Editors
Timothy J. Barth
Michael Griebel
David E. Keyes
Risto M. Nieminen
Dirk Roose
Tamar Schlick
A Primer on Scientific Programming with Python

2nd Edition
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 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 to use Python version 2.7 (although version 2.6 will work for most of the material). For Chapters 5–9 and Appendices A–E you also need the NumPy, Matplotlib, SciTools packages. 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.

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 3.1. The general recommendation is to go for version 3, but programs are then not compatible with version 2 and vice versa. There is still a problem that much useful mathematical software in Python has not yet been ported to version 3. Therefore, scientific computing with Python still goes mostly with version 2. A widely used strategy for software developers who want to write Python code that works with both versions, is to develop for v2.7, which is very close to v3.1, and then use the translation tool 2to3 to automatically translate the code to version 3.1.

When using v2.7, one should employ the newest syntax and modules that make the differences between version 2 and 3 very small. This strategy is adopted in the present book. Only two differences between version 2 and 3 are expected to be significant for the programs in the book: a/b implies float division in version 3 if a and b are integers, and print ‘Hello’ in version 2 must be turned into a function call print(‘Hello’) in version 3. None of these differences should lead to any annoying problems when future readers study the book’s v2.7 examples, but program in version 3. Anyway, running 2to3 on the example files generates the corresponding version 3 code.

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 with lists, including nested lists. The next chapter deals with two other fundamental concepts in programming: functions and
if-else tests. Successful further reading of the book demands that Chapters 1–3 are digested.

How to read data into programs and deal with errors in input are the subjects of Chapter 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, or from a question-answer dialog with the user, and validity checks of the input are added. The formulas are then shown as graphs in Chapter 5. After having studied Chapters 1–5, the reader should have enough knowledge of programming to solve mathematical problems by “Matlab-style” programming.

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 walks. Object-oriented programming, in the meaning of class hierarchies and inheritance, is the subject of Chapter 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. We also treat sound as a sequence. Only programming concepts from Chapters 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 Chapters 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 systems of ordinary differential equations, primarily using classes and object-oriented programming, is the subject of Appendix E. The material in this appendix brings together many of the programming concepts from Chapters 1–9 in a mathematical setting and ends up with a flexible and general tool for solving differential equations.
Appendix F is devoted to the art of debugging, and in fact problem solving in general, while Appendix G 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 (3.7, 4.18–4.20, 5.29, 5.30), Taylor series (3.21, 5.20, 5.27, A.16, A.17, 7.23), falling objects (E.5, E.6, E.7, E.25, E.26), oscillatory population growth (A.21, A.22, 6.25, 7.34, 7.35), analysis of web data (6.22, 6.28–6.30), graphics and animation (9.20–9.23), optimization and finance (A.23, 8.42, 8.43), statistics and probability (4.24–4.26, 8.22–8.24), hazard games (8.8–8.13), random walk and statistical physics (8.33–8.40), noisy data analysis (8.44–8.48), numerical methods (5.13, 5.14, 7.9, A.12, 7.22, 9.16–9.18, E.15–E.23), building a calculus calculator (7.36, 7.37, 9.24, 9.25), and creating a toolkit for simulating vibrating engineering systems (E.30–E.37).

Chapters 1–9 and Appendix E have, from 2007, formed the core of an introductory first-semester course on scientific programming, INF1100, at the University of Oslo (see below).

Changes to the First Edition. Besides numerous corrections of misprints, the second edition features a major reorganization of several chapters. Chapter 2 in the first edition, Basic Constructions, was a comprehensive chapter, both with respect to length and topics. This chapter has therefore been split in two for the second edition: a new Chapter 2 Loops and Lists and a new Chapter 3 Functions and Branching. A new Chapter 2.1.4 explicitly explains how to implement a summation expression by a loop, and later examples present alternative implementations.

All text and program files that used the getopt module to parse command-line options in the first edition now make use of the simpler and more flexible argparse module (new in Python v2.7/3.1).

The material on curve plotting in Chapter 5 has been thoroughly revised. Now we give an introduction to plotting with Matplotlib as well as SciTools/Easyviz. Both tools are very similar from a syntax point of view. Much of the more detailed information on Easyviz plotting in the first edition has been removed, and the reader is directed to the online manuals for more details.

While the first edition almost exclusively used “star import” for convenience (e.g., from numpy import * and from scitools.std import *), the second edition tries to adhere to the standard import numpy as np. However, in mathematical formulas that are to work with scalar and array variables, we do not want an explicit prefix. Avoiding the namespace prefixes is important for making formulas as close to the mathematical notation as possible as well as for making the transition
from or to Matlab smooth. The two import styles have different merits and applications. The choice of style in various examples is carefully thought through in the second edition.

Chapter 5 in the first edition, *Sequences and Difference Equations*, has now become Appendix A since the material is primarily about mathematical modeling, and no new basic programming concepts are introduced.

Chapter 6 in the first edition, *Files, Strings, and Dictionaries*, has been substantially revised. Now, Chapter 6.4, on downloading and interpreting data from web pages, have completely new examples. Many of the exercises in this chapter are also reworked to fit with the new examples.

The material on differential equations in chapters on classes (Ch. 7 and 9 in the first edition) has been extracted, reworked, slightly expanded, and placed in Appendix E. This restructuring allows a more flexible treatment of differential equations, and parts of this important topic can be addressed right after Chapter 3, if desired. Also, the changes make readers of Chapters 7 and 9 less disturbed with more difficult mathematical subjects.

To distinguish between Python’s `random` module and the one in `numpy`, we have in Chapter 8 changed practice compared with the first edition. Now `random` always refers to Python’s `random` module, while the `random` module in `numpy` is normally invoked as `np.random` (or occasionally as `numpy.random`). The associated software has been revised similarly.

