

Editors

Timothy J. Barth

Michael Griebel

David E. Keyes

Risto M. Nieminen

Dirk Roose

Tamar Schlick

Hans Petter Langtangen

A Primer on Scientific Programming with Python

Hans Petter Langtangen
Simula Research Laboratory
Martin Linges vei 17
1325 Lysaker, Fornebu
Norway
hpl@simula.no

On leave from:

Department of Informatics
University of Oslo
P.O. Box 1080 Blindern
0316 Oslo, Norway
<http://folk.uio.no/hpl>

ISSN 1611-0994
ISBN 978-3-642-02474-0 e-ISBN 978-3-642-02475-7
DOI 10.1007/978-3-642-02475-7
Springer Dordrecht Heidelberg London New York

Library of Congress Control Number: 2009931367

Mathematics Subject Classification (2000): 26-01, 34A05, 34A30, 34A34, 39-01, 40-01, 65D15, 65D25, 65D30, 68-01, 68N01, 68N19, 68N30, 70-01, 92D25, 97-04, 97U50

© Springer-Verlag Berlin Heidelberg 2009

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Production: le-tex publishing services GmbH, Leipzig, Germany
Cover design: deblik, Berlin

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

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 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. A seamless integration of Python with Java is offered by a special version of Python called Jython.

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. Special attention is paid to verification of programs and to finding errors. These topics are very demanding for mathematical software, because we have approximation errors possibly mixed with programming errors.

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. Therefore, the book is full of exercises of various types: modifications of existing examples, completely new problems, or debugging of given programs.

To work with this book, you need to install Python version 2.6. In Chapter 4 and later chapters, you also need the NumPy and SciTools packages. To make curve plots with SciTools, you must have a plotting program, for example, Gnuplot or Matplotlib. There is a web page associated with this book, <http://www.simula.no/intro-programming>, which lists the software you need and explains briefly how to install it. On this page, you will also find all the files associated with the program examples in this book. Download `book-examples.tar.gz`, store this file in some folder of your choice, and unpack it using WinZip on Windows or the command `tar xzf book-examples.tar.gz` on Linux and Mac. This unpacking yields a folder `src` with subfolders for the various chapters in the book.

Contents. Chapter 1 introduces variables, objects, modules, and text formatting through examples concerning evaluation of mathematical formulas. Chapter 2 presents fundamental elements of programming: loops, lists, and functions. This is a comprehensive and important chapter that should be digested before proceeding. How to read data into programs and deal with errors in input are the subjects of Chapter 3. Many of the examples in the first three chapters are strongly related. Typically, formulas from the first chapter are encapsulated in functions in the second chapter, and in the third chapter the input to the functions are fetched from the command line or from a question-answer dialog with the user, and validity of the data is checked. Chapter 4 introduces arrays and array computing (including vectorization) and how this is used for plotting $y = f(x)$ curves. After the first four chapters, the reader should have enough knowledge of programming to solve mathematical problems by “Matlab-style” programming.

Chapter 5 introduces mathematical modeling, using sequences and difference equations. We also treat sound as a sequence. No new programming concepts are introduced in this chapter, the aim being to consolidate the programming knowledge and apply it to mathematical problems.

Chapter 6 explains how to work with files and text data. Class programming, including user-defined types for mathematical computations (with overloaded operators), is introduced in Chapter 7. Chapter 8 deals with random numbers and statistical computing with applications to games and random walk. Object-oriented programming (class hierarchies and inheritance) is the subject of Chapter 9. The

key examples here deal with building toolkits for graphics and for numerical differentiation, integration, and solution of ordinary differential equations.

Appendix A deals with functions on a mesh, numerical differentiation, and numerical integration. The next appendix gives an introduction to numerical solution of ordinary differential equations. These two appendices provide the theoretical background on numerical methods that are much in use in Chapters 7 and 9. Moreover, the appendices exemplify basic programming from the first four chapters.

Appendix C shows how a complete project in physics can be solved by mathematical modeling, numerical methods, and programming elements from the first four chapters. This project is a good example on problem solving in computational science, where it is necessary to integrate physics, mathematics, numerics, and computer science.

Appendix D is devoted to the art of debugging, and in fact problem solving in general, while Appendix E deals with various more advanced technical topics.

Most of the examples and exercises in this book are quite compact and limited. However, many of the exercises are related, and together they form larger projects in science, for example on Fourier Series (1.13, 2.39, 3.17, 3.18, 3.19, 4.20), Taylor series (2.38, 4.16, 4.18, 5.16, 5.17, 7.31), falling objects (7.25, 9.30, 7.26, 9.31, 9.32, 9.34), oscillatory population growth (5.21, 5.22, 6.28, 7.41, 7.42), visualization of web data (6.22, 6.23, 6.24, 6.25), graphics and animation (9.36, 9.37, 9.38, 9.39), optimization and finance (5.23, 8.42, 8.43), statistics and probability (3.25, 3.26, 3.27, 8.19, 8.20, 8.21), random walk and statistical physics (8.33, 8.34, 8.35, 8.36, 8.37, 8.38, 8.39, 8.40), noisy data analysis (8.44, 8.45, 8.46, 8.47, 9.40), numerical methods (5.12, 7.9, 7.22, 9.15, 9.16, 9.26, 9.27, 9.28), building a calculus calculator (9.41, 9.42, 9.43, 9.44), and creating a toolkit for simulating oscillatory systems (9.45–9.52).

Chapters 1–9 and Appendix C form the core of an introductory first-semester course on scientific programming (INF1100) at the University of Oslo. Normally, each chapter is suited for a 2×45 min lecture, but Chapters 2 and 7 are challenging, and each of them have consumed two lectures in the course.

