Book review

Review of "A functional start to computing with Python", Ted Herman, CRC Press, 2014, ISBN 978-1-4665-0455-4
doi:10.1017/S0956796815000222

Python is an extremely popular programming language in many fields, including my field of data analysis and bioinformatics (Chapman & Irwin, 2015). It's a great language for getting quick jobs done quickly. It's also commonly used as a teaching language in beginner programming courses (Guo, 2014). So the premise of this book was intriguing: take a popular language and use it to teach functional programming concepts. I currently teach Haskell and Python to different groups of beginner students at university, and I've found pros and cons to both as teaching languages for beginners. I was curious to read this book. Would the mixture proposed here be the answer?

The author, Ted Herman, states that he observes that beginners find two things difficult: assignment and control structures (iteration and exceptions). Therefore the first half of the book contains neither and instead makes use of functions, expressions, conditions and comprehensions to construct code. The second half of the book introduces the rest of Python, including mutable variables, loops, modules, IO, classes, etc. He wants to take a functional first approach to learning a popular language. I admire the unusual aim. Python books are not normally pitched in this way. Functional concepts are more usually introduced as an advanced curiosity after the main topics. Python is currently far more popular than functional languages, and this approach could open up functional programming ideas to a new audience. The book is aimed at beginner programmers, and assumes no prior knowledge of programming.

The book has four sections: Motivation and Background (introductory chapters that include how to set up Python), Functional-Style Python, Imperative-Style Python and Appendices. There are exercises after each chapter, with selected answers available at the back of the book. There is also an accompanying website with flashcards to test the student, an interactive tracing environment to step through code, downloads of unit tests and a drawing library, and extra snippets of info and code to support some parts of the book.

This is a fairly large book (415 pages) packed with information, examples and ideas. In its scope, it's almost a reference book. However, as a book aimed at beginners I feel that it takes on too much material, and would have been better if it had taken on just half the material, but explained it in more detail. Programming is a large subject. Python is a hybrid language crossing multipleprogramming paradigms. Python has issues caused by version differences (version 2 versus version 3), it has different styles (object-oriented versus procedural versus FP), and it also has specialist "Pythonic" ways to do a variety of tasks. The book does reflect the untidiness of the language, and has to take care to explain quirks that don't quite fit the functional programming narrative. The book is honest, and indicates where life is just not ideal and the functional style is not considered to be the Pythonic way to work.

There are plenty of functional programming concepts addressed in the first half of the book, but often too briefly. Function composition, local functions and currying are each introduced in less than a page, with one example for each. Recursion can be a difficult concept for beginner programmers but it is not given the space that it needs. It is introduced

first as "tail recursion". The statement that "Tail recursion is the pattern of processing the items of a sequence one by one, starting from the first items, and continuing until the last item" on page 160 is illustrated at first with a function that is entirely about side effects (printing uppercase versions of a list of strings). I imagine that beginners are going to be particularly confused by these recursion explanations.

It is always difficult to structure a subject as complex as programming into a linear flow, ordered through chapters, one following another, with dependencies satisfied, and with important ideas dealt with in enough detail at appropriate times. However, I don't understand why comments are not introduced until page 246 (nor why after pointing out the PEP 8 style guide, Herman chooses not to conform to the Python conventions for naming variables, methods and classes). Exceptions are somewhat hidden away in a late chapter on "Network Programs". There are also a huge number of other important concepts crammed towards the back of the book but given not enough space to really explain why they are all useful. For example, it is difficult to understand the point of decorators from the one example given, which uses global variables to do a meaningless task.

Many books that aim to teach functional programming to beginners tend to assume a relatively high level of mathematics competence from the students (familiarity with notation and terminology), and this book does not do this. It is practical in its style. Python programmers generally are a practical bunch, and other beginner Python books (for example Learning Python (Lutz, 2013)) also tend to follow this get-things-done style rather than the more mathematical style beloved of functional programming books. On the back of the book Herman promises core computer science ideas, including "self-referencing structures, aliases and finite state machines (FSMs)". There are indeed many core computer science concepts scattered through the book, sometimes hidden within other topics, sometimes highlighted in grey boxes, sometimes the focus of a chapter. The ideas that he lists here on the cover seem an odd choice to highlight. The discussion of aliases is limited to the issue of multiple variable names referencing the same data item, and neither self-referencing structures nor FSMs are to be found in the index (the index does need to be much more comprehensive). FSMs are introduced in an interlude on game development. This section on FSMs is quite representative of the book as a whole: it motivates them and then briefly introduces the idea by the use of examples, without ever formalising the idea, and then suggests that an interested reader can find out more about the idea online using a search engine.

I would perhaps buy the book for myself, as someone who teaches programming, rather than recommend it to my students. It's expensive for a paperback at £49.99 but it is a large book with some interesting ideas and exercises. It is easy to read, clearly laid out and doesn't shy away from the awkward parts. The different approach that Herman uses may help me to provide alternative approaches when explaining concepts to students. We all need to use a variety of approaches to reach students who haven't yet grasped an idea. However, after reading the book, I am now well aware of the hurdles that would be need to be overcome in order to use Python as a popular vehicle to teach functional programming concepts.

As for using a functional programming perspective to help teach Python, this book has clarified to me why this will also be difficult, and how this often goes against the grain of Python. So I cannot see a situation where I would recommend it as a course textbook to my students. This is not really the fault of the book. Python offers functional features without ever actually being a functional language. Assignments, loops, classes and objects are not just supported but encouraged in Python, and this is the more Pythonic way to write Python. For the purposes of clearly teaching or learning about the benefits of functional programming, we have a more straightforward story using a functional language where the ideas are a more natural fit. I am now curious to know whether this experiment has worked for Herman. Has he taught students using this style, and was it more effective than teaching Python using a Pythonic style or teaching functional concepts using a functional language? He does not let on.

References

Chapman, B. E. & Irwin, J. (2015) Python as a first programming language for biomedical scientists. In Proceedings of the 14th Python in Science Conference (SCIPY 2015): Published online at http://conference.scipy.org/proceedings/scipy2015/ (last accessed 23-09-2015).

Guo, P. (2014) Python is now the most popular introductory teaching language at Top U.S. Universities, Communications of the ACM. Available at: http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/ (last accessed 23-09-2015).

Lutz, M. (2013) *Learning Python*. O'Reilly Media, ISBN: 978-1-449-35573-9.

AMANDA CLARE
Department of Computer Science, Aberystwyth University, Aberystwyth, UK

https://doi.org/10.1017/S0956796815000222 Published online by Cambridge University Press