log_2$	◆ Recursive functions
6. *Spam, Spam, Spam, Mail, and Spam*	Machine learning, Bayesian inference	File paths, text files, string processing, dictionaries
7. *Now for Something Completely Different*	Pseudorandom numbers, permutations, testing	mod operator, namespaces, classes, instance variables
8. *Bit by Bit*	Binary representations, trees, queues, parity bits, text compression	ASCII, Unicode, bitwise operators
9. *The War of the Words*	von Neumann architecture	
10. *I'd Like to Have an Argument, Please*	Natural language processing	String library, more string methods, regular expressions
11. *The Music of the Spheres*	Computer simulation, computational science	
12. *The Traveling Salesman*	Graphs, genetic algorithms	◆ Generators

◆ denotes extra-credit project

The first four chapters introduce fundamental programming constructs and should be presented in order. Chapters 5 through 7 also present important programming constructs but may be omitted if later projects will not use these constructs.



5 Divide and Conquer

The important idea in this chapter is that a more sophisticated strategy for solving a problem can lead to a more efficient computation. The project shows how binary search takes up to $\log_2 n$ steps instead of n, and merge sort takes at most $n \log_2 n$ steps instead of n^2.

6 Spam, Spam, Spam, Mail, and Spam

The algorithm used for experiments in this chapter is a Bayesian spam filter, a simple example of how "big data" can influence decision making. The chapter also introduces Python constructs for reading text files, Python dictionary (associative array) objects, and has additional exercises in string processing.

7 Now for Something Completely Different

The big idea in this chapter is randomness, and how random numbers can be used in a variety of algorithms, from games to scientific applications. There is an interesting paradox here: can we really generate random outputs from an algorithm? The answer is that random numbers generated by an algorithm are *pseudorandom*, and the project takes students through the steps in the development and testing of a pseudorandom number generator. This is the chapter where students are introduced to module structure in Python and get a chance to write their own simple class definitions.

8 Bit by Bit

The projects in this chapter are related to encoding data: using patterns of binary digits to encode numbers and letters, the number of bits required to encode a set of items, text compression with Huffman trees, and error correction with parity bits.

9 The War of the Words

This chapter introduces the important ideas that functions can also be encoded as a string of bits and that instructions are stored in a computer's memory along with data. The project uses the game of Corewar, which is a contest between two programs running in the same virtual machine; a program wins if it can write a halt instruction over the opponent's code. The projects lead the student through the phases of a processor's fetch-decode-execute cycle and emphasizes how a word that is a piece of data (the constant 0) for one program becomes an instruction (halt) for the other program.

10 I'd Like to Have an Argument, Please

The project in this chapter is based on a Python implementation of Joseph Weizenbaum's ELIZA program, and shows how very simple pattern matching rules can be used to transform input sentences, giving the illusion that the computer is carrying on a conversation. By the end of the chapter students will see how difficult natural language processing is, and how semantics and real-world knowledge are required for effective natural language understanding.

11 The Music of the Spheres

The big idea in this chapter is computer simulation. The project leads to an *ab initio* simulation of the motion of planets in the solar system. The chapter introduces issues related to verification and other topics in computer simulation.

12 The Traveling Salesman

The final chapter introduces the idea of intractable problems, building on ideas of scalability from earlier chapters. The project is based on a genetic algorithm and gives students the opportunity to explore probabilistic solutions. The tutorial has students use predefined code for Map and Tour classes to create random tours, so they can see how tours can be mutated and how collections of tours evolve until an optimal or near-optimal solution is obtained.

Pedagogical Considerations

This book was written primarily for CS0 courses, where the goal is to introduce the key concepts and, as much as possible, give a broad overview of the field. If augmented by additional material on Python programming and further programming exercises it could also be used for a more in-depth introduction as part of a CS1 course.

The projects have been used in courses at the University of Oregon. At UO we cover the first two chapters during the first week, but after that we spend between one and two weeks on the remaining topics chosen for that term. Lectures emphasize material from the first sections of a chapter, describing the problem and how it might be solved computationally, and explaining how that week's lab project gives some experience with the computation. Students have an option of attending a lab session, where an instructor is available to help them work through the material, but many students do the tutorials on their own. Live demonstrations of the tutorial projects, both in lecture and in lab sessions, have proved to be very effective.

At the end of each chapter there is a set of exercises that asks questions about issues raised in the chapter. After the students have completed the tutorial, they are asked to answer a selected set of questions and submit them as a "lab report" that gives them a chance to explain what they learned. Similar questions are given on exams.

Software, Documentation, and Lab Manuals

The software that accompanies this book is a set of modules named PythonLabs. PythonLabs is written exclusively in Python, using only libraries and modules that are part of the standard Python distribution. There is one Python module for each lab project. All of the modules have been collected into a single "egg," which makes it easy to install all the lab software in one step at the beginning of the term. The PythonLabs modules also include data files and sample Python code that students can copy and modify.

A Lab Manual with step-by-step instructions for installing Python and PythonLabs is available from the book website at `http://www.cs.uoregon.edu/eic`. There is a separate version of the manual for Windows XP, Mac OS X, and Linux. The manual also includes tips for editing programs and running commands in a terminal emulator.

Acknowledgments

This book is the result of many years of teaching introductory courses, and the material has evolved considerably over that time. The students and teaching assistants involved with these courses, and my colleagues at Oregon and elsewhere, have had a major influence on the topics and exercises presented here. I am very grateful for their comments and feedback.

I would like to thank Tom Cortina, Dilsun Kaynar, and Roger Dannenberg from Carnegie-Mellon University and the students in their Principles of Computing course who offered to "test drive" preliminary versions of this Python edition.

Randi Cohen, my editor at CRC Press, was instrumental in getting the first book published. I never would have considered writing a new version of *Explorations in Computing* without her patience and timely encouragement.

As always, I am eternally grateful for the support of my wife Leslie and my daughter Kathleen, who bears at least part of the responsibility for the Monty Python references in the section titles. I love you both.

John Conery
Eugene, Oregon

Website

The website for this book is at

```
http://www.cs.uoregon.edu/eic
```

The website has

- Copies of the lab manual (PDF documents that can be downloaded for free)
- Links to the latest versions of the PythonLabs software and documentation
- Errata and other news