Acknowledgments. First, I want to express my thanks to Aslak Tveito for his enthusiastic role in the initiation of this book project and for writing Appendices A and B about numerical methods. Without Aslak there would be no book. Another key contributor is Ilmar Wilbers. His extensive efforts with assisting the book project and teaching the associated course (INF1100) at the University of Oslo are greatly appreciated. Without Ilmar and his solutions to numerous technical problems the book would never have been completed. Johannes H. Ring also deserves a special acknowledgment for the development of the Easyviz

graphics tool, which is much used throughout this book, and for his careful maintenance and support of software associated with this book.

Several people have helped to make substantial improvements of the text, the exercises, and the associated software infrastructure. The author is thankful to Ingrid Eide, Tobias Vidarssønn Langhoff, Mathias Nedrebø, Arve Knudsen, Marit Sandstad, Lars Storjord, Fredrik Heffer Valdmanis, and Torkil Vederhus for their contributions. Hakon Adler is greatly acknowledged for his careful reading of various versions of the manuscript. The professors Fred Espen Bent, Ørnulf Borgan, Geir Dahl, Knut Mørken, and Geir Pedersen have contributed with many exciting exercises from various application fields. Great thanks also go to Jan Olav Langseth for creating the cover image.

This book and the associated course are parts of a comprehensive reform at the University of Oslo, called *Computers 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 with the driving forces behind this reform, in particular the professors Morten Hjorth-Jensen, Anders Malthe-Sørenssen, Knut Mørken, and Arnt Inge Vistnes.

The excellent assistance from the Springer and Le-Tex teams, consisting of Martin Peters, Thanh-Hà Lê Thi, Ruth Allewelt, Peggy Glauch-Ruge, Nadja Kroke, and Thomas Schmidt, is highly appreciated and ensured a smooth and rapid production of this book.

Oslo, May 2009

Hans Petter Langtangen

Contents

1	Computing with Formulas	1
1.1	The First Programming Encounter: A Formula	1
1.1.1	Using a Program as a Calculator	2
1.1.2	About Programs and Programming	2
1.1.3	Tools for Writing Programs	3
1.1.4	Using Idle to Write the Program	4
1.1.5	How to Run the Program	7
1.1.6	Verifying the Result	8
1.1.7	Using Variables	8
1.1.8	Names of Variables	9
1.1.9	Reserved Words in Python	10
1.1.10	Comments	10
1.1.11	Formatting Text and Numbers	11
1.2	Computer Science Glossary	13
1.3	Another Formula: Celsius-Fahrenheit Conversion	18
1.3.1	Potential Error: Integer Division	19
1.3.2	Objects in Python	20
1.3.3	Avoiding Integer Division	21
1.3.4	Arithmetic Operators and Precedence	21
1.4	Evaluating Standard Mathematical Functions	22
1.4.1	Example: Using the Square Root Function	22
1.4.2	Example: Using More Mathematical Functions	25
1.4.3	A First Glimpse of Round-Off Errors	25
1.5	Interactive Computing	26
1.5.1	Calculating with Formulas in the Interactive Shell	27
1.5.2	Type Conversion	28
1.5.3	IPython	29
1.6	Complex Numbers	31

1.6.1	Complex Arithmetics in Python	32
1.6.2	Complex Functions in Python	32
1.6.3	Unified Treatment of Complex and Real Functions	33
1.7	Summary	35
1.7.1	Chapter Topics	35
1.7.2	Summarizing Example: Trajectory of a Ball	38
1.7.3	About Typesetting Conventions in This Book ..	39
1.8	Exercises	40
2	Basic Constructions	51
2.1	Loops and Lists for Tabular Data	51
2.1.1	A Naive Solution	51
2.1.2	While Loops	52
2.1.3	Boolean Expressions	54
2.1.4	Lists	56
2.1.5	For Loops	58
2.1.6	Alternative Implementations with Lists and Loops	60
2.1.7	Nested Lists	64
2.1.8	Printing Objects	65
2.1.9	Extracting Sublists	66
2.1.10	Traversing Nested Lists	68
2.1.11	Tuples	70
2.2	Functions	71
2.2.1	Functions of One Variable	71
2.2.2	Local and Global Variables	73
2.2.3	Multiple Arguments	75
2.2.4	Multiple Return Values	77
2.2.5	Functions with No Return Values	79
2.2.6	Keyword Arguments	80
2.2.7	Doc Strings	83
2.2.8	Function Input and Output	84
2.2.9	Functions as Arguments to Functions	84
2.2.10	The Main Program	86
2.2.11	Lambda Functions	87
2.3	If Tests	88
2.4	Summary	91
2.4.1	Chapter Topics	91
2.4.2	Summarizing Example: Tabulate a Function	94
2.4.3	How to Find More Python Information	98
2.5	Exercises	99
3	Input Data and Error Handling	119
3.1	Asking Questions and Reading Answers	120

3.1.1	Reading Keyboard Input	120
3.1.2	The Magic “eval” Function	121
3.1.3	The Magic “exec” Function	125
3.1.4	Turning String Expressions into Functions	126
3.2	Reading from the Command Line	127
3.2.1	Providing Input on the Command Line	127
3.2.2	A Variable Number of Command-Line Arguments	128
3.2.3	More on Command-Line Arguments	129
3.2.4	Option–Value Pairs on the Command Line	130
3.3	Handling Errors	132
3.3.1	Exception Handling	133
3.3.2	Raising Exceptions	136
3.4	A Glimpse of Graphical User Interfaces	139
3.5	Making Modules	141
3.5.1	Example: Compund Interest Formulas	142
3.5.2	Collecting Functions in a Module File	143
3.5.3	Using Modules	148
3.6	Summary	150
3.6.1	Chapter Topics	150
3.6.2	Summarizing Example: Bisection Root Finding .	152
3.7	Exercises	160
4	Array Computing and Curve Plotting	169
4.1	Vectors	170
4.1.1	The Vector Concept	170
4.1.2	Mathematical Operations on Vectors	171
4.1.3	Vector Arithmetics and Vector Functions	173
4.2	Arrays in Python Programs	175
4.2.1	Using Lists for Collecting Function Data	175
4.2.2	Basics of Numerical Python Arrays	176
4.2.3	Computing Coordinates and Function Values ...	177
4.2.4	Vectorization	178
4.3	Curve Plotting	179
4.3.1	The SciTools and Easyviz Packages	180
4.3.2	Plotting a Single Curve	181
4.3.3	Decorating the Plot	183
4.3.4	Plotting Multiple Curves	183
4.3.5	Controlling Line Styles	185
4.3.6	Interactive Plotting Sessions	189
4.3.7	Making Animations	190
4.3.8	Advanced Easyviz Topics	193
4.3.9	Curves in Pure Text	198
4.4	Plotting Difficulties	199
4.4.1	Piecewisely Defined Functions	199

