

Undergraduate Topics in Computer Science

Series Editor

Ian Mackie, University of Sussex, Brighton, UK

Advisory Editors

Samson Abramsky, Department of Computer Science, University of Oxford, Oxford, UK

Chris Hankin, Department of Computing, Imperial College London, London, UK

Dexter C. Kozen, Department of Computer Science, Cornell University, Ithaca, NY, USA

Andrew Pitts, University of Cambridge, Cambridge, UK

Hanne Riis Nielson , Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kongens Lyngby, Denmark

Steven S. Skiena, Department of Computer Science, Stony Brook University, Stony Brook, NY, USA

Iain Stewart, Department of Computer Science, Science Labs, University of Durham, Durham, UK

Mike Hinchey, University of Limerick, Limerick, Ireland

'Undergraduate Topics in Computer Science' (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems, many of which include fully worked solutions.

The UTiCS concept relies on high-quality, concise books in softback format, and generally a maximum of 275–300 pages. For undergraduate textbooks that are likely to be longer, more expository, Springer continues to offer the highly regarded Texts in Computer Science series, to which we refer potential authors.

More information about this series at <http://www.springer.com/series/7592>

John Hunt

Advanced Guide to Python 3 Programming

 Springer

John Hunt
Marshfield
Midmarsh Technology Ltd.
Chippenham, Wiltshire, UK

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-030-25942-6 ISBN 978-3-030-25943-3 (eBook)
<https://doi.org/10.1007/978-3-030-25943-3>

© Springer Nature Switzerland AG 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, 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.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

For Denise, my wife.

Preface

Some of the key aspects of this book are:

1. It assumes knowledge of Python 3 and of concepts such as functions, classes, protocols, Abstract Base Classes, decorators, iterables, collection types (such as List and Tuple) etc.
2. However, the book assumes very little knowledge or experience of the topics presented.
3. The book is divided into eight topic areas; Computer graphics, Games, Testing, File Input/Output, Database Access, Logging, Concurrency and Parallelism and Network Programming.
4. Each topic in the book has an introductory chapter followed by chapters that delve into that topic.
5. The book includes exercises at the end of most chapters.
6. All code examples (and exercise solutions) are provided on line in a GitHub repository.

Chapter Organisation

Each chapter has a brief introduction, the main body of the chapter, followed by a list of online references that can be used for further reading.

Following this there is typically an *Exercises* section that lists one or more exercises that build on the skills you will have learnt in that chapter.

Sample solutions to the exercises are available in a GitHub repository that supports this book.

What You Need

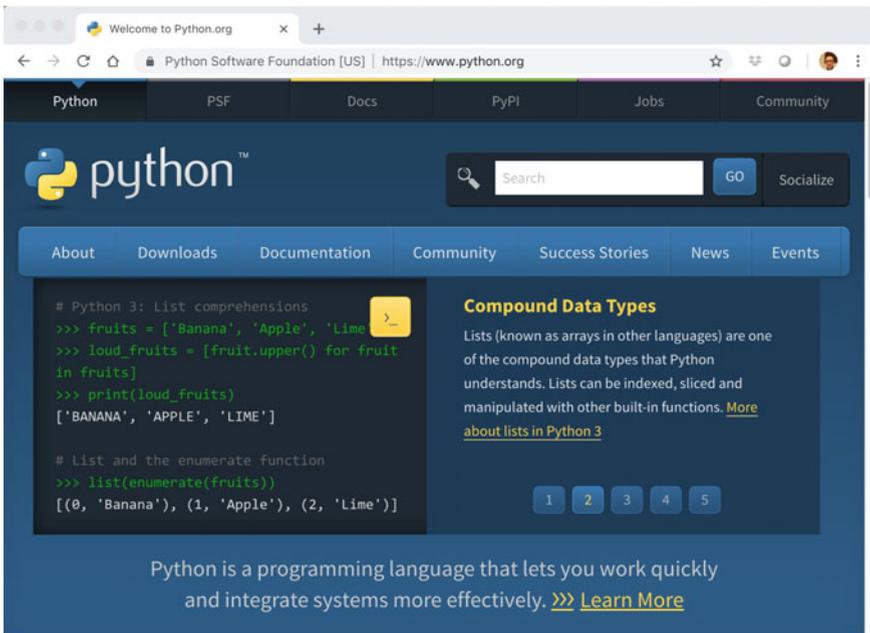
You can of course just read this book; however following the examples in this book will ensure that you get as much as possible out of the content.