**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 B and C 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 help establishing 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, Arve Knudsen, Tobias Viddarsønn Langhoff, Solveig Masvie, Håkon Møller, Mathias Nedrebø, 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 pro-
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-Ha Le Thi, Ruth Allewelt, Peggy Glauch-Ruge, Nadja Kroke, Thomas Schmidt, and Patrick Waltemate, is highly appreciated, and ensured a smooth and rapid production of both the first and the second edition of this book.

*Oslo, February 2011*  
Hans Petter Langtangen
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 ................................ 14
  1.3 Another Formula: Celsius-Fahrenheit Conversion .... 19
    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 .......... 22
  1.4 Evaluating Standard Mathematical Functions .......... 22
    1.4.1 Example: Using the Square Root Function ...... 23
    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
### Contents

1.6.1 Complex Arithmetics in Python ....................... 32  
1.6.2 Complex Functions in Python ................. 33  
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 .......... 40

1.8 Exercises .............................................. 41

2 Loops and Lists ......................................... 51

2.1 While Loops ............................................. 51

2.1.1 A Naive Solution ................................... 51

2.1.2 While Loops ...................................... 52

2.1.3 Boolean Expressions ............................. 54

2.1.4 Loop Implementation of a Sum ............... 56

2.2 Lists ................................................ 57

2.2.1 Basic List Operations ......................... 58

2.2.2 For Loops ..................................... 60

2.3 Alternative Implementations with Lists and Loops .... 62

2.3.1 While Loop Implementation of a For Loop .... 62

2.3.2 The Range Construction ...................... 62

2.3.3 For Loops with List Indices .................. 63

2.3.4 Changing List Elements ...................... 65

2.3.5 List Comprehension ........................... 65

2.3.6 Traversing Multiple Lists Simultaneously .... 66

2.4 Nested Lists .......................................... 66

2.4.1 A Table as a List of Rows or Columns ........ 67

2.4.2 Printing Objects ................................... 68

2.4.3 Extracting Sublists ............................. 69

2.4.4 Traversing Nested Lists ...................... 71

2.5 Tuples ................................................ 73

2.6 Summary .............................................. 74

2.6.1 Chapter Topics ................................... 74

2.6.2 Summarizing Example: Analyzing List Data .... 77

2.6.3 How to Find More Python Information ........ 80

2.7 Exercises .............................................. 81

3 Functions and Branching .................................. 91

3.1 Functions .............................................. 91

3.1.1 Functions of One Variable ...................... 91

3.1.2 Local and Global Variables .................. 93

3.1.3 Multiple Arguments ............................. 95

3.1.4 Multiple Return Values ...................... 97

3.1.5 Functions with No Return Values ........... 99
3.1.6 Keyword Arguments ........................................ 100
3.1.7 Doc Strings .................................................. 103
3.1.8 Function Input and Output ................................. 104
3.1.9 Functions as Arguments to Functions ................. 104
3.1.10 The Main Program ........................................ 106
3.1.11 Lambda Functions ......................................... 107
3.2 Branching .......................................................... 108
  3.2.1 If-Else Blocks ............................................. 108
  3.2.2 Inline If Tests ............................................. 110
3.3 Summary .......................................................... 111
  3.3.1 Chapter Topics ............................................. 111
  3.3.2 Summarizing Example: Numerical Integration .... 113
3.4 Exercises .......................................................... 116

4 Input Data and Error Handling ................................. 129
  4.1 Asking Questions and Reading Answers .................. 130
    4.1.1 Reading Keyboard Input ................................ 130
    4.1.2 The Magic “eval” Function ............................. 131
    4.1.3 The Magic “exec” Function ............................. 135
    4.1.4 Turning String Expressions into Functions .......... 136
  4.2 Reading from the Command Line ......................... 137
    4.2.1 Providing Input on the Command Line ............... 137
    4.2.2 A Variable Number of Command-Line Arguments .... 138
    4.2.3 More on Command-Line Arguments .................. 139
    4.2.4 Option–Value Pairs on the Command Line .......... 140
  4.3 Handling Errors ............................................... 142
    4.3.1 Exception Handling .................................... 143
    4.3.2 Raising Exceptions .................................... 146
  4.4 A Glimpse of Graphical User Interfaces ................. 148
  4.5 Making Modules ................................................ 151
    4.5.1 Example: Compound Interest Formulas ............. 152
    4.5.2 Collecting Functions in a Module File .............. 153
    4.5.3 Using Modules .......................................... 157
  4.6 Summary ........................................................ 159
    4.6.1 Chapter Topics .......................................... 159
    4.6.2 Summarizing Example: Bisection Root Finding ... 162
  4.7 Exercises ....................................................... 170

5 Array Computing and Curve Plotting ........................ 177
  5.1 Vectors ........................................................ 178
    5.1.1 The Vector Concept ................................... 178
    5.1.2 Mathematical Operations on Vectors ................. 179
    5.1.3 Vector Arithmetics and Vector Functions .......... 181
  5.2 Arrays in Python Programs ................................. 183
5.2.1 Using Lists for Collecting Function Data ............. 183
5.2.2 Basics of Numerical Python Arrays ..................... 184
5.2.3 Computing Coordinates and Function Values ............. 185
5.2.4 Vectorization ........................................ 186

5.3 Curve Plotting ........................................ 188
5.3.1 Matplotlib; Pylab ..................................... 188
5.3.2 Matplotlib; Pyplot .................................... 192
5.3.3 SciTools and Easyviz ................................. 194
5.3.4 Making Animations ................................... 199
5.3.5 Curves in Pure Text .................................. 203