4.4.2	Rapidly Varying Functions	205
4.4.3	Vectorizing StringFunction Objects	206
4.5	More on Numerical Python Arrays	207
4.5.1	Copying Arrays	207
4.5.2	In-Place Arithmetics	207
4.5.3	Allocating Arrays	208
4.5.4	Generalized Indexing	209
4.5.5	Testing for the Array Type	210
4.5.6	Equally Spaced Numbers	211
4.5.7	Compact Syntax for Array Generation.....	212
4.5.8	Shape Manipulation.....	212
4.6	Higher-Dimensional Arrays	213
4.6.1	Matrices and Arrays	213
4.6.2	Two-Dimensional Numerical Python Arrays....	214
4.6.3	Array Computing	216
4.6.4	Two-Dimensional Arrays and Functions of Two Variables	217
4.6.5	Matrix Objects	217
4.7	Summary	219
4.7.1	Chapter Topics	219
4.7.2	Summarizing Example: Animating a Function ..	220
4.8	Exercises	225
5	Sequences and Difference Equations	235
5.1	Mathematical Models Based on Difference Equations ..	236
5.1.1	Interest Rates	237
5.1.2	The Factorial as a Difference Equation	239
5.1.3	Fibonacci Numbers	240
5.1.4	Growth of a Population.....	241
5.1.5	Logistic Growth	242
5.1.6	Payback of a Loan	244
5.1.7	Taylor Series as a Difference Equation	245
5.1.8	Making a Living from a Fortune	246
5.1.9	Newton's Method	247
5.1.10	The Inverse of a Function	251
5.2	Programming with Sound	253
5.2.1	Writing Sound to File	253
5.2.2	Reading Sound from File	254
5.2.3	Playing Many Notes	255
5.3	Summary	256
5.3.1	Chapter Topics	256
5.3.2	Summarizing Example: Music of a Sequence....	257
5.4	Exercises	260
6	Files, Strings, and Dictionaries.....	269

6.1	Reading Data from File	269
6.1.1	Reading a File Line by Line	270
6.1.2	Reading a Mixture of Text and Numbers	273
6.1.3	What Is a File, Really?	274
6.2	Dictionaries	278
6.2.1	Making Dictionaries	278
6.2.2	Dictionary Operations	279
6.2.3	Example: Polynomials as Dictionaries	280
6.2.4	Example: File Data in Dictionaries	282
6.2.5	Example: File Data in Nested Dictionaries	283
6.2.6	Example: Comparing Stock Prices	287
6.3	Strings	291
6.3.1	Common Operations on Strings	292
6.3.2	Example: Reading Pairs of Numbers	295
6.3.3	Example: Reading Coordinates	298
6.4	Reading Data from Web Pages	300
6.4.1	About Web Pages	300
6.4.2	How to Access Web Pages in Programs	302
6.4.3	Example: Reading Pure Text Files	302
6.4.4	Example: Extracting Data from an HTML Page	304
6.5	Writing Data to File	308
6.5.1	Example: Writing a Table to File	309
6.5.2	Standard Input and Output as File Objects	310
6.5.3	Reading and Writing Spreadsheet Files	312
6.6	Summary	317
6.6.1	Chapter Topics	317
6.6.2	Summarizing Example: A File Database	319
6.7	Exercises	323
7	Introduction to Classes	337
7.1	Simple Function Classes	338
7.1.1	Problem: Functions with Parameters	338
7.1.2	Representing a Function as a Class	340
7.1.3	Another Function Class Example	346
7.1.4	Alternative Function Class Implementations	347
7.1.5	Making Classes Without the Class Construct	349
7.2	More Examples on Classes	352
7.2.1	Bank Accounts	352
7.2.2	Phone Book	354
7.2.3	A Circle	355
7.3	Special Methods	356
7.3.1	The Call Special Method	357
7.3.2	Example: Automagic Differentiation	357
7.3.3	Example: Automagic Integration	360
7.3.4	Turning an Instance into a String	362

