Texts in Computational Science and Engineering

6

Editors

More information about this series at http://www.springer.com/series/5151

Hans Petter Langtangen

A Primer on Scientific Programming with Python

5th Edition

Springer

Hans Petter Langtangen
Simula Research Laboratory
Fornebu, Norway

On leave from:

Department of Informatics
University of Oslo
Oslo, Norway

ISSN 1611-0994
Texts in Computational Science and Engineering
ISBN 978-3-662-49886-6 ISBN 978-3-662-49887-3 (eBook)
DOI 10.1007/978-3-662-49887-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2016945366

Mathematic Subject Classification to (2010): 26-01, 34A05, 34A30, 34A34, 39-01, 40-01, 65D15, 65D25, 65D30, 68-01, 68N01, 68N19, 68N30, 70-01, 92D25, 97-04, 97U50

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer Berlin Heidelberg

Preface

The aim of this book is to teach computer programming using examples from mathematics and the natural sciences. We have chosen to use the Python programming language because it combines remarkable expressive power with very clean, simple, and compact syntax. Python is easy to learn and very well suited for an introduction to computer programming. Python is also quite similar to MATLAB and a good language for doing mathematical computing. It is easy to combine Python with compiled languages, like Fortran, C, and C++, which are widely used languages for scientific computations.

The examples in this book integrate programming with applications to mathematics, physics, biology, and finance. The reader is expected to have knowledge of basic one-variable calculus as taught in mathematics-intensive programs in high schools. It is certainly an advantage to take a university calculus course in parallel, preferably containing both classical and numerical aspects of calculus. Although not strictly required, a background in high school physics makes many of the examples more meaningful.

Many introductory programming books are quite compact and focus on listing functionality of a programming language. However, learning to program is learning how to *think* as a programmer. This book has its main focus on the thinking process, or equivalently: programming as a problem solving technique. That is why most of the pages are devoted to case studies in programming, where we define a problem and explain how to create the corresponding program. New constructions and programming styles (what we could call theory) is also usually introduced via examples. Particular attention is paid to verification of programs and to finding errors. These topics are very demanding for mathematical software, because the unavoidable numerical approximation errors are possibly mixed with programming mistakes.

By studying the many examples in the book, I hope readers will learn how to think right and thereby write programs in a quicker and more reliable way. Remember, nobody can learn programming by just reading – one has to solve a large amount of exercises hands on. The book is therefore full of exercises of various types: modifications of existing examples, completely new problems, or debugging of given programs.

To work with this book, I recommend using Python version 2.7. For Chaps. 5–9 and Appendices A–E, you need the NumPy and Matplotlib packages, preferably

also the IPython and SciTools packages, and for Appendix G, Cython is required. Other packages used in the text are nose and `sympy`. Section H.1 has more information on how you can get access to Python and the mentioned packages.

There is a web page associated with this book, http://hplgit.github.io/scipro-primer, containing all the example programs from the book as well as information on installation of the software on various platforms.

Python version 2 or 3? A common problem among Python programmers is to choose between version 2 or 3, which at the time of this writing means choosing between version 2.7 and 3.5. A common recommendation is to go for Python 3, because this is the version that will be further developed in the future. However, there is a problem that much useful mathematical software in Python has not yet been ported to Python 3. Therefore, Python version 2.7 is the most popular version for doing scientific computing, and that is why also this book applies version 2.7.

A widely used strategy for software developers who want to write Python code that works with both versions, is to develop a common version for Python 2 and 3. For the programs in this book, a common version can easily be produced by first developing for version 2.7 and then *automatically* convert the code by running the `futurize` program. Section 4.10 demonstrates how this is done in simple cases.

The Python 2.7 code in this book sticks to all modern constructions that are backported from version 3 such that the code becomes as close as possible to the equivalent Python 3 code. At any time, you can just run `futurize` to see the differences between your Python 2.7 version and the corresponding Python 3.5 version.

Contents Chapter 1 introduces variables, objects, modules, and text formatting through examples concerning evaluation of mathematical formulas. Chapter 2 presents programming with `while` and `for` loops as well as lists, including nested lists. The next chapter deals with two other fundamental concepts in programming: functions and `if-else` tests.

