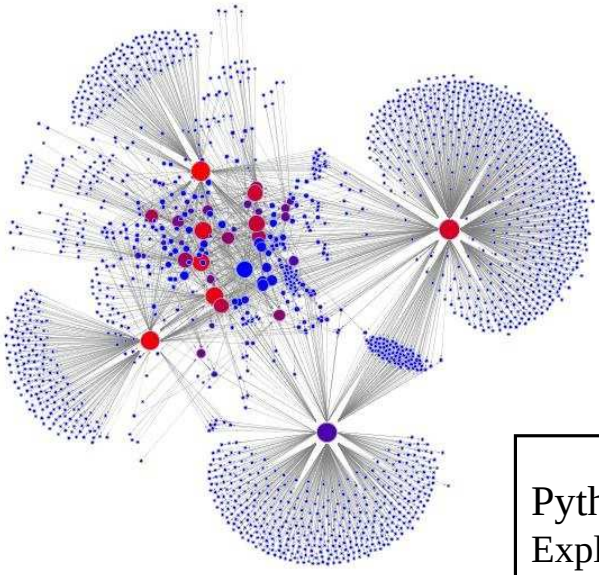
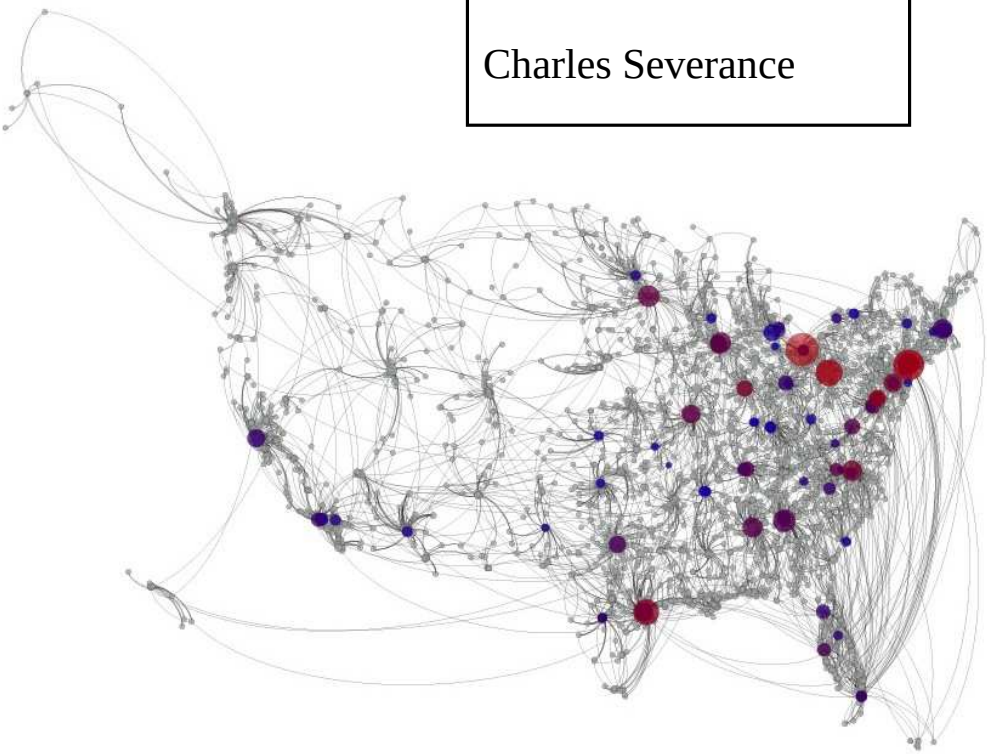


Python for Informatics
Exploring Data

Charles Severance

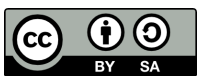


Python for Informatics

Exploring Information

Version 0.0.3

Charles Severance



Printing history:

December 2009: Begin to produce *Python for Informatics: Exploring Information* by re-mixing *Think Python: How to Think Like a Computer Scientist*

June 2008: Major revision, changed title to *Think Python: How to Think Like a Computer Scientist*.

August 2007: Major revision, changed title to *How to Think Like a (Python) Programmer*.

April 2002: First edition of *How to Think Like a Computer Scientist*.

The original form of this book is LaTeX source code. Compiling this LaTeX source has the effect of generating a device-independent representation of a textbook, which can be converted to other formats and printed.

The LaTeX source for the *Think Python: How to Think Like a Computer Scientist* version of this book is available from `http://www.thinkpython.com`.