7.3.5	Example: Phone Book with Special Methods . . .	363
7.3.6	Adding Objects	365
7.3.7	Example: Class for Polynomials	365
7.3.8	Arithmetic Operations and Other Special Methods	369
7.3.9	More on Special Methods for String Conversion .	370
7.4	Example: Solution of Differential Equations	372
7.4.1	A Function for Solving ODEs	373
7.4.2	A Class for Solving ODEs	374
7.4.3	Verifying the Implementation	376
7.4.4	Example: Logistic Growth	377
7.5	Example: Class for Vectors in the Plane	378
7.5.1	Some Mathematical Operations on Vectors	378
7.5.2	Implementation	378
7.5.3	Usage	380
7.6	Example: Class for Complex Numbers	382
7.6.1	Implementation	382
7.6.2	Illegal Operations	383
7.6.3	Mixing Complex and Real Numbers	384
7.6.4	Special Methods for “Right” Operands	387
7.6.5	Inspecting Instances	388
7.7	Static Methods and Attributes	389
7.8	Summary	391
7.8.1	Chapter Topics	391
7.8.2	Summarizing Example: Interval Arithmetics . . .	392
7.9	Exercises	397
8	Random Numbers and Simple Games	417
8.1	Drawing Random Numbers	418
8.1.1	The Seed	418
8.1.2	Uniformly Distributed Random Numbers	419
8.1.3	Visualizing the Distribution	420
8.1.4	Vectorized Drawing of Random Numbers	421
8.1.5	Computing the Mean and Standard Deviation . .	422
8.1.6	The Gaussian or Normal Distribution	423
8.2	Drawing Integers	424
8.2.1	Random Integer Functions	425
8.2.2	Example: Throwing a Die	426
8.2.3	Drawing a Random Element from a List	427
8.2.4	Example: Drawing Cards from a Deck	427
8.2.5	Example: Class Implementation of a Deck	429
8.3	Computing Probabilities	432
8.3.1	Principles of Monte Carlo Simulation	432
8.3.2	Example: Throwing Dice	433
8.3.3	Example: Drawing Balls from a Hat	435

8.3.4	Example: Policies for Limiting Population Growth	437
8.4	Simple Games	440
8.4.1	Guessing a Number	440
8.4.2	Rolling Two Dice	440
8.5	Monte Carlo Integration	443
8.5.1	Standard Monte Carlo Integration	443
8.5.2	Computing Areas by Throwing Random Points ..	446
8.6	Random Walk in One Space Dimension	447
8.6.1	Basic Implementation	448
8.6.2	Visualization	449
8.6.3	Random Walk as a Difference Equation	449
8.6.4	Computing Statistics of the Particle Positions ..	450
8.6.5	Vectorized Implementation	451
8.7	Random Walk in Two Space Dimensions	453
8.7.1	Basic Implementation	453
8.7.2	Vectorized Implementation	455
8.8	Summary	456
8.8.1	Chapter Topics	456
8.8.2	Summarizing Example: Random Growth	457
8.9	Exercises	463
9	Object-Oriented Programming	479
9.1	Inheritance and Class Hierarchies	479
9.1.1	A Class for Straight Lines	480
9.1.2	A First Try on a Class for Parabolas	481
9.1.3	A Class for Parabolas Using Inheritance	481
9.1.4	Checking the Class Type	483
9.1.5	Attribute versus Inheritance	484
9.1.6	Extending versus Restricting Functionality	485
9.1.7	Superclass for Defining an Interface	486
9.2	Class Hierarchy for Numerical Differentiation	488
9.2.1	Classes for Differentiation	488
9.2.2	A Flexible Main Program	491
9.2.3	Extensions	492
9.2.4	Alternative Implementation via Functions	495
9.2.5	Alternative Implementation via Functional Programming	496
9.2.6	Alternative Implementation via a Single Class ..	497
9.3	Class Hierarchy for Numerical Integration	499
9.3.1	Numerical Integration Methods	499
9.3.2	Classes for Integration	501
9.3.3	Using the Class Hierarchy	504
9.3.4	About Object-Oriented Programming	507
9.4	Class Hierarchy for Numerical Methods for ODEs	508

9.4.1	Mathematical Problem	508
9.4.2	Numerical Methods	510
9.4.3	The ODE Solver Class Hierarchy	511
9.4.4	The Backward Euler Method	515
9.4.5	Verification	518
9.4.6	Application 1: $u' = u$	518
9.4.7	Application 2: The Logistic Equation	519
9.4.8	Application 3: An Oscillating System	521
9.4.9	Application 4: The Trajectory of a Ball	523
9.5	Class Hierarchy for Geometric Shapes	525
9.5.1	Using the Class Hierarchy	526
9.5.2	Overall Design of the Class Hierarchy	527
9.5.3	The Drawing Tool	529
9.5.4	Implementation of Shape Classes	530
9.5.5	Scaling, Translating, and Rotating a Figure	534
9.6	Summary	538
9.6.1	Chapter Topics	538
9.6.2	Summarizing Example: Input Data Reader	540
9.7	Exercises	546
A	Discrete Calculus	573
A.1	Discrete Functions	573
A.1.1	The Sine Function	574
A.1.2	Interpolation	576
A.1.3	Evaluating the Approximation	576
A.1.4	Generalization	577
A.2	Differentiation Becomes Finite Differences	579
A.2.1	Differentiating the Sine Function	580
A.2.2	Differences on a Mesh	580
A.2.3	Generalization	582
A.3	Integration Becomes Summation	583
A.3.1	Dividing into Subintervals	584
A.3.2	Integration on Subintervals	585
A.3.3	Adding the Subintervals	586
A.3.4	Generalization	587
A.4	Taylor Series	589
A.4.1	Approximating Functions Close to One Point ...	589
A.4.2	Approximating the Exponential Function	589
A.4.3	More Accurate Expansions	590
A.4.4	Accuracy of the Approximation	592
A.4.5	Derivatives Revisited	594
A.4.6	More Accurate Difference Approximations	595
A.4.7	Second-Order Derivatives	597
A.5	Exercises	599