How to read data into programs and deal with errors in input are the subjects of Chap. 4. Chapter 5 introduces arrays and array computing (including vectorization) and how this is used for plotting $y = f(x)$ curves and making animation of curves. Many of the examples in the first five chapters are strongly related. Typically, formulas from the first chapter are used to produce tables of numbers in the second chapter. Then the formulas are encapsulated in functions in the third chapter. In the next chapter, the input to the functions are fetched from the command line, and validity checks of the input are added. The formulas are then shown as graphs in Chap. 5. After having studied Chaps. 1–5, the reader should have enough knowledge of programming to solve mathematical problems by what many refer to as "MATLAB-style" programming.

Chapter 6 explains how to work with dictionaries and strings, especially for interpreting text data in files and storing the extracted information in flexible data structures. Class programming, including user-defined types for mathematical computations (with overloaded operators), is introduced in Chap. 7. Chapter 8 deals with random numbers and statistical computing with applications to games and random walks. Object-oriented programming, in the meaning of class hierarchies and inheritance, is the subject of Chap. 9. The key examples here deal with building toolkits for numerical differentiation and integration as well as graphics.

Appendix A introduces mathematical modeling, using sequences and difference equations. Only programming concepts from Chaps. 1–5 are used in this appendix, the aim being to consolidate basic programming knowledge and apply it to mathematical problems. Some important mathematical topics are introduced via difference equations in a simple way: Newton's method, Taylor series, inverse functions, and dynamical systems.

Appendix B deals with functions on a mesh, numerical differentiation, and numerical integration. A simple introduction to ordinary differential equations and their numerical treatment is provided in Appendix C. Appendix D shows how a complete project in physics can be solved by mathematical modeling, numerical methods, and programming elements from Chaps. 1–5. This project is a good example on problem solving in computational science, where it is necessary to integrate physics, mathematics, numerics, and computer science.

How to create software for solving ordinary differential equations, using both function-based and object-oriented programming, is the subject of Appendix E. The material in this appendix brings together many parts of the book in the context of physical applications and differential equations.

Appendix F is devoted to the art of debugging, and in fact problem solving in general. Speeding up numerical computations in Python by migrating code to C via Cython is exemplified in Appendix G. Finally, Appendix H deals with various more advanced technical topics.

Most of the examples and exercises in this book are quite short. However, many of the exercises are related, and together they form larger projects, for example on Fourier Series (3.21, 4.21, 4.22, 5.41, 5.42), numerical integration (3.11, 3.12, 5.49, 5.50, A.12), Taylor series (3.37, 5.32, 5.39, A.14, A.15, 7.23), piecewise constant functions (3.29–3.33, 5.34, 5.47, 5.48, 7.19–7.21), inverse functions (E.17–E.20), falling objects (E.8, E.9, E.38, E.39), oscillatory population growth (A.19, A.21, A.22, A.23), epidemic disease modeling (E.41–E.48), optimization and finance (A.24, 8.42, 8.43), statistics and probability (4.24, 4.25, 8.23, 8.24), hazard games (8.8–8.14), random walk and statistical physics (8.32–8.40), noisy data analysis (8.44–8.46), numerical methods (5.25–5.27, 7.8, 7.9, A.9, 7.22, 9.15–9.17, E.30–E.37), building a calculus calculator (7.34, 9.18, 9.19), and creating a toolkit for simulating vibrating engineering systems (E.50–E.55).

Chapters 1–9 together with Appendices A and E have from 2007 formed the core of an introductory first semester bachelor course on scientific programming at the University of Oslo (INF1100, 10 ECTS credits).

Changes from the fourth to the fifth edition Substantial changes were introduced in the fourth edition, and the fifth edition is primarily a consolidation of those changes. Many typos have been corrected and many explanations and exercises have been improved. The emphasis on unit tests and test functions, especially in exercises, is stronger than in the previous edition. Symbolic computation with the aid of SymPy is used to a larger extent and integrated with numerical computing throughout the book. All classes are now new-style (instead of old-style/classic as in previous editions). Examples on Matplotlib do not use the `pylab` module anymore, but `pyplot` and MATLAB-like syntax is still favored to ease the transition between Python and MATLAB. The concept of closures is more explicit than in earlier editions (see the new Sect. 7.1.7) since this is a handy and popular construc-

tion much used in the scientific Python community. We also discuss the difference between Python 2 and 3 and demonstrate how to use the `future` module to write code that runs under both versions.