5.4 Plotting Difficulties ..................................... 204
5.4.1 Piecewisely Defined Functions .......................... 205
5.4.2 Rapidly Varying Functions ............................... 210
5.4.3 Vectorizing StringFunction Objects ..................... 211

5.5 More on Numerical Python Arrays .......................... 212
5.5.1 Copying Arrays ...................................... 212
5.5.2 In-Place Arithmetics ................................... 213
5.5.3 Allocating Arrays ..................................... 214
5.5.4 Generalized Indexing .................................. 214
5.5.5 Testing for the Array Type .............................. 215
5.5.6 Compact Syntax for Array Generation ................. 216
5.5.7 Shape Manipulation .................................. 216

5.6 Higher-Dimensional Arrays ................................. 217
5.6.1 Matrices and Arrays .................................. 217
5.6.2 Two-Dimensional Numerical Python Arrays ............ 218
5.6.3 Array Computing ..................................... 220
5.6.4 Two-Dimensional Arrays and Functions of Two Variables ........................................ 221
5.6.5 Matrix Objects ...................................... 222

5.7 Summary ................................................ 223
5.7.1 Chapter Topics ....................................... 223
5.7.2 Summarizing Example: Animating a Function .......... 224

5.8 Exercises ............................................... 229

6 Files, Strings, and Dictionaries ............................... 243
6.1 Reading Data from File .................................. 243
6.1.1 Reading a File Line by Line ......................... 244
6.1.2 Reading a Mixture of Text and Numbers .............. 247
6.1.3 What Is a File, Really? ............................... 248

6.2 Dictionaries ............................................. 252
6.2.1 Making Dictionaries ................................... 252
6.2.2 Dictionary Operations ................................. 253
6.2.3 Example: Polynomials as Dictionaries .................. 254
6.2.4 Example: File Data in Dictionaries .................... 256
6.2.5 Example: File Data in Nested Dictionaries ............ 257
8.7.1 Basic Implementation ........................................ 411
8.7.2 Vectorized Implementation ................................. 413
8.8 Summary .......................................................... 414
  8.8.1 Chapter Topics ................................................. 414
  8.8.2 Summarizing Example: Random Growth .................. 415
8.9 Exercises .......................................................... 421

9 Object-Oriented Programming .............................. 437
  9.1 Inheritance and Class Hierarchies ......................... 437
    9.1.1 A Class for Straight Lines ................................. 438
    9.1.2 A First Try on a Class for Parabolas ................. 439
    9.1.3 A Class for Parabolas Using Inheritance .......... 439
    9.1.4 Checking the Class Type ..................................... 441
    9.1.5 Attribute versus Inheritance ............................... 442
    9.1.6 Extending versus Restricting Functionality .... 443
    9.1.7 Superclass for Defining an Interface .................... 444
  9.2 Class Hierarchy for Numerical Differentiation .... 446
    9.2.1 Classes for Differentiation ............................... 447
    9.2.2 A Flexible Main Program .................................... 450
    9.2.3 Extensions .................................................. 451
    9.2.4 Alternative Implementation via Functions .......... 454
    9.2.5 Alternative Implementation via Functional
        Programming ...................................................... 455
    9.2.6 Alternative Implementation via a Single Class ...... 456
  9.3 Class Hierarchy for Numerical Integration .......... 458
    9.3.1 Numerical Integration Methods ............................ 458
    9.3.2 Classes for Integration .................................... 459
    9.3.3 Using the Class Hierarchy ................................ 463
    9.3.4 About Object-Oriented Programming .................. 466
  9.4 Class Hierarchy for Geometric Shapes ............... 467
    9.4.1 Using the Class Hierarchy ................................ 467
    9.4.2 Overall Design of the Class Hierarchy ............. 469
    9.4.3 The Drawing Tool ............................................ 470
    9.4.4 Implementation of Shape Classes ................. 472
    9.4.5 Scaling, Translating, and Rotating a Figure .... 476
  9.5 Summary .......................................................... 480
    9.5.1 Chapter Topics ................................................. 480
    9.5.2 Summarizing Example: Input Data Reader ........... 482
  9.6 Exercises .......................................................... 488

A Sequences and Difference Equations .................. 497
  A.1 Mathematical Models Based on Difference Equations . 498
    A.1.1 Interest Rates ............................................... 499
    A.1.2 The Factorial as a Difference Equation ............. 501
    A.1.3 Fibonacci Numbers ......................................... 502
## D A Complete Differential Equation Project

D.1 About the Problem: Motion and Forces in Physics
   - D.1.1 The Physical Problem
   - D.1.2 The Computational Algorithm
   - D.1.3 Derivation of the Mathematical Model
   - D.1.4 Derivation of the Algorithm

D.2 Program Development and Testing
   - D.2.1 Implementation
   - D.2.2 Callback Functionality
   - D.2.3 Making a Module
   - D.2.4 Verification

D.3 Visualization
   - D.3.1 Simultaneous Computation and Plotting
   - D.3.2 Some Applications
   - D.3.3 Remark on Choosing $\Delta t$
   - D.3.4 Comparing Several Quantities in Subplots
   - D.3.5 Comparing Approximate and Exact Solutions
   - D.3.6 Evolution of the Error as $\Delta t$ Decreases

D.4 Exercises

## E Programming of Differential Equations

E.1 Scalar Ordinary Differential Equations
   - E.1.1 Examples on Right-Hand-Side Functions
   - E.1.2 The Forward Euler Scheme
   - E.1.3 Function Implementation
   - E.1.4 Verifying the Implementation
   - E.1.5 Switching Numerical Method
   - E.1.6 Class Implementation
   - E.1.7 Example: Logistic Growth