B	Differential Equations	605
B.1	The Simplest Case	606
B.2	Exponential Growth	608
B.3	Logistic Growth	612
B.4	A General Ordinary Differential Equation	614
B.5	A Simple Pendulum	615
B.6	A Model for the Spread of a Disease	619
B.7	Exercises	621
C	A Complete Project	625
C.1	About the Problem: Motion and Forces in Physics	626
C.1.1	The Physical Problem	626
C.1.2	The Computational Algorithm	628
C.1.3	Derivation of the Mathematical Model	628
C.1.4	Derivation of the Algorithm	631
C.2	Program Development and Testing	632
C.2.1	Implementation	632
C.2.2	Callback Functionality	635
C.2.3	Making a Module	636
C.2.4	Verification	637
C.3	Visualization	639
C.3.1	Simultaneous Computation and Plotting	639
C.3.2	Some Applications	642
C.3.3	Remark on Choosing Δt	643
C.3.4	Comparing Several Quantities in Subplots	644
C.3.5	Comparing Approximate and Exact Solutions	645
C.3.6	Evolution of the Error as Δt Decreases	646
C.4	Exercises	649
D	Debugging	651
D.1	Using a Debugger	651
D.2	How to Debug	653
D.2.1	A Recipe for Program Writing and Debugging	654
D.2.2	Application of the Recipe	656
E	Technical Topics	669
E.1	Different Ways of Running Python Programs	669
E.1.1	Executing Python Programs in IPython	669
E.1.2	Executing Python Programs on Unix	669
E.1.3	Executing Python Programs on Windows	671
E.1.4	Executing Python Programs on Macintosh	673
E.1.5	Making a Complete Stand-Alone Executable	673
E.2	Integer and Float Division	673
E.3	Visualizing a Program with Lumpy	674
E.4	Doing Operating System Tasks in Python	675
E.5	Variable Number of Function Arguments	678

E.5.1	Variable Number of Positional Arguments	679
E.5.2	Variable Number of Keyword Arguments	681
E.6	Evaluating Program Efficiency	683
E.6.1	Making Time Measurements	683
E.6.2	Profiling Python Programs	685
Bibliography	687
Index	689

List of Exercises

Exercise 1.1	Compute $1+1$	42
Exercise 1.2	Write a “Hello, World!” program	43
Exercise 1.3	Convert from meters to British length units	43
Exercise 1.4	Compute the mass of various substances	43
Exercise 1.5	Compute the growth of money in a bank.....	43
Exercise 1.6	Find error(s) in a program	43
Exercise 1.7	Type in program text	43
Exercise 1.8	Type in programs and debug them	44
Exercise 1.9	Evaluate a Gaussian function.....	44
Exercise 1.10	Compute the air resistance on a football	45
Exercise 1.11	Define objects in IPython	45
Exercise 1.12	How to cook the perfect egg.....	46
Exercise 1.13	Evaluate a function defined by a sum.....	46
Exercise 1.14	Derive the trajectory of a ball	47
Exercise 1.15	Find errors in the coding of formulas	48
Exercise 1.16	Find errors in Python statements	48
Exercise 1.17	Find errors in the coding of a formula	49
Exercise 2.1	Make a Fahrenheit–Celsius conversion table	99
Exercise 2.2	Generate odd numbers	99
Exercise 2.3	Store odd numbers in a list	100
Exercise 2.4	Generate odd numbers by the range function	100
Exercise 2.5	Simulate operations on lists by hand	100
Exercise 2.6	Make a table of values from formula (1.1)	100
Exercise 2.7	Store values from formula (1.1) in lists.....	100
Exercise 2.8	Work with a list	100
Exercise 2.9	Generate equally spaced coordinates	100
Exercise 2.10	Use a list comprehension to solve Exer. 2.9.....	101
Exercise 2.11	Store data from Exer. 2.7 in a nested list	101
Exercise 2.12	Compute a mathematical sum	101
Exercise 2.13	Simulate a program by hand	101

Exercise 2.14	Use a for loop in Exer. 2.12	102
Exercise 2.15	Index a nested lists	102
Exercise 2.16	Construct a double for loop over a nested list	102
Exercise 2.17	Compute the area of an arbitrary triangle	102
Exercise 2.18	Compute the length of a path	102
Exercise 2.19	Approximate π	103
Exercise 2.20	Write a Fahrenheit-Celsius conversion table	103
Exercise 2.21	Convert nested list comprehensions to nested standard loops	103
Exercise 2.22	Write a Fahrenheit-Celsius conversion function ..	104
Exercise 2.23	Write some simple functions	104
Exercise 2.24	Write the program in Exer. 2.12 as a function ...	104
Exercise 2.25	Implement a Gaussian function	104
Exercise 2.26	Find the max and min values of a function	104
Exercise 2.27	Explore the Python Library Reference	105
Exercise 2.28	Make a function of the formula in Exer. 1.12	105
Exercise 2.29	Write a function for numerical differentiation	105
Exercise 2.30	Write a function for numerical integration	105
Exercise 2.31	Improve the formula in Exer. 2.30	106
Exercise 2.32	Compute a polynomial via a product	106
Exercise 2.33	Implement the factorial function	106
Exercise 2.34	Compute velocity and acceleration from position data; one dimension	107
Exercise 2.35	Compute velocity and acceleration from position data; two dimensions	107
Exercise 2.36	Express a step function as a Python function	107
Exercise 2.37	Rewrite a mathematical function	108
Exercise 2.38	Make a table for approximations of $\cos x$	108
Exercise 2.39	Implement Exer. 1.13 with a loop	109
Exercise 2.40	Determine the type of objects	109
Exercise 2.41	Implement the sum function	109
Exercise 2.42	Find the max/min elements in a list	110
Exercise 2.43	Demonstrate list functionality	110
Exercise 2.44	Write a sort function for a list of 4-tuples	110
Exercise 2.45	Find prime numbers	111
Exercise 2.46	Condense the program in Exer. 2.14	111
Exercise 2.47	Values of boolean expressions	112
Exercise 2.48	Explore round-off errors from a large number of inverse operations	112
Exercise 2.49	Explore what zero can be on a computer	112
Exercise 2.50	Resolve a problem with a function	113
Exercise 2.51	Compare two real numbers on a computer	113
Exercise 2.52	Use None in keyword arguments	114
Exercise 2.53	Improve the program from Ch. 2.4.2	114
Exercise 2.54	Interpret a code	114

