Teaching programming skills is a hard task. It is even harder if one targets an audience with no or little mathematical background. Although there are books on programming that target such groups, they often fail to raise or maintain interest due to artificial examples that lack reference to the professional issues that the audience typically face. This book fills the gap by addressing linguistics, a profession and academic subject for which basic knowledge of script programming is becoming more and more important. The book *Python for Linguists* by Michael Hammond is an introductory Python course targeted at linguists with no prior programming background. It succeeds previous books for Perl (Hammond 2008) and Java (Hammond 2002) by the same author, and reflects the current de facto prevalence of Python when it comes to adoption and available packages for natural language processing.

We feel it necessary to clarify that the book aims at (general) linguists in the broad sense rather than computational linguists. Its aim is to teach linguists the fundamental concepts of programming using typical examples from linguistics. The book should not be mistaken as a course for learning basic algorithms in computational linguistics. We acknowledge that the author nowhere makes such a claim; however, given the thematic proximity to computational linguistics, one should have the right expectation before working with the book.

Chapters 1–5 lay the foundations of the Python programming language, introducing the most important language constructs but deferring object oriented programming to a later part of the book. The focus in Chapters 1 and 2 covers the basic data types (numbers, strings, dictionaries), with a particular emphasis on simple string operations, and introduces some more advanced concepts such as mutability.

Chapters 3–5 introduce control structures, input–output operations, and modules. The book goes at great length to visualize the program flow and the state of different variables for different steps in a program execution, which is certainly very helpful for learners with no prior programming experience. The book also guides the learner to understand certain error types that frequently occur in computer programming (but might be unintuitive for beginners). For example, when discussing function calls, much care is devoted to pointing out the unintended consequences stemming from mutability and side effects.
The book draws connections to linguistics by using a made-up, nonsensical language for some of the examples (e.g., producing artificial sentences that follow a particular pattern). These examples could be made more relevant for linguists if a real language fragment were used.

It is great that the book shows how to combine the power of character streams through piping (with command line tools on the operating system level) with further processing in a Python script, as mastery of this useful skill can become very handy for any language researcher. However, at certain places the chance was missed to teach beginners how canonical Python code should be written. For example, the syntax used for reading files does not correspond to the official recommendation for file input/output, which encourages the use of the `with open(filename)` construct. Also, for variable naming, the book includes many examples with very short one- or two-letter variables, or names in CamelCase, both of which are discouraged by the Python style guidelines.

The first part of the book ends with the step-by-step construction of a script that reads in a book from Project Gutenberg, showing the implementations of necessary helper methods, such as sentences splitting and tokenization, and prints out some basic statistics of the text.

The book does a great job introducing all important concepts in a meaningful order and laying the foundation for programming in Python without leaving gaps that could derail beginners that are not yet used to the frustrations inherent to writing software. The idea of efficiency and runtime complexity is less pronounced in this book—probably because it is directed to linguists rather than starting computer scientists. However, one might argue that conveying a basic idea of runtime and how it is dependent on certain choices (1 step vs. 1 loop vs. 2 nested loops) would have been a useful addition. This could have been combined with explaining the basic ideas and motivations behind frequently used data structures (finding an element in a list vs. in a set, lookup in dictionary).

Chapters 6–8 go into more detail on showing how more complicated text processing problems are solved using regular expressions, text manipulation, and Web crawling. Regular expressions are introduced using many step-by-step examples and explanations. The first exemplary use case is finding consonant clusters and is inspired by phonology. This use case also demonstrates how rule-based engineering is an iterative process, and initial regular expressions are refined once the effect of their application to actual language data can be observed. The second, more elaborate use case is a (somewhat lengthy) reimplementation of the classic Porter stemmer algorithm (Porter 1980). In order for readers to understand the development of the code better, the book presents the same script in several stages (i.e., repeating the code already shown before). Although this is a good idea in the first, foundational chapters of the book, it becomes tiring when done with the more complex examples like the Porter stemmer (it would be better to present small code snippets, and then their composition in an entire script only once).