E.2 Systems of Ordinary Differential Equations
   - E.2.1 Mathematical Problem
   - E.2.2 Example of a System of ODEs
   - E.2.3 From Scalar ODE Code to Systems
   - E.2.4 Numerical Methods
   - E.2.5 The ODE Solver Class Hierarchy
   - E.2.6 The Backward Euler Method
   - E.2.7 Application 1: $u' = u$
   - E.2.8 Application 2: The Logistic Equation
   - E.2.9 Application 3: An Oscillating System
   - E.2.10 Application 4: The Trajectory of a Ball

E.3 Exercises

## F Debugging

F.1 Using a Debugger
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>F.2</td>
<td>How to Debug</td>
<td>658</td>
</tr>
<tr>
<td>F.2.1</td>
<td>A Recipe for Program Writing and Debugging</td>
<td>658</td>
</tr>
<tr>
<td>F.2.2</td>
<td>Application of the Recipe</td>
<td>660</td>
</tr>
<tr>
<td>G</td>
<td>Technical Topics</td>
<td>673</td>
</tr>
<tr>
<td>G.1</td>
<td>Different Ways of Running Python Programs</td>
<td>673</td>
</tr>
<tr>
<td>G.1.1</td>
<td>Executing Python Programs in IPython</td>
<td>673</td>
</tr>
<tr>
<td>G.1.2</td>
<td>Executing Python Programs on Unix</td>
<td>673</td>
</tr>
<tr>
<td>G.1.3</td>
<td>Executing Python Programs on Windows</td>
<td>675</td>
</tr>
<tr>
<td>G.1.4</td>
<td>Executing Python Programs on Macintosh</td>
<td>677</td>
</tr>
<tr>
<td>G.1.5</td>
<td>Making a Complete Stand-Alone Executable</td>
<td>677</td>
</tr>
<tr>
<td>G.2</td>
<td>Integer and Float Division</td>
<td>677</td>
</tr>
<tr>
<td>G.3</td>
<td>Visualizing a Program with Lumpy</td>
<td>678</td>
</tr>
<tr>
<td>G.4</td>
<td>Doing Operating System Tasks in Python</td>
<td>681</td>
</tr>
<tr>
<td>G.5</td>
<td>Variable Number of Function Arguments</td>
<td>683</td>
</tr>
<tr>
<td>G.5.1</td>
<td>Variable Number of Positional Arguments</td>
<td>683</td>
</tr>
<tr>
<td>G.5.2</td>
<td>Variable Number of Keyword Arguments</td>
<td>686</td>
</tr>
<tr>
<td>G.6</td>
<td>Evaluating Program Efficiency</td>
<td>688</td>
</tr>
<tr>
<td>G.6.1</td>
<td>Making Time Measurements</td>
<td>688</td>
</tr>
<tr>
<td>G.6.2</td>
<td>Profiling Python Programs</td>
<td>690</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>693</td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td>695</td>
</tr>
</tbody>
</table>
List of Exercises