Exercise 2.55	Explore problems with inaccurate indentation . . .	115
Exercise 2.56	Find an error in a program	115
Exercise 2.57	Find programming errors	116
Exercise 2.58	Simulate nested loops by hand	117
Exercise 2.59	Explore punctuation in Python programs	117
Exercise 2.60	Investigate a for loop over a changing list	117
Exercise 3.1	Make an interactive program	160
Exercise 3.2	Read from the command line in Exer. 3.1	160
Exercise 3.3	Use exceptions in Exer. 3.2	160
Exercise 3.4	Read input from the keyboard	161
Exercise 3.5	Read input from the command line	161
Exercise 3.6	Prompt the user for input to the formula (1.1) . . .	161
Exercise 3.7	Read command line input for the formula (1.1) . .	161
Exercise 3.8	Make the program from Exer. 3.7 safer	161
Exercise 3.9	Test more in the program from Exer. 3.7	161
Exercise 3.10	Raise an exception in Exer. 3.9	161
Exercise 3.11	Look up calendar functionality	162
Exercise 3.12	Use the StringFunction tool	162
Exercise 3.13	Extend a program from Ch. 3.2.1	162
Exercise 3.14	Why we test for specific exception types	162
Exercise 3.15	Make a simple module	163
Exercise 3.16	Make a useful main program for Exer. 3.15	163
Exercise 3.17	Make a module in Exer. 2.39	163
Exercise 3.18	Extend the module from Exer. 3.17	163
Exercise 3.19	Use options and values in Exer. 3.18	163
Exercise 3.20	Use optparse in the program from Ch. 3.2.4	163
Exercise 3.21	Compute the distance it takes to stop a car	163
Exercise 3.22	Check if mathematical rules hold on a computer .	164
Exercise 3.23	Improve input to the program in Exer. 3.22	164
Exercise 3.24	Apply the program from Exer. 3.23	165
Exercise 3.25	Compute the binomial distribution	165
Exercise 3.26	Apply the binomial distribution	166
Exercise 3.27	Compute probabilities with the Poisson distribution	166
Exercise 4.1	Fill lists with function values	225
Exercise 4.2	Fill arrays; loop version	226
Exercise 4.3	Fill arrays; vectorized version	226
Exercise 4.4	Apply a function to a vector	226
Exercise 4.5	Simulate by hand a vectorized expression	226
Exercise 4.6	Demonstrate array slicing	227
Exercise 4.7	Plot the formula (1.1)	227
Exercise 4.8	Plot the formula (1.1) for several v_0 values	227
Exercise 4.9	Plot exact and inexact Fahrenheit–Celsius formulas	227
Exercise 4.10	Plot the trajectory of a ball	227

Exercise 4.11	Plot a wave packet	227
Exercise 4.12	Use pyreport in Exer. 4.11	227
Exercise 4.13	Judge a plot	228
Exercise 4.14	Plot the viscosity of water	228
Exercise 4.15	Explore a function graphically	228
Exercise 4.16	Plot Taylor polynomial approximations to $\sin x$. .	228
Exercise 4.17	Animate a wave packet	229
Exercise 4.18	Animate the evolution of Taylor polynomials . . .	229
Exercise 4.19	Plot the velocity profile for pipeflow	230
Exercise 4.20	Plot the approximate function from Exer. 1.13 . .	230
Exercise 4.21	Plot functions from the command line	231
Exercise 4.22	Improve the program from Exercise 4.21	231
Exercise 4.23	Demonstrate energy concepts from physics	231
Exercise 4.24	Plot a w-like function	231
Exercise 4.25	Plot a smoothed “hat” function	232
Exercise 4.26	Experience overflow in a function	232
Exercise 4.27	Experience less overflow in a function	233
Exercise 4.28	Extend Exer. 4.4 to a rank 2 array	233
Exercise 4.29	Explain why array computations fail	233
Exercise 5.1	Determine the limit of a sequence	260
Exercise 5.2	Determine the limit of a sequence	260
Exercise 5.3	Experience convergence problems	260
Exercise 5.4	Convergence of sequences with π as limit	261
Exercise 5.5	Reduce memory usage of difference equations . . .	261
Exercise 5.6	Development of a loan over N months	261
Exercise 5.7	Solve a system of difference equations	261
Exercise 5.8	Extend the model (5.27)–(5.28)	261
Exercise 5.9	Experiment with the program from Exer. 5.8 . . .	262
Exercise 5.10	Change index in a difference equation	262
Exercise 5.11	Construct time points from dates	262
Exercise 5.12	Solve nonlinear equations by Newton’s method . .	263
Exercise 5.13	Visualize the convergence of Newton’s method . .	263
Exercise 5.14	Implement the Secant method	264
Exercise 5.15	Test different methods for root finding	264
Exercise 5.16	Difference equations for computing $\sin x$	264
Exercise 5.17	Difference equations for computing $\cos x$	265
Exercise 5.18	Make a guitar-like sound	265
Exercise 5.19	Damp the bass in a sound file	265
Exercise 5.20	Damp the treble in a sound file	266
Exercise 5.21	Demonstrate oscillatory solutions of (5.13)	266
Exercise 5.22	Make the program from Exer. 5.21 more flexible .	267
Exercise 5.23	Simulate the price of wheat	267
Exercise 6.1	Read a two-column data file	323
Exercise 6.2	Read a data file	323
Exercise 6.3	Simplify the implementation of Exer. 6.1	323