The LaTeX source for the *Python for Informatics: Exploring Information* version of the book is available (for the moment) from `http://source.sakaiproject.org/contrib//csev/trunk/pyinf/`.

The cover images were provided by Dr. Lada Adamic and are used with permission.

Preface

Python for Informatics: Remixing an Open Book

It is quite natural for academics who are continuously told to "publish or perish" to want to always create something from scratch that is their own fresh creation. This book is an experiment in not starting from scratch, but instead "re-mixing" the book titled *Think Python: How to Think Like a Computer Scientist* written by Allen B. Downey, Jeff Elkner and others.

In December of 2009, I was preparing to teach **SI502 - Networked Programming** at the University of Michigan for the fifth semester in a row and decided it was time to write a Python textbook that focused on exploring data instead of understanding algorithms and abstractions. My goal in SI502 is to teach people life-long data handling skills using Python. Few of my students were planning to be be professional computer programmers. Instead, they planned be librarians, managers, lawyers, biologists, economists, etc. who happened to want to skillfully use technology in their chosen field.

I never seemed to find the perfect data-oriented Python book for my course so I set out to write just such a book. Luckily at a faculty meeting three weeks before I was about to start my new book from scratch over the holiday break, Dr. Atul Prakash showed me the *Think Python* book which he had used to teach his Python course that semester. It is a well-written Computer Science text with a focus on short, direct explanations and ease of learning.

As the copyright holder of *Think Python*, Allen has given me permission to change the book's license from the GNU Free Documentation License to the more recent Creative Commons Attribution — Share Alike license. This follows a general shift in open documentation licenses moving from the GFDL to the CC-BY-SA (i.e. Wikipedia). Using the CC-BY-SA license maintains the book's strong copyleft tradition while making it even more straightforward for new authors to reuse this material as they see fit.

I expect that by the time I am done with *Python for Informatics* over fifty percent of the book will be new. The overall structure will be changed to get to doing data analysis problems as quickly as possible and have a series of running examples and exercises about data analysis. Then I will add chapters on regular expressions, data visualization, working with spreadsheet data, structured query language using SQLite, web scraping, and calling REST-based Application Program Interfaces.

The ultimate goal in the shift from a Computer Science to an Informatics focus is to pull topics into the first programming class that can be applied even if one chooses not to become a professional programmer.

What is interesting even with this change of focus is how much of the original *Think Python* book material is directly relevant to this book and how much will fit right into *Python for Informatics* with virtually no change.

By starting with the *Think Python* book, I don't have to write the basic descriptions of the Python language or how to debug programs and instead focus on the topical material that is the value-add of *Python for Informatics*.

Students who find this book interesting and want to further explore a career as a professional programmer should probably look at the *Think Python* book. Because there is a lot of overlap between the two books, you will quickly pick up skills in the additional areas of Computer Science which are covered in *Think Python*. And given that the books have a similar writing style and at times have identical text and examples, you should be able to pick up these new topics with a minimum of effort.

I hope that this book serves an example of why open materials are so important to the future of education, and want to thank Allen B. Downey and Cambridge University Press for their forward looking decision to make the book available under an open Copyright. I hope they are pleased with the results of my efforts and I hope that you the reader are pleased with *our* collective efforts.

Charles Severance
www.dr-chuck.com
December 19, 2009

Charles Severance is a Clinical Assistant Professor at the University of Michigan School of Information.

Draft Version Instructions

The copy of this book you are looking at is currently a draft and still in development. The general roadmap for the rest of the development book is as follows:

- Teach SI502 - Networked Programming at University of Michigan Winter 2010. The first 10 chapters of the book will be used for the first four weeks of the course. At least three more chapters will be written for SI502 and distributed during the semester that line up with the topics in the second half of SI502 (Networked Programming, Databases, and Using Web Services).
- There are four more chapters planned at some point (Advanced Functions, Regular Expressions, Automating Common Tasks, and Visualizing data). These are not currently in the scope of SI502 for Winter 2010.

Like all books being written and used in a course at the same time, student feedback is essential to producing a strong book. So I hope that students will look at the book and

help me find simple errors, places where ideas jump too fast, improvements in the glossary, debugging, and exercises in each chapter.

You can also send comments to csev (at) umich.edu at any time.

Thanks in advance for your patience and assistance.

Preface for "Think Python"

The strange history of "Think Python"

(Allen B. Downey)

In January 1999 I was preparing to teach an introductory programming class in Java. I had taught it three times and I was getting frustrated. The failure rate in the class was too high and, even for students who succeeded, the overall level of achievement was too low.