For this you will need a computer.

Python is a cross platform programming language and as such you can use Python on a Windows PC, a Linux Box or an Apple Mac etc. This means that you are not tied to a particular type of operating system; you can use whatever you have available.

However you will need to install some software on your computer. At a minimum you will need Python. The focus of this book is Python 3 so that is the version that is assumed for all examples and exercises. As Python is available for a wide range of platforms from Windows, to Mac OS and Linux; you will need to ensure that you download the version for your operating system.

Python can be downloaded from the main Python web site which can be found at <http://www.python.org>.



You will also need some form of editor in which to write your programs. There are numerous generic programming editors available for different operating systems with VIM on Linux, Notepad++ on Windows and Sublime Text on Windows and Macs being popular choices.

However, using a IDE (Integrated Development Environment) editor such as PyCharm can make writing and running your programs much easier.

However, this book doesn't assume any particular editor, IDE or environment (other than Python 3 itself).

Python Versions

Currently there are two main versions of Python called Python 2 and Python 3.

- Python 2 was launched in October 2000 and has been, and still is, very widely used.
- Python 3 was launched in December 2008 and is a major revision to the language that is not backward compatible.

The issues between the two versions can be highlighted by the simple print facility:

- In Python 2 this is written as `print 'Hello World'`
- In Python 3 this is written as `print('Hello World')`

It may not look like much of a difference but the inclusion of the `()` marks a major change and means that any code written for one version of Python will probably not run on the other version. There are tools available, such as the `2to3` utility, that will (partially) automate translation from Python 2 to Python 3 but in general you are still left with significant work to do.

This then raises the question which version to use?

Although interest in Python 3 is steadily increasing there are many organisations that are still using Python 2. Choosing which version to use is a constant concern for many companies.

However, the Python 2 end of life plan was initially announced back in 2015 and although it has been postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3, it is still living on borrowed time. Python 3 is the future of the Python language and it is this version that has introduced many of the new and improved language and library features (that have admittedly been back ported to Python 2 in many cases). This book is solely focussed on Python 3.

Useful Python Resources

There are a wide range of resources on the web for Python; we will highlight a few here that you should bookmark. We will not keep referring to these to avoid repetition but you can refer back to this section whenever you need to:

- https://en.wikipedia.org/wiki/Python_Software_Foundation Python Software Foundation.

- <https://docs.python.org/3/> The main Python 3 documentation site. It contains tutorials, library references, set up and installation guides as well as Python how-tos.
- <https://docs.python.org/3/library/index.html> A list of all the builtin features for the Python language—this is where you can find online documentation for the various class and functions that we will be using throughout this book.
- <https://pymotw.com/3/> the Python 3 Module of the week site. This site contains many, many Python modules with short examples and explanations of what the modules do. A Python module is a library of features that build on and expand the core Python language. For example, if you are interested in building games using Python then pygame is a module specifically designed to make this easier.
- <https://www.fullstackpython.com/email.html> is a monthly newsletter that focusses on a single Python topic each month, such as a new library or module.
- <http://www.pythonweekly.com/> is a free weekly summary of the latest Python articles, projects, videos and upcoming events.

Each section of the book will provide additional online references relevant to the topic being discussed.

Conventions

Throughout this book you will find a number of conventions used for text styles. These text styles distinguish between different kinds of information.

Code words, variable and Python values, used within the main body of the text, are shown using a Courier font. For example:

```
This program creates a top level window (the wx.Frame) and gives it a title. It also creates a label (a wx.StaticText object) to be displayed within the frame.
```

In the above paragraph `wx.Frame` and `wx.StaticText` are classes available in a Python graphical user interface library.

A block of Python code is set out as shown here:

```
def draw_koch(size, depth):  
    if depth > 0:  
        for angle in ANGLES:  
            draw_koch(size / 3, depth - 1)
```

```

        turtle.left(angle)
    else:
        turtle.forward(size)

# Draw three sides of snowflake
for _ in range(3):
    draw_koch(SIZE_OF_SNOWFLAKE, depth)
    turtle.right(120)

```

Note that keywords are shown in bold font.

