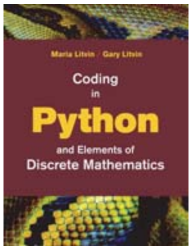
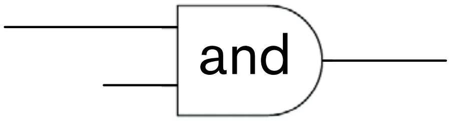


Coding in **Python**



Elements of Discrete Mathematics

Maria Litvin
Phillips Academy, Andover, Massachusetts

Gary Litvin
Skylight Software, Inc.

Skylight Publishing
Andover, Massachusetts

Skylight Publishing
9 Bartlet Street, Suite 70
Andover, MA 01810

web: http://www.skylit.com
e-mail: sales@skylit.com
support@skylit.com

Library of Congress Control Number: 2019905086

ISBN 978-0-9972528-4-2

1 2 3 4 5 6 7   23 22 21 20 19

Printed in the United States of America

To Henry and Esther, digital natives

Brief Contents

About the Authors

Maria Litvin has taught computer science and mathematics at Phillips Academy in Andover, Massachusetts, since 1987. Prior to joining Phillips Academy, Maria taught computer science at Boston University. Maria has co-authored several popular computer science textbooks — *C++ for You++: An Introduction to Programming and Computer Science* (1998), *Java Methods: Object-Oriented Programming and Data Structures* (2001-2015), *Be Prepared for the AP Computer Science Exam in Java*, and *250 Multiple-Choice Computer Science Questions* — as well as Continental Mathematics League (CML) computer science contests for elementary and middle school students. As a consultant for the College Board, Maria provides training for high school AP Computer Science teachers, and, since 2014, as a Code.org facilitator, Maria has trained hundreds of New England elementary school teachers in teaching computer science to children in grades K-5. Maria is the recipient of the 1999 Siemens Award for Advanced Placement for Mathematics, Science, and Technology for New England and the 2003 RadioShack National Teacher Award.

Gary Litvin is the co-author of *C++ for You++*, *Java Methods*, *Be Prepared for the AP Computer Science Exam in Java*, *250 MC Questions,* and CML computer science contests. Gary has worked in many areas of software development, including artificial intelligence, pattern recognition, computer graphics, and neural networks. As the founder of Skylight Software, Inc., he developed SKYLIGHTS/GX, one of the first visual programming tools for C and C++ programmers. Gary led the development of several state-of-the-art software products, including interactive touch screen development tools, OCR and handwritten character recognition systems, and credit card fraud detection software.

Contents

Preface

This book is a “Python early” remake of our earlier book *Mathematics for the Digital Age and Programming in Python*. We introduce more Python features earlier, giving the reader the necessary tools to start writing Python code sooner and in a more “pythonic” (idiomatic) manner. We have added two chapters — “Turtle Graphics” and “Vectors and Matrices” — and a separate section on Fibonacci numbers (in the “Sums and Sequences” chapter). We have updated many examples, exercises, and solutions. And we have changed the title to better match the sequence of topics and the fast-changing technological landscape and vocabulary.

But the main idea of the book remains the same: to introduce the discrete mathematics concepts that we consider essential knowledge for any literate coder. This kind of math is very accessible, yet most K-12 math curricula in the United States do not yet include it. The math segments of this book include many hands-on coding exercises, which reinforce students’ learning of both coding and math.

❖ ❖ ❖

“So, is this a math book or a computer programming book?” This is probably the first question on the impatient reader’s mind. But why should it be? It is a librarian’s dilemma: “Does it go on the math shelf or on the computer shelf?” There is a simple solution: put a copy on each.

The purpose of this book is to teach a particular way of thinking — precision thinking — and how to solve problems that require this way of thinking. Both mathematics and computer programming nourish the ability to think with precision and to solve problems that call for exact solutions.

Mathematics teaches us to appreciate the beauty of a rigorous argument. In the long run, this is more valuable than a lesson on solving today’s practical problems. Still, mathematics does not exist in a vacuum — its abstractions are rooted in practical knowledge accumulated over centuries. The teaching of mathematics draws on examples and analogies from the world around us. At least, it should. However, the world around us is changing more and more rapidly. In the past 50 or 60 years, our world has gone digital. This change is so profound that it is sometimes hard to fully comprehend. Is that why the change remains largely ignored in our K-12 math curricula? We need to start filling the gap.

If we could build a time machine and bring Euclid over for a visit, he would find it comforting, amid the chaos of modern technologies, that geometry familiar to him is still taught in schools. Old rivals Newton and Leibniz would both find great satisfaction in the fact that tens of thousands of 11th and 12th graders are learning how to take derivatives and use integrals. But George Boole, a visitor from the more recent past, would have to search through dozens of school textbooks before he could find his algebra of propositions mentioned even in passing, even though his name is immortalized in every modern computer programming language. As for John von Neumann, a brilliant mathematician and one of the fathers of computer technology... well, with his usual optimism he would predict that within 20 years or so, every elementary school student will be learning about the AND, OR, and NOT gates. And why not?