One of the problems I saw was the books. They were too big, with too much unnecessary detail about Java, and not enough high-level guidance about how to program. And they all suffered from the trap door effect: they would start out easy, proceed gradually, and then somewhere around Chapter 5 the bottom would fall out. The students would get too much new material, too fast, and I would spend the rest of the semester picking up the pieces.

Two weeks before the first day of classes, I decided to write my own book. My goals were:

- Keep it short. It is better for students to read 10 pages than not read 50 pages.
- Be careful with vocabulary. I tried to minimize the jargon and define each term at first use.
- Build gradually. To avoid trap doors, I took the most difficult topics and split them into a series of small steps.
- Focus on programming, not the programming language. I included the minimum useful subset of Java and left out the rest.

I needed a title, so on a whim I chose *How to Think Like a Computer Scientist*.

My first version was rough, but it worked. Students did the reading, and they understood enough that I could spend class time on the hard topics, the interesting topics and (most important) letting the students practice.

I released the book under the GNU Free Documentation License, which allows users to copy, modify, and distribute the book.

What happened next is the cool part. Jeff Elkner, a high school teacher in Virginia, adopted my book and translated it into Python. He sent me a copy of his translation, and I had the unusual experience of learning Python by reading my own book.

Jeff and I revised the book, incorporated a case study by Chris Meyers, and in 2001 we released *How to Think Like a Computer Scientist: Learning with Python*, also under the

GNU Free Documentation License. As Green Tea Press, I published the book and started selling hard copies through Amazon.com and college book stores. Other books from Green Tea Press are available at `greenteapress.com`.

In 2003 I started teaching at Olin College and I got to teach Python for the first time. The contrast with Java was striking. Students struggled less, learned more, worked on more interesting projects, and generally had a lot more fun.

Over the last five years I have continued to develop the book, correcting errors, improving some of the examples and adding material, especially exercises. In 2008 I started work on a major revision—at the same time, I was contacted by an editor at Cambridge University Press who was interested in publishing the next edition. Good timing!

I hope you enjoy working with this book, and that it helps you learn to program and think, at least a little bit, like a computer scientist.

Acknowledgements for "Think Python"

(Allen B. Downey)

First and most importantly, I thank Jeff Elkner, who translated my Java book into Python, which got this project started and introduced me to what has turned out to be my favorite language.

I also thank Chris Meyers, who contributed several sections to *How to Think Like a Computer Scientist.*

And I thank the Free Software Foundation for developing the GNU Free Documentation License, which helped make my collaboration with Jeff and Chris possible.

I also thank the editors at Lulu who worked on *How to Think Like a Computer Scientist.*

I thank all the students who worked with earlier versions of this book and all the contributors (listed in an Appendix) who sent in corrections and suggestions.

And I thank my wife, Lisa, for her work on this book, and Green Tea Press, and everything else, too.

Allen B. Downey
Needham MA

Allen Downey is an Associate Professor of Computer Science at the Franklin W. Olin College of Engineering.

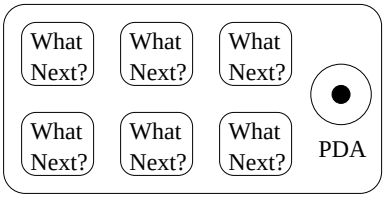
Contents

Chapter 1

Why should you learn to write programs?

Writing programs (or programming) is a very creative and rewarding activity. You can write programs for many reasons ranging from making your living to solving a difficult data analysis problem to having fun to helping someone else solve a problem. This book assumes that *everyone* needs to know how to program and that once you know how to program, you will figure out what you want to do with your newfound skills.

We are surrounded in our daily lives with computers ranging from laptops to cell phones. We can think of these computers as our "personal assistants" who can take care of many things on our behalf. The hardware in our current-day computers is essentially built to continuously ask us the question, "What would you like me to do next?".



Programmers add and operating system and a set of applications to the hardware and we end up with a Personal Digital Assistant that is quite helpful and calable of helping many different things.

Our computers are fast and have vast amounts of memory and could be very helpful to us if we only knew the language to speak to explain to the computer what we would like it to "do next". If we knew this language we could tell the computer to do tasks on our behalf that were repetitive. Interestingly, the kinds of things computers can do best are often the kinds of things that we humans find boring and mind-numbing.

For example, look at the first three paragraphs of this chapter and tell me the most commonly used word and how many times the word is used. While you were able to read and