In some cases something of particular interest may be highlighted with colour:

```

# Submit the function of the pool to run
# concurrently - obtain a future from pool
print('Submitting the worker to the pool')
future = pool.submit(worker, 'A')

```

Any command line or user input is shown in italics and coloured purple; for example:

```
> cp file.p ~otheruser/projdir/srcdir/newfile.p
```

Or

```

Hello, world
Enter your name: John
Hello John

```

Example Code and Sample Solutions

The examples used in this book (along with sample solutions for the exercises at the end of most chapters) are available in a GitHub repository. GitHub provides a web interface to Git, as well as a server environment hosting Git.

Git is a version control system typically used to manage source code files (such as those used to create systems in programming languages such as Python but also Java, C#, C++, Scala etc.). Systems such as Git are very useful for collaborative development as they allow multiple people to work on an implementation and to merge their work together. They also provide a useful historical view of the code (which also allows developers to roll back changes if modifications prove to be unsuitable).

If you already have Git installed on your computer then you can clone (obtain a copy of) the repository locally using:

```
git clone https://github.com/johnehunt/advancedpython3.git
```

If you do not have Git then you can obtain a zip file of the examples using <https://github.com/johnehunt/advancedpython3/archive/master.zip>

You can of course install Git yourself if you wish. To do this see <https://git-scm.com/downloads>. Versions of the Git client for Mac OS, Windows and Linux/Unix are available here.

However, many IDEs such as PyCharm come with Git support and so offer another approach to obtaining a Git repository.

For more information on Git see <http://git-scm.com/doc>. This Git guide provides a very good primer and is highly recommended.

Acknowledgements I would like to thank Phoebe Hunt for creating the pixel images used for the StarshipMeteors game in Chap. 8.

Contents

1	Introduction	1
1.1	Introduction	1
Part I Computer Graphics		
2	Introduction to Computer Graphics	5
2.1	Introduction	5
2.2	Background	6
2.3	The Graphical Computer Era	6
2.4	Interactive and Non Interactive Graphics	7
2.5	Pixels	8
2.6	Bit Map Versus Vector Graphics	10
2.7	Buffering	10
2.8	Python and Computer Graphics	10
2.9	References	11
2.10	Online Resources	11
3	Python Turtle Graphics	13
3.1	Introduction	13
3.2	The Turtle Graphics Library	13
3.2.1	The Turtle Module	13
3.2.2	Basic Turtle Graphics	14
3.2.3	Drawing Shapes	17
3.2.4	Filling Shapes	19
3.3	Other Graphics Libraries	19
3.4	3D Graphics	20
3.4.1	PyOpenGL	20
3.5	Online Resources	21
3.6	Exercises	21

- 4 Computer Generated Art 23**
 - 4.1 Creating Computer Art 23
 - 4.2 A Computer Art Generator 25
 - 4.3 Fractals in Python 28
 - 4.3.1 The Koch Snowflake 28
 - 4.3.2 Mandelbrot Set 31
 - 4.4 Online Resources 33
 - 4.5 Exercises 33

- 5 Introduction to Matplotlib 35**
 - 5.1 Introduction 35
 - 5.2 Matplotlib 36
 - 5.3 Plot Components 37
 - 5.4 Matplotlib Architecture 38
 - 5.4.1 Backend Layer 39
 - 5.4.2 The Artist Layer 40
 - 5.4.3 The Scripting Layer 41
 - 5.5 Online Resources 42

- 6 Graphing with Matplotlib pyplot 43**
 - 6.1 Introduction 43
 - 6.2 The pyplot API 43
 - 6.3 Line Graphs 44
 - 6.3.1 Coded Format Strings 46
 - 6.4 Scatter Graph 47
 - 6.4.1 When to Use Scatter Graphs 49
 - 6.5 Pie Charts 50
 - 6.5.1 Expanding Segments 52
 - 6.5.2 When to Use Pie Charts 53
 - 6.6 Bar Charts 54
 - 6.6.1 Horizontal Bar Charts 55
 - 6.6.2 Coloured Bars 56
 - 6.6.3 Stacked Bar Charts 57
 - 6.6.4 Grouped Bar Charts 58
 - 6.7 Figures and Subplots 60
 - 6.8 3D Graphs 63
 - 6.9 Exercises 65