Exercise 1.1  Compute 1+1 ................................................. 43
Exercise 1.2  Write a “Hello, World!” program ..................... 43
Exercise 1.3  Derive and compute a formula ......................... 43
Exercise 1.4  Convert from meters to British length units .... 43
Exercise 1.5  Compute the mass of various substances .......... 43
Exercise 1.6  Compute the growth of money in a bank ........ 43
Exercise 1.7  Find error(s) in a program ............................ 44
Exercise 1.8  Type in program text ................................... 44
Exercise 1.9  Type in programs and debug them ................. 44
Exercise 1.10 Evaluate a Gaussian function ......................... 45
Exercise 1.11 Compute the air resistance on a football .......... 45
Exercise 1.12 Define objects in IPython ............................. 46
Exercise 1.13 How to cook the perfect egg .......................... 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 Explain why a program does not work ........... 48
Exercise 1.17 Find errors in Python statements ................... 48
Exercise 1.18 Find errors in the coding of a formula ........... 49
Exercise 2.1  Make a Fahrenheit–Celsius conversion table ........ 81
Exercise 2.2  Write an approximate Fahrenheit–Celsius conversion table ............................................. 81
Exercise 2.3  Generate odd numbers .................................. 81
Exercise 2.4  Store odd numbers in a list ............................ 81
Exercise 2.5  Generate odd numbers by a list comprehension .... 82
Exercise 2.6  Make a table of function values ....................... 82
Exercise 2.7  Store numbers in lists .................................. 82
Exercise 2.8  Work with a list .......................................... 82
Exercise 2.9  Simulate operations on lists by hand ............... 82
Exercise 2.10 Generate equally spaced coordinates ............. 82
Exercise 2.11 Use a list comprehension to solve Exer. 2.10 ...... 82
| Exercise 2.12 | Compute a mathematical sum | 83 |
| Exercise 2.13 | Use a for loop in Exer. 2.12 | 83 |
| Exercise 2.14 | Condense the program in Exer. 2.13 | 83 |
| Exercise 2.15 | Compute a polynomial via a product | 83 |
| Exercise 2.16 | Simulate a program by hand | 84 |
| Exercise 2.17 | Explore the Python Library Reference | 84 |
| Exercise 2.18 | Implement the sum function | 84 |
| Exercise 2.19 | Index a nested lists | 85 |
| Exercise 2.20 | Construct a double for loop over a nested list | 85 |
| Exercise 2.21 | Store data in lists in Exercise 2.2 | 85 |
| Exercise 2.22 | Store data from Exer. 2.7 in a nested list | 85 |
| Exercise 2.23 | Convert nested list comprehensions to nested standard loops | 85 |
| Exercise 2.24 | Demonstrate list functionality | 86 |
| Exercise 2.25 | Values of boolean expressions | 86 |
| Exercise 2.26 | Explore round-off errors from a large number of inverse operations | 86 |
| Exercise 2.27 | Explore what zero can be on a computer | 86 |
| Exercise 2.28 | Compare two real numbers on a computer | 87 |
| Exercise 2.29 | Interpret a code | 87 |
| Exercise 2.30 | Explore problems with inaccurate indentation | 88 |
| Exercise 2.31 | Simulate nested loops by hand | 88 |
| Exercise 2.32 | Explore punctuation in Python programs | 89 |
| Exercise 2.33 | Investigate a for loop over a changing list | 89 |
| Exercise 3.1 | Write a Fahrenheit–Celsius conversion function | 116 |
| Exercise 3.2 | Write the program in Exer. 2.12 as a function | 116 |
| Exercise 3.3 | Compute the area of an arbitrary triangle | 117 |
| Exercise 3.4 | Compute the length of a path | 117 |
| Exercise 3.5 | Approximate π | 117 |
| Exercise 3.6 | Write some simple functions | 117 |
| Exercise 3.7 | Approximate a function by a sum of sines | 118 |
| Exercise 3.8 | Implement a Gaussian function | 118 |
| Exercise 3.9 | Make a function of the formula in Exer. 1.13 | 118 |
| Exercise 3.10 | Write a function for numerical differentiation | 119 |
| Exercise 3.11 | Write a function for numerical integration | 119 |
| Exercise 3.12 | Improve the integration in Exer. 3.11 | 119 |
| Exercise 3.13 | Generalize the integration formula in Exer. 3.12 | 119 |
| Exercise 3.14 | Implement the factorial function | 120 |
| Exercise 3.15 | Compute velocity and acceleration from position data; one dimension | 120 |
| Exercise 3.16 | Compute velocity and acceleration from position data; two dimensions | 120 |
| Exercise 3.17 | Find the max and min values of a function | 121 |
| Exercise 3.18 | Find the max/min elements in a list | 121 |
| Exercise 3.19 | Express a step function as a Python function | 122 |
Exercise 3.20 Rewrite a mathematical function .................. 122
Exercise 3.21 Make a table for approximations of $\cos x$ ........ 122
Exercise 3.22 Write a sort function for a list of 4-tuples ........... 123
Exercise 3.23 Find prime numbers .................................. 124
Exercise 3.24 Explain why a program works ....................... 124
Exercise 3.25 Resolve a problem with a function ................. 124
Exercise 3.26 Use None in keyword arguments .................... 124
Exercise 3.27 Determine the types of some objects ................. 125
Exercise 3.28 Explain if vs. elif ................................... 125
Exercise 3.29 Find an error in a program ......................... 126
Exercise 3.30 Find programming errors ............................. 126
Exercise 4.1 Make an interactive program .......................... 170
Exercise 4.2 Read from the command line in Exer. 4.1 ............ 170
Exercise 4.3 Use exceptions in Exer. 4.2 .......................... 170
Exercise 4.4 Read input from the keyboard ....................... 170
Exercise 4.5 Read input from the command line ................... 170
Exercise 4.6 Prompt the user for input to a formula ............... 170
Exercise 4.7 Read command line input a formula .................. 171
Exercise 4.8 Make the program from Exer. 4.7 safer ............... 171
Exercise 4.9 Test more in the program from Exer. 4.7 .............. 171
Exercise 4.10 Raise an exception in Exer. 4.9 ..................... 171
Exercise 4.11 Compute the distance it takes to stop a car .......... 171
Exercise 4.12 Look up calendar functionality ...................... 171