In this book we have collected some of the easier mathematical topics that are relevant to the digital world. Many of these topics are often bundled together in freshman college courses under the name *discrete mathematics*. Discrete mathematics has become a euphemism for all elementary mathematics that is relevant today but neglected in standard middle and high school algebra, precalculus, and calculus courses. In the 1970s, Donald Knuth and his colleagues at Stanford coined the phrase “concrete mathematics” — a blend of CONtinuous and disCRETE mathematics (and also solid and not too abstract) — to describe the course Knuth taught at Stanford. Later, *Concrete Mathematics* became the title of their delightful book.[1] As they explain in their preface, Knuth “had found that there were mathematical tools missing from his repertoire; the mathematics he needed for a thorough, well-grounded understanding of computer programs was quite different from what he’d learned as a mathematics major in college.”

We believe that college is too late to start. Many concepts are completely accessible to middle and high school students. And there is also another side to the relationship: just as mathematics helps achieve a deeper understanding of computer programs, some hands-on experience with computer programming helps make mathematics more tangible, more familiar, and easier to grasp.

So, if you are interested mainly in computers, we hope this book will make you a better computer programmer. If you are more interested in math, you will have ample opportunities to solve interesting problems and model some of them in computer programs. You will become familiar with fun areas of mathematics that are usually kept from middle and high school students; you will learn to solve real problems (that is, problems that you don’t already know how to “solve” ahead of

[1] Ronald L. Graham, Donald E. Knuth, Oren Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, Second Edition, Addison-Wesley, 1998.

time); you will learn the power of mathematical reasoning and proof. As a bonus, you will acquire the practical skill of programming in Python, a popular commercial programming language.

We chose Python for several reasons. First, Python gives you a chance to experiment with the language in an interactive setting with immediate feedback. Second, Python's syntax is not too complicated. Third, Python has simple yet powerful features for working with lists and "dictionaries" (maps). Finally, Python is easy to install and get started with, and it's free. Of course, there are other programming languages that have similar properties and would meet our needs. In the end, it is not any particular programming language that matters, but rather the ability to think with precision about both mathematical facts and computer programs.

❖ ❖ ❖

This book has benefitted from the energy and precision of many friends and allies in the K-12 computer science and mathematics community.

We particularly want to thank Abby Ross of Northfield Mount Hermon high school, who read the entire book carefully and made many valuable corrections and suggestions. Dr. Patricia M. Davies of Prince Mohammad Bin Fahd University, a supporter since this book's earlier incarnation, read this version carefully and offered many useful revisions. Hans Batra of Needham High School provided helpful comments.

The previous incarnation of this book, *Mathematics for the Digital Age and Programming in Python*, benefitted from the ideas and suggestions of friends including Dr. J. Adrian Zimmer (Oklahoma School of Science and Mathematics) and Kenneth S. Oliver (formerly of Amity Regional High School in Woodbridge, Connecticut). Prof. Duncan A. Buell, then Chair of the Department of Computer Science and Engineering at University of South Carolina in Columbia, read a draft and suggested many improvements, especially for the Number Theory and Cryptology chapter.

We are grateful to Henry Garden for advice on texting habits and to Margaret Litvin for proofreading help. As ever, our deepest gratitude goes to Maria's math and computer science students: they have cheerfully test-driven the exercises presented here, provoked many good clarifications, and generally buoyed us with their enthusiasm for this material.

How to Use This Book

The *Coding in Python and Elements of Discrete Mathematics* companion web site —

`http://www.skylit.com/python`

— is an integral part of this book. It contains downloadable student files for exercises, *Getting Started with Python* (Appendix A), links, errata, supplemental papers and syllabi, and technical support information for teachers.

`Py` refers to the *Coding in Python* student (or teacher) files. For example, "See `Py\PythonCode\Fibonacci.py`" means the `Fibonacci.py` file is located in the `PythonCode` folder in `StudentFiles` (and `TeacherFiles`).

⤵ ⤴ Arrow brackets like these, in the margin, mark supplementary material intended for a more inquisitive reader. This material either gives a glimpse of things to come in subsequent chapters or adds technical details.

1.■, **2.**◆ In exercises, a "blue" square indicates an "intermediate" question that may require more thought or work than an "easy" question or exercise. A black diamond indicates an "advanced" question that could be treacherous, take a lot of work, or lead to unexplored territory.

✓ A checkmark at the end of a question in the exercises means that the answer or a solution is included in the student files. We have included answers and solutions to about half of the exercises. They can be found in `www.skylit.com/python/studentfiles.zip/StudentFiles/AnswersAndSolutions.pdf`.

Digital teacher files, which contain complete solutions to all the exercises and labs, are available free of charge to teachers who use this book as a textbook in their school. Go to `skylit.com/python` and click on the "Teachers' Room" link for details. A printed version is available, too.