- 7 Graphical User Interfaces 67**
 - 7.1 Introduction 67
 - 7.2 GUIs and WIMPS 68

- 7.3 Windowing Frameworks for Python 69
 - 7.3.1 Platform-Independent GUI Libraries 70
 - 7.3.2 Platform-Specific GUI Libraries 70
- 7.4 Online Resources 71
- 8 The wxPython GUI Library 73**
 - 8.1 The wxPython Library 73
 - 8.1.1 wxPython Modules 74
 - 8.1.2 Windows as Objects 75
 - 8.1.3 A Simple Example 75
 - 8.2 The wx.App Class 76
 - 8.3 Window Classes 78
 - 8.4 Widget/Control Classes 80
 - 8.5 Dialogs 81
 - 8.6 Arranging Widgets Within a Container 82
 - 8.7 Drawing Graphics 84
 - 8.8 Online Resources 86
 - 8.9 Exercises 86
 - 8.9.1 Simple GUI Application 86
- 9 Events in wxPython User Interfaces 87**
 - 9.1 Event Handling 87
 - 9.2 Event Definitions 87
 - 9.3 Types of Events 88
 - 9.4 Binding an Event to an Event Handler 89
 - 9.5 Implementing Event Handling 89
 - 9.6 An Interactive wxPython GUI 92
 - 9.7 Online Resources 96
 - 9.8 Exercises 96
 - 9.8.1 Simple GUI Application 96
 - 9.8.2 GUI Interface to a Tic Tac Toe Game 98
- 10 PyDraw wxPython Example Application 99**
 - 10.1 Introduction 99
 - 10.2 The PyDraw Application 99
 - 10.3 The Structure of the Application 100
 - 10.3.1 Model, View and Controller Architecture 101
 - 10.3.2 PyDraw MVC Architecture 102
 - 10.3.3 Additional Classes 103
 - 10.3.4 Object Relationships 104
 - 10.4 The Interactions Between Objects 105
 - 10.4.1 The PyDrawApp 105
 - 10.4.2 The PyDrawFrame Constructor 106

- 10.4.3 Changing the Application Mode 106
- 10.4.4 Adding a Graphic Object 107
- 10.5 The Classes 108
 - 10.5.1 The PyDrawConstants Class 108
 - 10.5.2 The PyDrawFrame Class 109
 - 10.5.3 The PyDrawMenuBar Class 110
 - 10.5.4 The PyDrawToolBar Class 111
 - 10.5.5 The PyDrawController Class 111
 - 10.5.6 The DrawingModel Class 113
 - 10.5.7 The DrawingPanel Class 113
 - 10.5.8 The DrawingController Class 114
 - 10.5.9 The Figure Class 115
 - 10.5.10 The Square Class 115
 - 10.5.11 The Circle Class 116
 - 10.5.12 The Line Class 116
 - 10.5.13 The Text Class 117
- 10.6 References 117
- 10.7 Exercises 117

Part II Computer Games

- 11 Introduction to Games Programming 121**
 - 11.1 Introduction 121
 - 11.2 Games Frameworks and Libraries 121
 - 11.3 Python Games Development 122
 - 11.4 Using Pygame 123
 - 11.5 Online Resources 123
- 12 Building Games with pygame 125**
 - 12.1 Introduction 125
 - 12.2 The Display Surface 126
 - 12.3 Events 127
 - 12.3.1 Event Types 127
 - 12.3.2 Event Information 128
 - 12.3.3 The Event Queue 129
 - 12.4 A First pygame Application 130
 - 12.5 Further Concepts 133
 - 12.6 A More Interactive pygame Application 136
 - 12.7 Alternative Approach to Processing Input Devices 138
 - 12.8 pygame Modules 138
 - 12.9 Online Resources 139