The most substantial new material in the fifth edition appears toward the end of Chap. 5 and regards high-performance computing, linear algebra, and visualization of scalar and vector fields. Although this material is not used elsewhere in the book, many readers have requested basic recipes when going from one to two variables or from vectors to matrices later when solving more advanced problems and using the book as their programming reference. The new matrial in Chap. 5 was written jointly with Dr. Øyvind Ryan.

Acknowledgments This book was born out of stimulating discussions with my close colleague Aslak Tveito, and he started writing what is now Appendix B and C. The whole book project and the associated university course were critically dependent on Aslak's enthusiastic role back in 2007. The continuous support from Aslak regarding my book projects is much appreciated and contributes greatly to my strong motivation. Another key contributor in the early days was Ilmar Wilbers. He made extensive efforts with assisting the book project and establishing the university course INF1100. I feel that without Ilmar and his solutions to numerous technical problems the first edition of the book would never have been completed. Johannes H. Ring also deserves special acknowledgment for the development of the Easyviz graphics tool back in the days when Python plotting was a hassle, and later for his maintenance of software associated with the book.

Professor Loyce Adams studied the entire book, solved all the exercises, found numerous errors, and suggested many improvements. Her contributions are so much appreciated. More recently, Helmut Büch worked extremely carefully through all details in Chaps. 1–6, tested the software, found many typos, and asked critical questions that led to lots of significant improvements. I am so thankful for all his efforts and for his enthusiasm during the preparations of the fourth edition. The fifth edition has benefited much from Hakki Eres' careful examination of the fourth edition. He found several typos and code errors, some of which go back to the first edition.

Special thanks go to Geir Kjetil Sandve for being the primary author of the computational bioinformatics examples in Sects. 3.3, 6.5, 8.3.4, and 9.5, with contributions from Sveinung Gundersen, Ksenia Khelik, Halfdan Rydbeck, and Kai Trengereid. I am also greatful to Øyvind Ryan's work with linear algebra and visualization of scalar and vector fields in Chap. 5.

Several people have contributed with suggestions for improvements of the text, the exercises, and the associated software. I will in particular mention Ingrid Eide, Ståle Zerener Haugnæss, Kristian Hiorth, Timothy Keough, Arve Knudsen, Espen Kristensen, Tobias Vidarssønn Langhoff, Martin Vonheim Larsen, Kine Veronica Lund, Solveig Masvie, Håkon Møller, Rebekka Mørken, Mathias Nedrebø, Marit Sandstad, Helene Norheim Semmerud, Lars Storjord, Fredrik Heffer Valdmanis, and Torkil Vederhus. Hakon Adler is greatly acknowledged for his careful reading of early various versions of the manuscript. Many thanks go to the professors Fred Espen Bent, Ørnulf Borgan, Geir Dahl, Knut Mørken, and Geir Pedersen for formulating several exciting exercises from various application fields. I also appreciate the cover image made by my good friend Jan Olav Langseth.

This book and the associated course are parts of a comprehensive and successful reform at the University of Oslo, called *Computing in Science Education*. The goal of the reform is to integrate computer programming and simulation in all bachelor courses in natural science where mathematical models are used. The present book lays the foundation for the modern computerized problem solving technique to be applied in later courses. It has been extremely inspiring to work closely with the driving forces behind this reform, especially the professors Morten Hjorth-Jensen, Anders Malthe-Sørenssen, Knut Mørken, and Arnt Inge Vistnes.

The excellent assistance from the Springer system over the years, in particular Martin Peters, Thanh-Ha Le Thi, Ruth Allewelt, Peggy Glauch-Ruge, Nadja Kroke, Thomas Schmidt, Patrick Waltemate, Donatas Akmanavicius, and Yvonne Schlatter, is highly appreciated, and ensured a smooth and rapid production of all editions of this book.

Oslo, February 2016 Hans Petter Langtangen

Contents

List of Exercises