

## Preface

By Jeffrey Elkner

This book owes its existence to the collaboration made possible by the Internet and the free software movement. Its three authors—a college professor, a high school teacher, and a professional programmer—never met face to face to work on it, but we have been able to collaborate closely, aided by many other folks who have taken the time and energy to send us their feedback.

We think this book is a testament to the benefits and future possibilities of this kind of collaboration, the framework for which has been put in place by Richard Stallman and the Free Software Foundation.

### How and why I came to use Python

In 1999, the College Board's Advanced Placement (AP) Computer Science exam was given in C++ for the first time. As in many high schools throughout the country, the decision to change languages had a direct impact on the computer science curriculum at Yorktown High School in Arlington, Virginia, where I teach. Up to this point, Pascal was the language of instruction in both our first-year and AP courses. In keeping with past practice of giving students two years of exposure to the same language, we made the decision to switch to C++ in the first year course for the 1997–98 school year so that we would be in step with the College Board's change for the AP course the following year.

Two years later, I was convinced that C++ was a poor choice to use for introducing students to computer science. While it is certainly a very powerful programming language, it is also an extremely difficult language to learn and teach. I found myself constantly fighting with C++'s difficult syntax and multiple ways of doing things, and I was losing many students unnecessarily as a result. Convinced there had to be a better language choice for our first-year class, I went looking for an alternative to C++.

I needed a language that would run on the machines in our GNU/Linux lab as well as on the Windows and Macintosh platforms most students have at home. I wanted it to be free software, so that students could use it at home regardless of their income. I wanted a language that was used by professional programmers, and one that had an active developer community around it. It had to support both procedural and object-oriented programming. And most importantly, it had to be easy to learn and teach. When I investigated the choices with these goals in mind, Python stood out as the best candidate for the job.

I asked one of Yorktown's talented students, Matt Ahrens, to give Python a try. In two months he not only learned the language but wrote an application called pyTicket that enabled our staff to report technology problems via the Web. I knew that Matt could not have finished an application of that scale in so short a time in C++, and this accomplishment, combined with Matt's positive assessment of Python, suggested that Python was the solution I was looking for.

### Finding a textbook

Having decided to use Python in both of my introductory computer science classes the following year, the most pressing problem was the lack of an available textbook.

Free documents came to the rescue. Earlier in the year, Richard Stallman had introduced me to Allen Downey. Both of us had written to Richard expressing an interest in developing free educational materials. Allen had already written a first-year computer science textbook, *How to Think Like a Computer Scientist*. When I read this book, I knew immediately that I wanted to use it in my class. It was the clearest and most helpful computer science text I had seen. It emphasized the processes of thought involved in programming rather than the features of a particular language. Reading it immediately made me a better teacher.

*How to Think Like a Computer Scientist* was not just an excellent book, but it had been released under the GNU public license, which meant it could be used freely and modified to meet the needs of its user. Once I decided to use Python, it occurred to me that I could translate Allen's original Java version of the book into the new language. While I would not have been able to write a textbook on my own, having

Allen's book to work from made it possible for me to do so, at the same time demonstrating that the cooperative development model used so well in software could also work for educational materials.

Working on this book for the last two years has been rewarding for both my students and me, and my students played a big part in the process. Since I could make instant changes whenever someone found a spelling error or difficult passage, I encouraged them to look for mistakes in the book by giving them a bonus point each time they made a suggestion that resulted in a change in the text. This had the double benefit of encouraging them to read the text more carefully and of getting the text thoroughly reviewed by its most important critics, students using it to learn computer science.

For the second half of the book on object-oriented programming, I knew that someone with more real programming experience than I had would be needed to do it right. The book sat in an unfinished state for the better part of a year until the open source community once again provided the needed means for its completion.

I received an email from Chris Meyers expressing interest in the book. Chris is a professional programmer who started teaching a programming course last year using Python at Lane Community College in Eugene, Oregon. The prospect of teaching the course had led Chris to the book, and he started helping out with it immediately. By the end of the school year he had created a companion project on our Website at <http://openbookproject.net> called *\*Python for Fun\** and was working with some of my most advanced students as a master teacher, guiding them beyond where I could take them.