- 13 StarshipMeteors pygame** 141
 - 13.1 Creating a Spaceship Game 141
 - 13.2 The Main Game Class 142
 - 13.3 The GameObject Class 144
 - 13.4 Displaying the Starship 145
 - 13.5 Moving the Spaceship 146
 - 13.6 Adding a Meteor Class 150
 - 13.7 Moving the Meteors 152
 - 13.8 Identifying a Collision 152
 - 13.9 Identifying a Win 154
 - 13.10 Increasing the Number of Meteors 154
 - 13.11 Pausing the Game 155
 - 13.12 Displaying the Game Over Message 156
 - 13.13 The StarshipMeteors Game 157
 - 13.14 Online Resources 162
 - 13.15 Exercises 162

Part III Testing

- 14 Introduction to Testing** 165
 - 14.1 Introduction 165
 - 14.2 Types of Testing 165
 - 14.3 What Should Be Tested? 166
 - 14.4 Testing Software Systems 167
 - 14.4.1 Unit Testing 168
 - 14.4.2 Integration Testing 169
 - 14.4.3 System Testing 169
 - 14.4.4 Installation/Upgrade Testing 170
 - 14.4.5 Smoke Tests 170
 - 14.5 Automating Testing 170
 - 14.6 Test Driven Development 171
 - 14.6.1 The TDD Cycle 172
 - 14.6.2 Test Complexity 173
 - 14.6.3 Refactoring 173
 - 14.7 Design for Testability 173
 - 14.7.1 Testability Rules of Thumb 173
 - 14.8 Online Resources 174
 - 14.9 Book Resources 174
- 15 PyTest Testing Framework** 175
 - 15.1 Introduction 175
 - 15.2 What Is PyTest? 175
 - 15.3 Setting Up PyTest 176
 - 15.4 A Simple PyTest Example 176

- 15.5 Working with PyTest 179
- 15.6 Parameterised Tests 183
- 15.7 Online Resources 185
- 15.8 Exercises 185
- 16 Mocking for Testing 187**
 - 16.1 Introduction 187
 - 16.2 Why Mock? 188
 - 16.3 What Is Mocking? 190
 - 16.4 Common Mocking Framework Concepts 191
 - 16.5 Mocking Frameworks for Python 192
 - 16.6 The unittest.mock Library 192
 - 16.6.1 Mock and Magic Mock Classes 193
 - 16.6.2 The Patchers 194
 - 16.6.3 Mocking Returned Objects 195
 - 16.6.4 Validating Mocks Have Been Called 196
 - 16.7 Mock and MagicMock Usage 197
 - 16.7.1 Naming Your Mocks 197
 - 16.7.2 Mock Classes 197
 - 16.7.3 Attributes on Mock Classes 198
 - 16.7.4 Mocking Constants 199
 - 16.7.5 Mocking Properties 199
 - 16.7.6 Raising Exceptions with Mocks 199
 - 16.7.7 Applying Patch to Every Test Method 200
 - 16.7.8 Using Patch as a Context Manager 200
 - 16.8 Mock Where You Use It 201
 - 16.9 Patch Order Issues 201
 - 16.10 How Many Mocks? 202
 - 16.11 Mocking Considerations 202
 - 16.12 Online Resources 203
 - 16.13 Exercises 203

Part IV File Input/Output

- 17 Introduction to Files, Paths and IO 207**
 - 17.1 Introduction 207
 - 17.2 File Attributes 209
 - 17.3 Paths 211
 - 17.4 File Input/Output 212
 - 17.5 Sequential Access Versus Random Access 213
 - 17.6 Files and I/O in Python 214
 - 17.7 Online Resources 214

- 18 Reading and Writing Files** 215
 - 18.1 Introduction 215
 - 18.2 Obtaining References to Files 215
 - 18.3 Reading Files 217
 - 18.4 File Contents Iteration 218
 - 18.5 Writing Data to Files 218
 - 18.6 Using Files and with Statements 219
 - 18.7 The Fileinput Module 219
 - 18.8 Renaming Files 220
 - 18.9 Deleting Files 220
 - 18.10 Random Access Files 221
 - 18.11 Directories 222
 - 18.12 Temporary Files 224
 - 18.13 Working with Paths 225
 - 18.14 Online Resources 229
 - 18.15 Exercise 229

