Programming has rapidly become an essential tool for nearly all students in the physical sciences. One of the most widely used languages in this realm is Python, which has swiftly gained popularity due to its readability and intuitive syntax. The core philosophy or “Zen of Python” dictates, “simple is better than complex, complex is better than complicated” (https://www.python.org/dev/peps/pep-0020/). Because of this motto, learning how to use Python effectively on your own is a very doable task, given the right resources.

This is where A Student’s Guide to Python for Physical Modeling by Jesse M. Kinder and Philip Nelson comes in. The text serves as an excellent stepping stone into the world of using Python in computational science for undergraduate students with a strong background in mathematics. After working through the chapters and their accompanying exercises, readers can expect not only to know how to write and read Python but also to achieve a thorough understanding for developing complex physical models and calculations.

**Approachability and Organization**
From the get-go, this aptly named “student’s guide” presupposes no prior knowledge of programming. Appendix A contains instructions on how to install, launch, and set up everything Python that you’ll need for this book, including the suggested development environment (Spyder) and relevant packages (such as Anaconda). For slightly more experienced users, the text offers advice and information on extensions for auto updating, acceleration, and FFmpeg. The book accounts for backward incompatibility (Python 2 and 3) in Appendix B, shedding light—in simple terms—on how the reader can overcome errors raised by this issue.

Much of the first chapter, “Getting Started with Python,” is written for those who have little or no background in programming in general. It establishes basic ideas such as algorithmic thinking, clarifies the use and nature of algorithms by comparing and contrasting them with mathematical proofs, and discusses and demonstrates the use of common mathematical symbols in Python. In the last few pages, it succinctly introduces the use of modules, error resolution, variable creation, and functions through try-it-yourself snippets of code.

Learning through experimentation is highly emphasized in this book, which states outright that “reading this tutorial won’t teach you Python. You can teach yourself Python by working through all the examples and exercises here, and then using what you’ve learned on your own problems.” New concepts are almost always introduced first through coding exercises, then explained and elaborated upon afterward. The authors also provide and encourage readers to use online resources such as stackoverflow.com, where Python programmers go to resolve errors, find more efficient methods, and learn new methods.

The language used throughout the book is simple and conversational, with concepts presented and explained in a much less intimidating manner than, say, that used in a conventional textbook. The text is very thoughtfully organized and covers the essentials for physical modeling. Sometimes, examples of the application of new concepts appear before their explanation, helping students develop intuition to better understand Python constructions. The chapters are organized so that structure and control are covered first, then data handling (calculations and graphics) and functions, before using these basic concepts as a gateway to more complex Python constructions (such as contour plots, numerical solutions to nonlinear equations, vector fields, image processing, and animation).

However, in contrast to how approachable the authors make programming in Python, the rather complex physics and math models in later chapters could be confusing for undergraduate students lacking this same background, as the knowledge of these topics are assumed to be a priori.

**Insights and Analysis**

One of the inconveniences (probably intentional) in this book is how some chapters are challenging to follow and learn from without actively working through the examples in a Python integrated development environment (IDE) as you read. In particular, examples within chapters lack line numbers, but the text still refers to line numbers in the example, which makes it hard to follow unless you have the IDE open with the example loaded. Moreover, the text doesn’t always reveal what to expect after running an example code and instead encourages readers to make the discovery themselves.

Although the book is intended for students with some or no experience with computer science, the potential for a more technical understanding of programming, and some of the more technical aspects of Python, are made available through the appendices as well as the supplied online materials. As students with some experience with Python, we found the appendices to be some of the more interesting parts of the book. Here, the authors briefly but thoroughly go over topics such as debugging as well as the more mysterious, less intuitive, but still prevalently encountered “inner workings” of Python. The appendix section on debugging categorizes and interprets some of the most common errors that Python programmers come across. Some errors of particular interest include topics such as ZeroDivisionError versus RuntimeWarning, and the manipulation of AssertionError.

Appendix D, titled “Under the Hood,” takes a straightforward and practical approach to exploring the mystifying topic of how Python handles variables and objects internally in light of exception handling and interacting with more advanced code. This section, in layman terms, sheds light on how assignment statements for arrays when used incorrectly can sometimes result in perplexing results. Kinder and Nelson also touch on memory management, the interaction between variables and objects within functions, and finally how Python keeps track of variables using namespaces.

We found the examples and problems extremely constructive and relevant to the text. The greater than 10 lines of try-it-yourself code snippets sprinkled generously throughout the chapters is as essential as the text itself. These small test codes encourage readers to not only learn new concepts but also gain intuition on how to effectively communicate with Python. They also help build confidence by letting students write, run, and successfully debug small fragments of code. A total of 17 problems (referred to as ‘your turn’) appear over the five chapters, with an additional three “computer lab” chapters sandwiched between sections. The difficulty of the “your turn” questions ranges from array creation and writing simple functions to multipart word problems.
Appendix E holds the answers to these problems. The computer labs are much more complex, walking readers through the creation of a physical model. They reinforce and put learned concepts from previous chapters to use for creating a more complex model.

In fact, some of our favorite models were from the second and third computer labs. The first part of the second computer lab dealt with various computations, trajectory plots, and displacement distributions for 2D random walks. The lab is structured so that assignments lead the reader through these tasks while the text sandwiched between them explains the theory, meaning, and significance of the results. The third computer lab focuses on image manipulation and explores local averaging, decreasing noise, and feature amplification through filters. We thought this was a great final lab because image distortion and editing is very familiar to the audience that the book is intended for. It’s exciting to be able to create programs and explore the mathematical and computational side of science that’s so relevant and prevalent in this day and age.

Overall, we feel the text offers a great primer for mathematically inclined undergraduate students (potentially advanced high school students as well) to get started in Python, programming, and physical modeling. For a full semester-long course, the book might fall somewhat short on material and rigor if used as the sole material. However, if it’s used along with instructor exercises or other surplus material, the text could become a starting point from which to branch out to more sophisticated topics or augment the proffered topics with increased complexity.

Kevin Thielen is an undergraduate student at Eckerd College majoring in mathematics and physics. He has been exposed to using Python for physical modeling through school courses as well as through several research projects. Contact him at kdthiele@eckerd.edu.

Vivienne Tien is an undergraduate student at Eckerd College at St. Petersburg in Florida majoring in mathematics and minoring in physics. She has learned and used Python through research projects and courses. Contact her at wtien@eckerd.edu.

Selected articles and columns from IEEE Computer Society publications are also available for free at http://ComputingNow.computer.org.