Exercise 4.13 Use the StringFunction tool ....................... 171
Exercise 4.14 Extend a program from Ch. 4.2.1 .................... 172
Exercise 4.15 Why we test for specific exception types .......... 172
Exercise 4.16 Make a simple module .............................. 172
Exercise 4.17 Make a useful main program for Exer. 4.16 ......... 172
Exercise 4.18 Make a module in Exer. 3.7 ......................... 172
Exercise 4.19 Extend the module from Exer. 4.18 ................. 173
Exercise 4.20 Use options and values in Exer. 4.19 ............... 173
Exercise 4.21 Check if mathematical identities hold on a computer ........................................... 173
Exercise 4.22 Improve input to the program in Exer. 4.21 ........ 173
Exercise 4.23 Apply the program from Exer. 4.22 ................. 174
Exercise 4.24 Compute the binomial distribution .................. 174
Exercise 4.25 Apply the binomial distribution ..................... 175
Exercise 4.26 Compute probabilities with the Poisson distribution ................................................................. 175
Exercise 5.1 Fill lists with function values .......................... 229
Exercise 5.2 Fill arrays; loop version ................................ 230
Exercise 5.3 Fill arrays; vectorized version ....................... 230
Exercise 5.4 Apply a function to a vector ......................... 230
Exercise 5.5 Simulate by hand a vectorized expression .......... 230
Exercise 5.6 Demonstrate array slicing ........................... 230
Exercise 5.7  Use array computing in the example from Chap. 2.6.2 ........................................ 231
Exercise 5.8  Plot a formula ........................................ 231
Exercise 5.9  Plot a formula for several parameters ................. 231
Exercise 5.10 Specify the $x$ and $y$ axes in Exer. 5.9 .............. 231
Exercise 5.11 Plot exact and inexact Fahrenheit–Celsius formulas ........................................ 231
Exercise 5.12 Plot the trajectory of a ball ........................................ 231
Exercise 5.13 Implement Lagrange’s interpolation formula ............ 231
Exercise 5.14 Plot the polynomial in Exer. 5.13 .......................... 232
Exercise 5.15 Plot a wave packet ........................................ 233
Exercise 5.16 Use pyreport in Exer. 5.15 .......................... 233
Exercise 5.17 Judge a plot ........................................ 233
Exercise 5.18 Plot the viscosity of water ........................................ 233
Exercise 5.19 Explore a function graphically .......................... 234
Exercise 5.20 Plot Taylor polynomial approximations to $\sin x$ ...... 234
Exercise 5.21 Animate a wave packet ........................................ 234
Exercise 5.22 Animate a smoothed Heaviside function ............... 235
Exercise 5.23 Animate two-scale temperature variations .............. 235
Exercise 5.24 Improve the solution in Exer. 5.23 .......................... 236
Exercise 5.25 Animate a sequence of approximations to $\pi$ .......... 236
Exercise 5.26 Animate a planet’s orbit ........................................ 236
Exercise 5.27 Animate the evolution of Taylor polynomials ........... 237
Exercise 5.28 Plot the velocity profile for pipeflow .................... 238
Exercise 5.29 Plot the functions from Exer. 3.7 .......................... 238
Exercise 5.30 Make a movie of the functions from Exer. 3.7 .......................... 238
Exercise 5.31 Plot functions from the command line .................... 239
Exercise 5.32 Improve the program from Exercise 5.31 .............. 239
Exercise 5.33 Demonstrate energy concepts from physics .......... 239
Exercise 5.34 Plot a $w$-like function ........................................ 239
Exercise 5.35 Plot a smoothed “hat” function .......................... 239
Exercise 5.36 Experience overflow in a function .......................... 240
Exercise 5.37 Experience less overflow in a function .................... 241
Exercise 5.38 Extend Exer. 5.4 to a rank 2 array .......................... 241
Exercise 5.39 Explain why array computations fail .................... 241
Exercise 6.1  Read a two-column data file ........................................ 294
Exercise 6.2  Read a data file ........................................ 295
Exercise 6.3  Simplify the implementation of Exer. 6.1 ....................... 295
Exercise 6.4  Fit a polynomial to data ........................................ 295
Exercise 6.5  Read acceleration data and find velocities .................... 296
Exercise 6.6  Read acceleration data and plot velocities .............. 296
Exercise 6.7  Find velocity from GPS coordinates .......................... 297
Exercise 6.8  Make a dictionary from a table .......................... 297
Exercise 6.9  Explore syntax differences: lists vs. dictionaries .......... 297
Exercise 6.10 Improve the program from Ch. 6.2.4 .......................... 298
Exercise 6.11 Interpret output from a program .................. 298
Exercise 6.12 Make a dictionary ........................................ 298
Exercise 6.13 Make a nested dictionary ......................... 298
Exercise 6.14 Make a nested dictionary from a file ............ 299
Exercise 6.15 Compute the area of a triangle .................... 299
Exercise 6.16 Compare data structures for polynomials ....... 299
Exercise 6.17 Compute the derivative of a polynomial ....... 299
Exercise 6.18 Generalize the program from Ch. 6.2.6 ......... 300
Exercise 6.19 Write function data to file ......................... 300
Exercise 6.20 Specify functions on the command line ......... 300
Exercise 6.21 Interpret function specifications .................. 301
Exercise 6.22 Compare average temperatures in cities .......... 301
Exercise 6.23 Try Word or OpenOffice to write a program ... 302
Exercise 6.24 Evaluate objects in a boolean context ........... 302
Exercise 6.25 Generate an HTML report .......................... 303
Exercise 6.26 Fit a polynomial to experimental data .......... 303
Exercise 6.27 Generate an HTML report with figures ......... 304
Exercise 6.28 Extract information from a weather page ...... 304
Exercise 6.29 Compare alternative weather forecasts ......... 304