- 19 Stream IO** 231
 - 19.1 Introduction 231
 - 19.2 What is a Stream? 231
 - 19.3 Python Streams 232
 - 19.4 IOBase 233
 - 19.5 Raw IO/UnBuffered IO Classes 234
 - 19.6 Binary IO/Buffered IO Classes 234
 - 19.7 Text Stream Classes 236
 - 19.8 Stream Properties 237
 - 19.9 Closing Streams 238
 - 19.10 Returning to the open() Function 238
 - 19.11 Online Resources 240
 - 19.12 Exercise 240

- 20 Working with CSV Files** 241
 - 20.1 Introduction 241
 - 20.2 CSV Files 241
 - 20.2.1 The CSV Writer Class 242
 - 20.2.2 The CSV Reader Class 243
 - 20.2.3 The CSV DictWriter Class 244
 - 20.2.4 The CSV DictReader Class 245
 - 20.3 Online Resources 246
 - 20.4 Exercises 246

- 21 Working with Excel Files** 249
 - 21.1 Introduction 249
 - 21.2 Excel Files 249

- 21.3 The Openpyxl. Workbook Class 250
- 21.4 The Openpyxl. WorkSheet Objects 250
- 21.5 Working with Cells 250
- 21.6 Sample Excel File Creation Application 251
- 21.7 Loading a Workbook from an Excel File 253
- 21.8 Online Resources 254
- 21.9 Exercises 254
- 22 Regular Expressions in Python 257**
 - 22.1 Introduction 257
 - 22.2 What Are Regular Expressions? 257
 - 22.3 Regular Expression Patterns 258
 - 22.3.1 Pattern Metacharacters 259
 - 22.3.2 Special Sequences 259
 - 22.3.3 Sets 260
 - 22.4 The Python re Module 261
 - 22.5 Working with Python Regular Expressions 261
 - 22.5.1 Using Raw Strings 261
 - 22.5.2 Simple Example 262
 - 22.5.3 The Match Object 262
 - 22.5.4 The search() Function 263
 - 22.5.5 The match() Function 264
 - 22.5.6 The Difference Between Matching and Searching 265
 - 22.5.7 The findall() Function 265
 - 22.5.8 The finditer() Function 266
 - 22.5.9 The split() Function 266
 - 22.5.10 The sub() Function 267
 - 22.5.11 The compile() Function 268
 - 22.6 Online Resources 270
 - 22.7 Exercises 270

Part V Database Access

- 23 Introduction to Databases 275**
 - 23.1 Introduction 275
 - 23.2 What Is a Database? 275
 - 23.2.1 Data Relationships 276
 - 23.2.2 The Database Schema 277
 - 23.3 SQL and Databases 279
 - 23.4 Data Manipulation Language 280
 - 23.5 Transactions in Databases 281
 - 23.6 Further Reading 282

- 24 Python DB-API** 283
 - 24.1 Accessing a Database from Python 283
 - 24.2 The DB-API 283
 - 24.2.1 The Connect Function 284
 - 24.2.2 The Connection Object 284
 - 24.2.3 The Cursor Object 285
 - 24.2.4 Mappings from Database Types to Python Types . . . 286
 - 24.2.5 Generating Errors 286
 - 24.2.6 Row Descriptions 287
 - 24.3 Transactions in PyMySQL 288
 - 24.4 Online Resources 288
- 25 PyMySQL Module** 291
 - 25.1 The PyMySQL Module 291
 - 25.2 Working with the PyMySQL Module 291
 - 25.2.1 Importing the Module 292
 - 25.2.2 Connect to the Database 292
 - 25.2.3 Obtaining the Cursor Object 293
 - 25.2.4 Using the Cursor Object 293
 - 25.2.5 Obtaining Information About the Results 294
 - 25.2.6 Fetching Results 294
 - 25.2.7 Close the Connection 295
 - 25.3 Complete PyMySQL Query Example 295
 - 25.4 Inserting Data to the Database 296
 - 25.5 Updating Data in the Database 298
 - 25.6 Deleting Data in the Database 299
 - 25.7 Creating Tables 300
 - 25.8 Online Resources 301
 - 25.9 Exercises 301

