Python for Linguists

Specifically designed for linguists, this book provides an introduction to programming using Python for those with little to no experience with coding. Python is one of the most popular and widely used programming languages as it's also available for free and runs on any operating system. All examples in the text involve language data and can be adapted or used directly for language research. The text focuses on key language-related issues: searching, text manipulation, text encoding and internet data, providing an excellent resource for language research. More experienced users of Python will also benefit from the advanced chapters on graphical user interfaces and functional programming.

MICHAEL HAMMOND is Professor of Linguistics and Human Language Technology at the University of Arizona. His previous titles include *Programming for Linguists: Perl for Language Professionals* (2003) and *Programming for Linguists: Java Technology for Language Professionals* (2002).

Python for Linguists

Michael Hammond
University of Arizona

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre,
New Delhi – 110025, India

79 Anson Road, #06–04/06, Singapore 079906

Cambridge University Press is part of the University of Cambridge.

It furthers the University's mission by disseminating knowledge in the pursuit of education, learning, and research at the highest international levels of excellence.

www.cambridge.org
Information on this title: www.cambridge.org/9781108493444
DOI: 10.1017/9781108642408

First published 2020

Printed in the United Kingdom by TJ International Ltd. Padstow Cornwall

A catalogue record for this publication is available from the British Library.

ISBN 978-1-108-49344-4 Hardback
ISBN 978-1-108-73707-4 Paperback

Contents

Preface

This is a book on how to program for linguistic purposes using the Python programming language. In this preface we outline our goals, justify using Python to achieve them, and explain how best to use this book.

Why Do Linguists Need to Learn How to Program?

Programming is an extremely useful skill in many areas of linguistics and in other language-related fields like speech and hearing sciences, psychology, psycholinguistics, and quantitative literary studies.

Within linguistics, it used to be the case that programming skills were required only for computational linguists, but this is far from true these days. Programming now is used in phonology, syntax, morphology, semantics, pragmatics, psycholinguistics, phonetics, discourse analysis, essentially every area of linguistic investigation. This change reflects broader methodological changes in the field, a response to the fact that (i) more and more data are available electronically, and (ii) we have much richer techniques for examining and manipulating massive amounts of electronic data.

Here are some examples of what you can do with fairly modest programming skills:

- Build a simple list of occurring words from a text file written in some language along with the frequency of those words.
- Find items for psycholinguistic or phonetic experiments from text resources, e.g., frequent two-syllable words that begin with a three-consonant cluster and don't otherwise contain nasal consonants.
- Construct every possible one-syllable word given a set of possible onsets, vowels, and codas.
- Construct every possible two-word compound given a list of words.
- Find the average number of words per sentence from a text corpus and find the longest sentence in that corpus.

- Build a model of syllabification for some language and syllabify candidate words.
- Find the average length and amplitude for some set of sound files.

This is just a small sample of the sorts of things that programming might help you do as part of your work with language.

Why Is Python a Good Choice?

There is a wide variety of programming languages you might choose to learn and work with. Every language has its virtues, the things that make it a good choice for this or that purpose. The choice of language is a function of three basic factors.

First, what is it that you want to do? Some languages are optimized for different sorts of goals, e.g., writing applications, developing system tools, and scripting. Second, what kind of programming experience do you have? Some languages are relatively easy to pick up and others, not so much. Finally, what kind of programming style do you want to use? Different languages lend themselves to different kinds of programming approaches, e.g., procedural, functional, object-oriented, and parallel.

In this book, we assume that you want to write programs that will let you answer questions about language, programs that may be run only by you. Thus programming languages like `java`, objective `c`, or `c#` that allow you to write full applications are not optimal choices. In addition, we assume that you have little or no prior programming experience. Thus interesting and challenging programming systems like `Haskell`, `Lisp`, or `Prolog` are best left for later.

Consequently, we will use the Python programming language. There are specific reasons for this choice:

- Python is *extremely* widely used, so you most likely have friends, colleagues, or classmates who use it and so can help you if needed. There are myriad resources on the web that can help as well.
- Python has stable and clear semantics. This means that the meaning and use of Python elements is clearly defined, and you can rely on your programs working as you intend. Similarly, program examples in this text should work exactly as shown on your system.
- Modules. There are tons of optional modules that others have written with useful functions and objects to simplify your programming tasks.
- Python is practical. Python is widely used in many areas, so your language-related programming skills may help you in other domains.

- NLTK. The Natural Language Toolkit is a freely available suite of modules tailored for working with language. You can use these for high-level statistical natural language processing or for simple language-related tasks.

There are some challenges to working with Python too. The biggest is that Python is an object-oriented (OO) programming language, and using it requires that you at least understand what objects are. The language thus lends itself to an OO programming style, but you need not use that specific style at first.

Our approach in this book is to first ignore OO aspects of the language. As we proceed, we introduce what you need to know of the OO system to make use of the concepts introduced to that point. Finally, in the latter part of the book, we explain OO programming in depth. Ultimately, we leave it up to you to decide how much OO programming is necessary for your programming goals.

How This Book Is Different

There are any number of book-length introductions to Python out there. How is this one different?

First, this book is written for linguists and other people who work with language data. What this means is that we give you examples that should make sense to you. If you have specific programs you want to write right away, you may even find some snippet here that (almost) does what you want and can be easily adapted to your purposes.

We continue with the language focus throughout the book. This means that, rather than trying to learn some programming concept exemplified in a program that has no relation to your goals, you can learn critical concepts with programs that are comprehensible and, we hope, useful as well.

Most chapters conclude with an extended example that shows how a larger program should be developed, using the concepts learned in the chapter, and with a focus on some language-related task. All chapters conclude with exercises, and all the exercises are linguistic in their orientation as well.

Another consequence of this approach is that we take a linguistic approach to the structure of Python. As much as possible, we treat it as a language with a syntax and a semantics.

Overview of the Book

The structure of this book is roughly as follows.

First, we introduce the basic syntax and semantics of the language: the primitive elements of the language and how those elements can be combined to make legal statements and larger structures.

As we introduce those topics, we elaborate the imperative semantics of the language, how we can use the specific Python language components covered to achieve different programming goals.

We next consider specific language-related tasks in depth: searching text, manipulating text, internet data, and text encodings.

Finally, we conclude with more advanced discussions of Python objects and OO programming generally, GUI programming, and functional programming. There is a brief appendix outlining the NLTK system as well.

How to Use the Book

The most important thing about using this book is that you should run the programs as you proceed. You can either download them from the book website[1] or type them in yourself.

If you have the patience for it, it is much better to type the programs in. This will really help you to notice aspects of the code you might not otherwise see and make the coding process more familiar to you. This will be frustrating and you will make errors as you type things in, but figuring out these errors will really help you learn the material.

It's also extremely important that you understand the code. You should make sure you understand each code snippet that's given before you go on – how it works, why it does what it does.

To this end, another really useful thing to do as you proceed is to play with the code. Tweak it in different ways, either to do something you'd rather it do or just to see what happens. (As you'll see below, the only time you *don't* want to do this is when you're performing file input – output operations where you can accidentally damage or lose things on your computer.)

Acknowledgments

Thanks to Sam Johnston, Dan Jurafsky, Nick Kloehn, and Ben Martin for never letting me forget the virtues of Python.

Thanks to the students in my Linguistics 408/508 course for letting me try out this material with them. Thanks to Damian Romero Diaz for helpful comments.

Thanks to Diane Ohala and Joey Rousos-Hammond for their love and support throughout.

All errors are my own.

[1] `www.u.arizona.edu/~hammond/`