Exercise 6.30 Improve the output in Exercise 6.29 ............ 304
Exercise 7.1 Make a function class .................................. 359
Exercise 7.2 Make a very simple class ............................. 359
Exercise 7.3 Extend the class from Ch. 7.2.1 ................. 359
Exercise 7.4 Make classes for a rectangle and a triangle ....... 360
Exercise 7.5 Make a class for straight lines .................... 360
Exercise 7.6 Improve the constructor in Exer. 7.5 ............ 360
Exercise 7.7 Make a class for quadratic functions ............. 360
Exercise 7.8 Make a class for linear springs ................. 361
Exercise 7.9 Implement code from Exer. 5.13 as a class ..... 361
Exercise 7.10 A very simple “Hello, World!” class .......... 361
Exercise 7.11 Use special methods in Exer. 7.1 ............... 361
Exercise 7.12 Make a class for nonlinear springs .......... 361
Exercise 7.13 Extend the class from Ch. 7.2.1 ................. 362
Exercise 7.14 Implement a class for numerical differentiation ... 362
Exercise 7.15 Verify a program ........................................ 363
Exercise 7.16 Test methods for numerical differentiation ...... 363
Exercise 7.17 Modify a class for numerical differentiation ... 363
Exercise 7.18 Make a class for summation of series .......... 364
Exercise 7.19 Apply the differentiation class from Ch. 7.3.2 ... 364
Exercise 7.20 Use classes for computing inverse functions .... 364
Exercise 7.21 Vectorize a class for numerical integration ... 365
Exercise 7.22 Speed up repeated integral calculations ....... 365
Exercise 7.23 Apply a polynomial class ......................... 366
Exercise 7.24 Find a bug in a class for polynomials .......... 366
Exercise 7.25 Subtraction of polynomials ....................... 366
Exercise 7.26  Represent a polynomial by an array ............... 367
Exercise 7.27  Vectorize a class for polynomials ................. 367
Exercise 7.28  Use a dict to hold polynomial coefficients; add ... 367
Exercise 7.29  Use a dict to hold polynomial coefficients; mul ... 367
Exercise 7.30  Extend class Vec2D to work with lists/tuples ...... 368
Exercise 7.31  Extend class Vec2D to 3D vectors ................. 368
Exercise 7.32  Use NumPy arrays in class Vec2D ................. 368
Exercise 7.33  Use classes in the program from Ch. 6.6.2 ....... 369
Exercise 7.34  Use a class in Exer. 6.25 .................... 369
Exercise 7.35  Apply the class from Exer. 7.34 interactively .... 370
Exercise 7.36  Find local and global extrema of a function ...... 370
Exercise 7.37  Improve the accuracy in Exer. 7.36 ............... 372
Exercise 7.38  Find the optimal production for a company ..... 372
Exercise 7.39  Extend the program from Exer. 7.38 .......... 373
Exercise 7.40  Model the economy of fishing .................. 374
Exercise 8.1   Flip a coin N times ............................... 421
Exercise 8.2   Compute a probability ........................... 421
Exercise 8.3   Choose random colors ............................. 421
Exercise 8.4   Draw balls from a hat ............................. 422
Exercise 8.5   Probabilities of rolling dice ....................... 422
Exercise 8.6   Estimate the probability in a dice game ......... 422
Exercise 8.7   Compute the probability of hands of cards ...... 422
Exercise 8.8   Decide if a dice game is fair .................... 422
Exercise 8.9   Adjust the game in Exer. 8.8 ................. 422
Exercise 8.10  Compare two playing strategies .................. 423
Exercise 8.11  Solve Exercise 8.10 with different no. of dice ... 423
Exercise 8.12  Extend Exercise 8.11 ....................... 423
Exercise 8.13  Investigate the winning chances of some games . 423
Exercise 8.14  Probabilities of throwing two dice .............. 424
Exercise 8.15  Play with vectorized boolean expressions ...... 424
Exercise 8.16  Vectorize the program from Exer. 8.1 .......... 424
Exercise 8.17  Vectorize the code in Exer. 8.2 ................ 424
Exercise 8.18  Throw dice and compute a small probability .... 425
Exercise 8.19  Difference equation for random numbers ...... 425
Exercise 8.20  Make a class for drawing balls from a hat ..... 425
Exercise 8.21  Independent vs. dependent random numbers ... 426
Exercise 8.22  Compute the probability of flipping a coin ...... 426
Exercise 8.23  Extend Exer. 8.22 .......................... 426
Exercise 8.24  Simulate the problems in Exer. 4.25 ............ 427
Exercise 8.25  Simulate a poker game .......................... 427
Exercise 8.26  Write a non-vectorized version of a code ...... 427
Exercise 8.27  Estimate growth in a simulation model .......... 427
Exercise 8.28  Investigate guessing strategies for Ch. 8.4.1 .... 428
Exercise 8.29  Make a vectorized solution to Exer. 8.8 ........ 428
Exercise 8.30  Compute \( \pi \) by a Monte Carlo method ........ 428
Exercise 8.31  Do a variant of Exer. 8.30  
Exercise 8.32  Compute $\pi$ by a random sum  
Exercise 8.33  1D random walk with drift  
Exercise 8.34  1D random walk until a point is hit  
Exercise 8.35  Make a class for 2D random walk  
Exercise 8.36  Vectorize the class code from Exer. 8.35  
Exercise 8.37  2D random walk with walls; scalar version  
Exercise 8.38  2D random walk with walls; vectorized version  
Exercise 8.39  Simulate the mixture of gas molecules  
Exercise 8.40  Simulate the mixture of gas molecules  
Exercise 8.41  Guess beer brands  
Exercise 8.42  Simulate stock prices  
Exercise 8.43  Compute with option prices in finance  
Exercise 8.44  Compute velocity and acceleration  
Exercise 8.45  Numerical differentiation of noisy signals  
Exercise 8.46  Model the noise in the data in Exer. 8.44  
Exercise 8.47  Reduce the noise in Exer. 8.44  
Exercise 8.48  Make a class for differentiating noisy data  
Exercise 8.49  Find the expected waiting time in traffic lights  
Exercise 9.1  Demonstrate the magic of inheritance  
Exercise 9.2  Inherit from classes in Ch. 9.1  
Exercise 9.3  Inherit more from classes in Ch. 9.1  
Exercise 9.4  Reverse the class hierarchy from Ch. 9.1  
Exercise 9.5  Make circle a subclass of an ellipse  
Exercise 9.6  Make super- and subclass for a point  
Exercise 9.7  Modify a function class by subclassing  