Part VI Logging

- 26 Introduction to Logging** 305
 - 26.1 Introduction 305
 - 26.2 Why Log? 305
 - 26.3 What Is the Purpose of Logging? 306
 - 26.4 What Should You Log? 306
 - 26.5 What Not to Log 307
 - 26.6 Why Not Just Use Print? 308
 - 26.7 Online Resources 309
- 27 Logging in Python** 311
 - 27.1 The Logging Module 311
 - 27.2 The Logger 312

- 27.3 Controlling the Amount of Information Logged 313
- 27.4 Logger Methods 315
- 27.5 Default Logger 316
- 27.6 Module Level Loggers 317
- 27.7 Logger Hierarchy 318
- 27.8 Formatters 319
 - 27.8.1 Formatting Log Messages 319
 - 27.8.2 Formatting Log Output 319
- 27.9 Online Resources 322
- 27.10 Exercises 322
- 28 Advanced Logging 323**
 - 28.1 Introduction 323
 - 28.2 Handlers 323
 - 28.2.1 Setting the Root Output Handler 325
 - 28.2.2 Programmatically Setting the Handler 326
 - 28.2.3 Multiple Handlers 328
 - 28.3 Filters 329
 - 28.4 Logger Configuration 330
 - 28.5 Performance Considerations 333
 - 28.6 Exercises 334
- Part VII Concurrency and Parallelism**
- 29 Introduction to Concurrency and Parallelism 337**
 - 29.1 Introduction 337
 - 29.2 Concurrency 337
 - 29.3 Parallelism 339
 - 29.4 Distribution 340
 - 29.5 Grid Computing 340
 - 29.6 Concurrency and Synchronisation 342
 - 29.7 Object Orientation and Concurrency 342
 - 29.8 Threads V Processes 343
 - 29.9 Some Terminology 344
 - 29.10 Online Resources 344
- 30 Threading 347**
 - 30.1 Introduction 347
 - 30.2 Threads 347
 - 30.3 Thread States 347
 - 30.4 Creating a Thread 348
 - 30.5 Instantiating the Thread Class 349
 - 30.6 The Thread Class 350

- 30.7 The Threading Module Functions 352
- 30.8 Passing Arguments to a Thread 352
- 30.9 Extending the Thread Class 354
- 30.10 Daemon Threads 355
- 30.11 Naming Threads 356
- 30.12 Thread Local Data 357
- 30.13 Timers 358
- 30.14 The Global Interpreter Lock 359
- 30.15 Online Resources 360
- 30.16 Exercise 360
- 31 Multiprocessing 363**
 - 31.1 Introduction 363
 - 31.2 The Process Class 363
 - 31.3 Working with the Process Class 365
 - 31.4 Alternative Ways to Start a Process 366
 - 31.5 Using a Pool 368
 - 31.6 Exchanging Data Between Processes 372
 - 31.7 Sharing State Between Processes 374
 - 31.7.1 Process Shared Memory 374
 - 31.8 Online Resources 375
 - 31.9 Exercises 376
- 32 Inter Thread/Process Synchronisation 377**
 - 32.1 Introduction 377
 - 32.2 Using a Barrier 377
 - 32.3 Event Signalling 380
 - 32.4 Synchronising Concurrent Code 382
 - 32.5 Python Locks 383
 - 32.6 Python Conditions 386
 - 32.7 Python Semaphores 388
 - 32.8 The Concurrent Queue Class 389
 - 32.9 Online Resources 391
 - 32.10 Exercises 391
- 33 Futures 395**
 - 33.1 Introduction 395
 - 33.2 The Need for a Future 395
 - 33.3 Futures in Python 396
 - 33.3.1 Future Creation 397
 - 33.3.2 Simple Example Future 397
 - 33.4 Running Multiple Futures 399
 - 33.4.1 Waiting for All Futures to Complete 400
 - 33.4.2 Processing Results as Completed 402