An entire chapter is devoted to collecting data from the Web; the use case is to crawl a small corpus for the Welsh language. Many real world problems are introduced that NLP practitioners have to face frequently when working with crawled data (such as inconsistent encodings, noisy markup, unresponsive Web pages), as well as tools for

---

1 https://docs.python.org/3/tutorial/inputoutput.html#tut-files.
2 https://www.python.org/dev/peps/pep-0008/#id36.
3 https://www.gutenberg.org/.
Book Review

The book focuses on parallel processing and text analysis, detailing the use of Python for creating programs that can be executed in parallel for increased efficiency. The authors present a mix of practical examples and conceptual descriptions, allowing readers to understand both the theoretical underpinnings and the practical applications of parallel processing.

The book starts with an introduction to the basics of Python programming, including its syntax and basic structures. It then moves on to more advanced topics, such as object-oriented programming (OOP) and functional programming. The authors provide numerous code examples and exercises to help readers understand and apply these concepts.

In terms of content, the book covers a wide range of topics. It starts with the fundamentals of Python programming, and then moves on to more advanced topics such as parallel processing, data manipulation, and text analysis. The book also includes a chapter on graphical user interfaces and a chapter on the natural language toolkit (NLTK), which is a popular library for natural language processing.

The book is well-organized and easy to follow. Each chapter is structured in a logical manner, starting with the basics and then moving on to more advanced topics. The authors also provide numerous examples and exercises to help readers understand and apply the concepts they are learning.

Overall, this book is a valuable resource for anyone interested in parallel processing and Python programming. It is well-written and easy to follow, and it provides a comprehensive introduction to the topic. It is highly recommended for anyone looking to learn more about parallel processing and Python programming.
in the chapter, for example, by modifying the existing examples, up to writing smaller programs from scratch solving particular linguistic tasks.

The linguistic examples chosen may occasionally look a bit construed. Yet, overall they are the main asset of this book and they will be more interesting to the target audience than the examples found in other existing books.

After reading this book, the reader will have a solid grasp of the Python programming language. It should suffice for solving typical daily tasks for linguistics, such as restructuring files or computing low-level statistics (e.g., collecting word frequencies). The knowledge presented in the book may also enable the reader to study more advanced topics related to Python, for example, computational linguistic algorithms, data science, or machine learning.

Some readers may be surprised to see only a few external libraries discussed in the book. However, this may be on purpose and be in line with the didactic concept pursued by the author. The aim of this book is to teach basic programming skills, namely, how to understand and structure code for data processing. This can better be conveyed by showing complete solutions to specific tasks rather than showing how to call a particular Python library. Novices are more likely to acquire a deeper understanding of how programs are written by explaining a programming solution from scratch. Of course, we should also bear in mind that many of the external libraries currently available for Python may be short-lived and not available or maintained in a few years’ time. Nonetheless, we would have appreciated a statement telling the reader to consider publicly available libraries in practice.

**Summary.** The main goal of *Python for Linguists* is to teach basic programming skills to linguists that do not have any prior background in computer science. By using linguistically motivated examples throughout, the book does a great job making the material covered relevant to the target group. The structure of the book is well thought-out, and ensures that prerequisites are covered before moving to more advanced topics. This and the exercises that come with each chapter would make the book a great companion for a foundational programming course targeted at linguists.

**References**


**Benjamin Roth** is a joint professor of Computer Science and Philology at the University of Vienna, Austria. His research interests are the extraction of knowledge from text with statistical methods and knowledge-supervised learning. His e-mail address is benjamin.roth@univie.ac.at.

**Michael Wiegand** is a professor of Computational Linguistics at the Digital Age Research Center, Alpen-Adria-Universität Klagenfurt, Austria. His research interests include abusive language detection, sentiment analysis, and information extraction. His e-mail address is michael.wiegand@aau.at.