Exercise 6.4	Fit a polynomial to data	323
Exercise 6.5	Read acceleration data and find velocities	324
Exercise 6.6	Read acceleration data and plot velocities	325
Exercise 6.7	Find velocity from GPS coordinates	325
Exercise 6.8	Make a dictionary from a table	325
Exercise 6.9	Explore syntax differences: lists vs. dictionaries . .	326
Exercise 6.10	Improve the program from Ch. 6.2.4	326
Exercise 6.11	Interpret output from a program	326
Exercise 6.12	Make a dictionary	327
Exercise 6.13	Make a nested dictionary	327
Exercise 6.14	Make a nested dictionary from a file	327
Exercise 6.15	Compute the area of a triangle	327
Exercise 6.16	Compare data structures for polynomials	327
Exercise 6.17	Compute the derivative of a polynomial	327
Exercise 6.18	Generalize the program from Ch. 6.2.6	328
Exercise 6.19	Write function data to file	328
Exercise 6.20	Specify functions on the command line	328
Exercise 6.21	Interpret function specifications	329
Exercise 6.22	Compare average temperatures in two cities	330
Exercise 6.23	Compare average temperatures in many cities . . .	330
Exercise 6.24	Plot the temperature in a city, 1995-today	331
Exercise 6.25	Plot temperatures in several cities	332
Exercise 6.26	Try Word or OpenOffice to write a program	332
Exercise 6.27	Evaluate objects in a boolean context	332
Exercise 6.28	Generate an HTML report	333
Exercise 6.29	Fit a polynomial to experimental data	333
Exercise 6.30	Interpret an HTML file with rainfall data	334
Exercise 6.31	Generate an HTML report with figures	334
Exercise 7.1	Make a function class	397
Exercise 7.2	Make a very simple class	398
Exercise 7.3	Extend the class from Ch. 7.2.1	398
Exercise 7.4	Make classes for a rectangle and a triangle	398
Exercise 7.5	Make a class for straight lines	398
Exercise 7.6	Improve the constructor in Exer. 7.5	398
Exercise 7.7	Make a class for quadratic functions	399
Exercise 7.8	Make a class for linear springs	399
Exercise 7.9	Implement Lagrange's interpolation formula	399
Exercise 7.10	A very simple "Hello, World!" class	400
Exercise 7.11	Use special methods in Exer. 7.1	400
Exercise 7.12	Modify a class for numerical differentiation	400
Exercise 7.13	Make a class for nonlinear springs	401
Exercise 7.14	Extend the class from Ch. 7.2.1	401
Exercise 7.15	Implement a class for numerical differentiation . .	401
Exercise 7.16	Verify a program	402
Exercise 7.17	Test methods for numerical differentiation	402

Exercise 7.18	Make a class for summation of series	403
Exercise 7.19	Apply the differentiation class from Ch. 7.3.2	403
Exercise 7.20	Use classes for computing inverse functions	404
Exercise 7.21	Vectorize a class for numerical integration	404
Exercise 7.22	Speed up repeated integral calculations	404
Exercise 7.23	Solve a simple ODE in two ways	405
Exercise 7.24	Solve the ODE (B.36)	405
Exercise 7.25	Simulate a falling or rising body in a fluid	405
Exercise 7.26	Check the solution's limit in Exer. 7.25	407
Exercise 7.27	Implement the modified Euler method; function	407
Exercise 7.28	Implement the modified Euler method; class	408
Exercise 7.29	Increase the flexibility in Exer. 7.28	408
Exercise 7.30	Solve an ODE specified on the command line	408
Exercise 7.31	Apply a polynomial class	409
Exercise 7.32	Find a bug in a class for polynomials	409
Exercise 7.33	Subtraction of polynomials	409
Exercise 7.34	Represent a polynomial by an array	409
Exercise 7.35	Vectorize a class for polynomials	409
Exercise 7.36	Use a dict to hold polynomial coefficients; add . . .	410
Exercise 7.37	Use a dict to hold polynomial coefficients; mul . . .	410
Exercise 7.38	Extend class Vec2D to work with lists/tuples	410
Exercise 7.39	Use NumPy arrays in class Vec2D	411
Exercise 7.40	Use classes in the program from Ch. 6.6.2	411
Exercise 7.41	Use a class in Exer. 6.28	412
Exercise 7.42	Apply the class from Exer. 7.41 interactively	413
Exercise 7.43	Find the optimal production for a company	413
Exercise 7.44	Extend the program from Exer. 7.43	415
Exercise 7.45	Model the economy of fishing	415
Exercise 8.1	Flip a coin N times	463
Exercise 8.2	Compute a probability	463
Exercise 8.3	Choose random colors	463
Exercise 8.4	Draw balls from a hat	464
Exercise 8.5	Probabilities of rolling dice	464
Exercise 8.6	Estimate the probability in a dice game	464
Exercise 8.7	Decide if a dice game is fair	464
Exercise 8.8	Adjust the game in Exer. 8.7	464
Exercise 8.9	Probabilities of throwing two dice	465
Exercise 8.10	Compute the probability of drawing balls	465
Exercise 8.11	Compute the probability of hands of cards	465
Exercise 8.12	Play with vectorized boolean expressions	466
Exercise 8.13	Vectorize the program from Exer. 8.1	466
Exercise 8.14	Vectorize the code in Exer. 8.2	466
Exercise 8.15	Throw dice and compute a small probability	466
Exercise 8.16	Difference equation for random numbers	466
Exercise 8.17	Make a class for drawing balls from a hat	467