- 33.5 Processing Future Results Using a Callback 403
- 33.6 Online Resources 405
- 33.7 Exercises 405
- 34 Concurrency with AsyncIO 407**
 - 34.1 Introduction 407
 - 34.2 Asynchronous IO 407
 - 34.3 Async IO Event Loop 408
 - 34.4 The Async and Await Keywords 409
 - 34.4.1 Using Async and Await 409
 - 34.5 Async IO Tasks 411
 - 34.6 Running Multiple Tasks 414
 - 34.6.1 Collating Results from Multiple Tasks 414
 - 34.6.2 Handling Task Results as They Are Made Available 415
 - 34.7 Online Resources 416
 - 34.8 Exercises 417
- Part VIII Reactive Programming**
- 35 Reactive Programming Introduction 421**
 - 35.1 Introduction 421
 - 35.2 What Is a Reactive Application? 421
 - 35.3 The ReactiveX Project 422
 - 35.4 The Observer Pattern 422
 - 35.5 Hot and Cold Observables 423
 - 35.5.1 Cold Observables 424
 - 35.5.2 Hot Observables 424
 - 35.5.3 Implications of Hot and Cold Observables 424
 - 35.6 Differences Between Event Driven Programming and Reactive Programming 425
 - 35.7 Advantages of Reactive Programming 425
 - 35.8 Disadvantages of Reactive Programming 426
 - 35.9 The RxPy Reactive Programming Framework 426
 - 35.10 Online Resources 426
 - 35.11 Reference 427
- 36 RxPy Observables, Observers and Subjects 429**
 - 36.1 Introduction 429
 - 36.2 Observables in RxPy 429
 - 36.3 Observers in RxPy 430
 - 36.4 Multiple Subscribers/Observers 432
 - 36.5 Subjects in RxPy 433

- 36.6 Observer Concurrency 435
 - 36.6.1 Available Schedulers 437
- 36.7 Online Resources 438
- 36.8 Exercises 438
- 37 RxPy Operators 439**
 - 37.1 Introduction 439
 - 37.2 Reactive Programming Operators 439
 - 37.3 Piping Operators 440
 - 37.4 Creational Operators 441
 - 37.5 Transformational Operators 441
 - 37.6 Combinatorial Operators 443
 - 37.7 Filtering Operators 444
 - 37.8 Mathematical Operators 445
 - 37.9 Chaining Operators 446
 - 37.10 Online Resources 448
 - 37.11 Exercises 448

Part IX Network Programming

- 38 Introduction to Sockets and Web Services 451**
 - 38.1 Introduction 451
 - 38.2 Sockets 451
 - 38.3 Web Services 452
 - 38.4 Addressing Services 452
 - 38.5 Localhost 453
 - 38.6 Port Numbers 454
 - 38.7 IPv4 Versus IPv6 455
 - 38.8 Sockets and Web Services in Python 455
 - 38.9 Online Resources 456
- 39 Sockets in Python 457**
 - 39.1 Introduction 457
 - 39.2 Socket to Socket Communication 457
 - 39.3 Setting Up a Connection 458
 - 39.4 An Example Client Server Application 458
 - 39.4.1 The System Structure 458
 - 39.4.2 Implementing the Server Application 459
 - 39.5 Socket Types and Domains 461
 - 39.6 Implementing the Client Application 461
 - 39.7 The Socketserver Module 463
 - 39.8 HTTP Server 465
 - 39.9 Online Resources 469
 - 39.10 Exercises 469

- 40 Web Services in Python** 471
 - 40.1 Introduction 471
 - 40.2 RESTful Services 471
 - 40.3 A RESTful API 472
 - 40.4 Python Web Frameworks 473
 - 40.5 Flask 474
 - 40.6 Hello World in Flask 474
 - 40.6.1 Using JSON 474
 - 40.6.2 Implementing a Flask Web Service 475
 - 40.6.3 A Simple Service 475
 - 40.6.4 Providing Routing Information 476
 - 40.6.5 Running the Service 477
 - 40.6.6 Invoking the Service 478
 - 40.6.7 The Final Solution 479
 - 40.7 Online Resources 479

- 41 Bookshop Web Service** 481
 - 41.1 Building a Flask Bookshop Service 481
 - 41.2 The Design 481
 - 41.3 The Domain Model 482
 - 41.4 Encoding Books Into JSON 484
 - 41.5 Setting Up the GET Services 486
 - 41.6 Deleting a Book 488
 - 41.7 Adding a New Book 489
 - 41.8 Updating a Book 491
 - 41.9 What Happens if We Get It Wrong? 492
 - 41.10 Bookshop Services Listing 494
 - 41.11 Exercises 497