Exercise 9.8  Explore the accuracy of difference formulas  
Exercise 9.9  Implement a subclass  
Exercise 9.10 Make classes for numerical differentiation  
Exercise 9.11 Implement a new subclass for differentiation  
Exercise 9.12 Understand if a class can be used recursively  
Exercise 9.13 Represent people by a class hierarchy  
Exercise 9.14 Add a new class in a class hierarchy  
Exercise 9.15 Change the user interface of a class hierarchy  
Exercise 9.16 Compute convergence rates of numerical integration methods  
Exercise 9.17 Add common functionality in a class hierarchy  
Exercise 9.18 Make a class hierarchy for root finding  
Exercise 9.19 Make a class for drawing an arrow  
Exercise 9.20 Make a class for drawing a person  
Exercise 9.21 Animate a person with waving hands  
Exercise 9.22 Make a class for drawing a car  
Exercise 9.23 Make a car roll  
Exercise 9.24 Make a calculus calculator class  
Exercise 9.25 Extend Exer. 9.24
| Exercise A.1 | Determine the limit of a sequence | 521 |
| Exercise A.2 | Determine the limit of a sequence | 521 |
| Exercise A.3 | Experience convergence problems | 521 |
| Exercise A.4 | Convergence of sequences with $\pi$ as limit | 521 |
| Exercise A.5 | Reduce memory usage of difference equations | 522 |
| Exercise A.6 | Development of a loan over $N$ months | 522 |
| Exercise A.7 | Solve a system of difference equations | 522 |
| Exercise A.8 | Extend the model (A.26)–(A.27) | 522 |
| Exercise A.9 | Experiment with the program from Exer. A.8 | 522 |
| Exercise A.10 | Change index in a difference equation | 523 |
| Exercise A.11 | Construct time points from dates | 523 |
| Exercise A.12 | Solve nonlinear equations by Newton’s method | 523 |
| Exercise A.13 | Visualize the convergence of Newton’s method | 524 |
| Exercise A.14 | Implement the Secant method | 524 |
| Exercise A.15 | Test different methods for root finding | 525 |
| Exercise A.16 | Difference equations for computing $\sin x$ | 525 |
| Exercise A.17 | Difference equations for computing $\cos x$ | 525 |
| Exercise A.18 | Make a guitar-like sound | 526 |
| Exercise A.19 | Damp the bass in a sound file | 526 |
| Exercise A.20 | Damp the treble in a sound file | 527 |
| Exercise A.21 | Demonstrate oscillatory solutions of (A.13) | 527 |
| Exercise A.22 | Improve the program from Exer. A.21 | 527 |
| Exercise A.23 | Simulate the price of wheat | 528 |
| Exercise B.1 | Interpolate a discrete function | 555 |
| Exercise B.2 | Study a function for different parameter values | 555 |
| Exercise B.3 | Study a function and its derivative | 556 |
| Exercise B.4 | Use the Trapezoidal method | 556 |
| Exercise B.5 | Compute a sequence of integrals | 557 |
| Exercise B.6 | Use the Trapezoidal method | 557 |
| Exercise B.7 | Trigonometric integrals | 558 |
| Exercise B.8 | Plot functions and their derivatives | 559 |
| Exercise B.9 | Use the Trapezoidal method | 559 |
| Exercise C.1 | Solve a nonhomogeneous linear ODE | 575 |
| Exercise C.2 | Solve a nonlinear ODE | 575 |
| Exercise C.3 | Solve an ODE for $y(x)$ | 575 |
| Exercise C.4 | Experience instability of an ODE | 576 |
| Exercise C.5 | Solve an ODE for the arc length | 576 |
| Exercise C.6 | Solve an ODE with time-varying growth | 576 |
| Exercise D.1 | Use a $w$ function with a step | 601 |
| Exercise D.2 | Make a callback function in Exercise D.1 | 601 |
| Exercise D.3 | Improve input to the simulation program | 601 |
| Exercise E.1 | Solve a simple ODE in two ways | 633 |
| Exercise E.2 | Use the ODESolver hierarchy to solve a simple ODE | 634 |
| Exercise E.3 | Solve an ODE for emptying a tank | 634 |
Exercise E.4  Logistic growth with time-varying carrying capacity ........................................ 634
Exercise E.5  Simulate a falling or rising body in a fluid ........... 635
Exercise E.6  Check the solution’s limit in Exer. E.5 ............... 636
Exercise E.7  Visualize the different forces in Exer. E.5 ........... 636
Exercise E.8  Solve an ODE until constant solution ................. 636
Exercise E.9  Use classes in Exer. E.8 ............................. 637
Exercise E.10  Scale away parameters in Exer. E.8 ................. 637
Exercise E.11  Use the 4th-order Runge-Kutta on (C.32) .......... 638
Exercise E.12  Compare ODE methods .............................. 638
Exercise E.13  Compare ODE methods .............................. 638
Exercise E.14  Solve two coupled ODEs for radioactive decay .... 638
Exercise E.15  Code a 2nd-order Runge-Kutta method; function 639
Exercise E.16  Code a 2nd-order Runge-Kutta method; class ... 639
Exercise E.17  Make an ODESolver subclass for Heun’s method 639
Exercise E.18  Make an ODESolver subclass for the Midpoint method ........................................ 639
Exercise E.19  Make an ODESolver subclass for an Adams-Bashforth method .............................. 640
Exercise E.20  Implement the iterated Midpoint method; function ........................................ 640
Exercise E.21  Implement the iterated Midpoint method; class .. 640
Exercise E.22  Make an ODESolver subclass for the iterated Midpoint method .......................... 640
Exercise E.23  Study convergence of numerical methods for ODEs ........................................ 641
Exercise E.24  Solve an ODE specified on the command line ... 641
Exercise E.25  Find the body’s position in Exer. E.5 ............... 641
Exercise E.26  Add the effect of air resistance on a ball ........... 642
Exercise E.27  Solve an ODE system for an electric circuit ....... 642
Exercise E.28  Compare methods for solving (E.73)—(E.74) ....... 643
Exercise E.29  Explore predator-prey population interactions ... 643
Exercise E.30  Formulate a 2nd-order ODE as a system .......... 644
Exercise E.31  Solve the system in Exer. E.30 in a special case .. 645
Exercise E.32  Enhance the code from Exer. E.31 ............... 646
Exercise E.33  Make a tool for analyzing oscillatory solutions ... 648
Exercise E.34  Replace functions by class in Exer. E.32 .......... 648
Exercise E.35  Allow flexible choice of functions in Exer. E.34 .. 652
Exercise E.36  Use the modules from Exer. E.34 and E.35 ....... 652
Exercise E.37  Use the modules from Exer. E.34 and E.35 ....... 653