## Introducing programming with Python

The process of translating and using *How to Think Like a Computer Scientist* for the past two years has confirmed Python's suitability for teaching beginning students. Python greatly simplifies programming examples and makes important programming ideas easier to teach.

The first example from the text illustrates this point. It is the traditional hello, world program, which in the Java version of the book looks like this:

```
class Hello {  
    public static void main (String[] args) {  
        System.out.println ("Hello, world.");  
    }  
}
```

in the Python version it becomes:

```
print("Hello, World!")
```

Even though this is a trivial example, the advantages of Python stand out. Yorktown's Computer Science I course has no prerequisites, so many of the students seeing this example are looking at their first program. Some of them are undoubtedly a little nervous, having heard that computer programming is difficult to learn. The Java version has always forced me to choose between two unsatisfying options: either to explain the `class Hello, public static void main, String[] args, {, and }`, statements and risk confusing or intimidating some of the students right at the start, or to tell them, Just don't worry about all of that stuff now; we will talk about it later, and risk the same thing. The educational objectives at this point in the course are to introduce students to the idea of a programming statement and to get them to write their first program, thereby introducing them to the programming environment. The Python program has exactly what is needed to do these things, and nothing more.

Comparing the explanatory text of the program in each version of the book further illustrates what this means to the beginning student. There are seven paragraphs of explanation of Hello, world! in the Java version; in the Python version, there are only a few sentences. More importantly, the missing six paragraphs do not deal with the big ideas in computer programming but with the minutia of Java syntax. I found this same thing happening throughout the book. Whole paragraphs simply disappear from the Python version of the text because Python's much clearer syntax renders them unnecessary.

Using a very high-level language like Python allows a teacher to postpone talking about low-level details of the machine until students have the background that they need to better make sense of the details. It

thus creates the ability to put first things first pedagogically. One of the best examples of this is the way in which Python handles variables. In Java a variable is a name for a place that holds a value if it is a built-in type, and a reference to an object if it is not. Explaining this distinction requires a discussion of how the computer stores data. Thus, the idea of a variable is bound up with the hardware of the machine. The powerful and fundamental concept of a variable is already difficult enough for beginning students (in both computer science and algebra). Bytes and addresses do not help the matter. In Python a variable is a name that refers to a thing. This is a far more intuitive concept for beginning students and is much closer to the meaning of variable that they learned in their math courses. I had much less difficulty teaching variables this year than I did in the past, and I spent less time helping students with problems using them.

Another example of how Python aids in the teaching and learning of programming is in its syntax for functions. My students have always had a great deal of difficulty understanding functions. The main problem centers around the difference between a function definition and a function call, and the related distinction between a parameter and an argument. Python comes to the rescue with syntax that is nothing short of beautiful. Function definitions begin with the keyword `def`, so I simply tell my students, When you define a function, begin with `def`, followed by the name of the function that you are defining; when you call a function, simply call (type) out its name. Parameters go with definitions; arguments go with calls. There are no return types, parameter types, or reference and value parameters to get in the way, so I am now able to teach functions in less than half the time that it previously took me, with better comprehension.

Using Python improved the effectiveness of our computer science program for all students. I saw a higher general level of success and a lower level of frustration than I experienced teaching with either C++ or Java. I moved faster with better results. More students left the course with the ability to create meaningful programs and with the positive attitude toward the experience of programming that this engenders.

## Building a community

I have received email from all over the globe from people using this book to learn or to teach programming. A user community has begun to emerge, and many people have been contributing to the project by sending in materials for the companion Website at <http://openbookproject.net/pybiblio>.

With the continued growth of Python, I expect the growth in the user community to continue and accelerate. The emergence of this user community and the possibility it suggests for similar collaboration among educators have been the most exciting parts of working on this project for me. By working together, we can increase the quality of materials available for our use and save valuable time. I invite you to join our community and look forward to hearing from you. Please write to me at [jeff@elkner.net](mailto:jeff@elkner.net).

Jeffrey Elkner  
Governor's Career and Technical Academy in Arlington  
Arlington, Virginia