Exercise 8.18	Independent vs. dependent random numbers	467
Exercise 8.19	Compute the probability of flipping a coin	467
Exercise 8.20	Extend Exer. 8.19	468
Exercise 8.21	Simulate the problems in Exer. 3.26	468
Exercise 8.22	Simulate a poker game	468
Exercise 8.23	Write a non-vectorized version of a code	469
Exercise 8.24	Estimate growth in a simulation model	469
Exercise 8.25	Investigate guessing strategies for Ch. 8.4.1	469
Exercise 8.26	Make a vectorized solution to Exer. 8.7	469
Exercise 8.27	Compare two playing strategies	470
Exercise 8.28	Solve Exercise 8.27 with different no. of dice	470
Exercise 8.29	Extend Exercise 8.28	470
Exercise 8.30	Compute π by a Monte Carlo method	470
Exercise 8.31	Do a variant of Exer. 8.30	470
Exercise 8.32	Compute π by a random sum	470
Exercise 8.33	1D random walk with drift	470
Exercise 8.34	1D random walk until a point is hit	471
Exercise 8.35	Make a class for 2D random walk	471
Exercise 8.36	Vectorize the class code from Exer. 8.35	471
Exercise 8.37	2D random walk with walls; scalar version	472
Exercise 8.38	2D random walk with walls; vectorized version ..	472
Exercise 8.39	Simulate the mixture of gas molecules	472
Exercise 8.40	Simulate the mixture of gas molecules	473
Exercise 8.41	Guess beer brands	473
Exercise 8.42	Simulate stock prices	473
Exercise 8.43	Compute with option prices in finance	474
Exercise 8.44	Compute velocity and acceleration	475
Exercise 8.45	Numerical differentiation of noisy signals	475
Exercise 8.46	Model the noise in the data in Exer. 8.44	476
Exercise 8.47	Reduce the noise in Exer. 8.44	477
Exercise 8.48	Find the expected waiting time in traffic lights ..	477
Exercise 9.1	Demonstrate the magic of inheritance	546
Exercise 9.2	Inherit from classes in Ch. 9.1	546
Exercise 9.3	Inherit more from classes in Ch. 9.1	547
Exercise 9.4	Reverse the class hierarchy from Ch. 9.1	547
Exercise 9.5	Super- and subclass for a point	547
Exercise 9.6	Modify a function class by subclassing	547
Exercise 9.7	Explore the accuracy of difference formulas	548
Exercise 9.8	Implement a subclass	548
Exercise 9.9	Make classes for numerical differentiation	548
Exercise 9.10	Implement a new subclass for differentiation	548
Exercise 9.11	Understand if a class can be used recursively	549
Exercise 9.12	Represent people by a class hierarchy	549
Exercise 9.13	Add a new class in a class hierarchy	550
Exercise 9.14	Change the user interface of a class hierarchy	550

Exercise 9.15	Compute convergence rates of numerical integration methods	550
Exercise 9.16	Add common functionality in a class hierarchy . .	551
Exercise 9.17	Make a class hierarchy for root finding	552
Exercise 9.18	Use the ODESolver hierarchy to solve a simple ODE	552
Exercise 9.19	Use the 4th-order Runge-Kutta on (B.34)	552
Exercise 9.20	Solve an ODE until constant solution	552
Exercise 9.21	Use classes in Exer. 9.20	553
Exercise 9.22	Scale away parameters in Exer. 9.20	553
Exercise 9.23	Compare ODE methods	553
Exercise 9.24	Solve two coupled ODEs for radioactive decay . .	554
Exercise 9.25	Compare methods for solving the ODE (B.36) . .	554
Exercise 9.26	Code a 2nd-order Runge-Kutta method; function	554
Exercise 9.27	Code a 2nd-order Runge-Kutta method; class . .	555
Exercise 9.28	Implement a midpoint method for ODEs	555
Exercise 9.29	Implement a modified Euler method for ODEs . .	555
Exercise 9.30	Improve the implementation in Exer. 7.25	555
Exercise 9.31	Visualize the different forces in Exer. 9.30	556
Exercise 9.32	Find the body's position in Exer. 9.30	556
Exercise 9.33	Compare methods for solving (B.37)–(B.38)	556
Exercise 9.34	Add the effect of air resistance on a ball	557
Exercise 9.35	Make a class for drawing an arrow	557
Exercise 9.36	Make a class for drawing a person	557
Exercise 9.37	Animate a person with waving hands	558
Exercise 9.38	Make a class for drawing a car	558
Exercise 9.39	Make a car roll	558
Exercise 9.40	Make a class for differentiating noisy data	558
Exercise 9.41	Find local and global extrema of a function	559
Exercise 9.42	Improve the accuracy in Exer. 9.41	560
Exercise 9.43	Make a calculus calculator class	561
Exercise 9.44	Extend Exer. 9.43	561
Exercise 9.45	Formulate a 2nd-order ODE as a system	562
Exercise 9.46	Solve the system in Exer. 9.45 in a special case . .	563
Exercise 9.47	Enhance the code from Exer. 9.46	563
Exercise 9.48	Make a tool for analyzing oscillatory solutions . .	565
Exercise 9.49	Replace functions by class in Exer. 9.46	566
Exercise 9.50	Allow flexible choice of functions in Exer. 9.49 . .	569
Exercise 9.51	Use the modules from Exer. 9.49 and 9.50	570
Exercise 9.52	Use the modules from Exer. 9.49 and 9.50	571
Exercise A.1	Interpolate a discrete function	599
Exercise A.2	Study a function for different parameter values . .	599
Exercise A.3	Study a function and its derivative	600
Exercise A.4	Use the Trapezoidal method	600
Exercise A.5	Compute a sequence of integrals	601

Exercise A.6	Use the Trapezoidal method	601
Exercise A.7	Trigonometric integrals	602
Exercise A.8	Plot functions and their derivatives	602
Exercise A.9	Use the Trapezoidal method	603
Exercise B.1	Solve a nonhomogeneous linear ODE	621
Exercise B.2	Solve a nonlinear ODE	621
Exercise B.3	Solve an ODE for $y(x)$	621
Exercise B.4	Experience instability of an ODE	622
Exercise B.5	Solve an ODE for the arc length	622
Exercise B.6	Solve an ODE with time-varying growth	622
Exercise B.7	Solve an ODE for emptying a tank	623
Exercise B.8	Solve an ODE system for an electric circuit	623
Exercise C.1	Use a w function with a step	649
Exercise C.2	Make a callback function in Exercise C.1	649
Exercise C.3	Improve input to the simulation program	650