LEARN MORE PYTHON 3
THE HARD WAY
LEARN MORE PYTHON 3
THE HARD WAY

The Next Step for New Python Programmers

Zed A. Shaw
Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2017946529

Copyright © 2018 Zed A. Shaw

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearsoned.com/permissions/.

ISBN-10: 0-13-412348-4

1  17
Contents

 Preface ................................................................. xiv
  It’s All Personal ................................................... xv
  Using the Included Videos .................................... xv

 PART I Initial Knowledge ............................................. 2
  What If I Hate Your Stupid Personal Process Zed? .......... 3
  What If I Find Out I’m Terrible? ............................... 3

 Exercise 0 The Setup ................................................ 6
  A Programmer’s Editor ............................................ 6
  Python 3.6 ........................................................ 6
  A Working Terminal .............................................. 6
  A Working pip+virtualenv Configuration ..................... 7
  Lab Journal ........................................................ 7
  A Github.com Account .......................................... 7
  git ................................................................. 7
  Optional: Screen-Recording Software ......................... 8
  Further Study ...................................................... 8

 Exercise 1 On Process .............................................. 10
  Exercise Challenge ............................................... 11
  Study Drills ....................................................... 12
  Further Study ...................................................... 12

 Exercise 2 On Creativity ........................................... 14
  Exercise Challenge ............................................... 14
  Study Drill ........................................................ 15

 Exercise 3 On Quality .............................................. 16
  Exercise Challenge ............................................... 18
  Study Drill ........................................................ 18

 PART II Quick Hacks ................................................ 20
  How to Practice Creativity .................................... 21
  A Process for Early Coders ..................................... 22
  An Early Coder’s Coding Process ............................ 23

 Exercise 4 Dealing with Command Line Arguments .......... 24
  Exercise Challenge ............................................... 24
  Solution ........................................................... 25
  Study Drills ....................................................... 25
<table>
<thead>
<tr>
<th>Exercise</th>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><code>cat</code></td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Exercise Challenge</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Solution</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Study Drills</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Further Study</td>
<td>27</td>
</tr>
<tr>
<td>6</td>
<td><code>find</code></td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Exercise Challenge</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Study Drills</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>Further Study</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td><code>grep</code></td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Exercise Challenge</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Study Drills</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td>Further Study</td>
<td>33</td>
</tr>
<tr>
<td>8</td>
<td><code>cut</code></td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>Exercise Challenge</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Study Drill</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Further Study</td>
<td>35</td>
</tr>
<tr>
<td>9</td>
<td><code>sed</code></td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Exercise Challenge</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>Study Drills</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>Further Study</td>
<td>37</td>
</tr>
<tr>
<td>10</td>
<td><code>sort</code></td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>Exercise Challenge</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>Study Drills</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Further Study</td>
<td>39</td>
</tr>
<tr>
<td>11</td>
<td><code>uniq</code></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Exercise Challenge</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Study Drills</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>Further Study</td>
<td>41</td>
</tr>
<tr>
<td>12</td>
<td>Review</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Exercise Challenge</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Study Drills</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>Further Study</td>
<td>43</td>
</tr>
<tr>
<td>PART III</td>
<td>Data Structures</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>Learning Quality through Data Structures</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>How to Study Data Structures</td>
<td>48</td>
</tr>
</tbody>
</table>
**CONTENTS**

**Exercise 13**  
Single Linked Lists ........................................... 50  
  Description .................................................. 50  
  Controller .................................................... 52  
  Test ............................................................. 53  
  Introductory Auditing ........................................ 55  
  Exercise Challenge ........................................... 56  
  Auditing ....................................................... 56  
  Study Drill ................................................... 56

**Exercise 14**  
Double Linked Lists .............................................. 58  
  Introducing Invariant Conditions ............................ 59  
  Exercise Challenge ........................................... 60  
  Study Drill ................................................... 61

**Exercise 15**  
Stacks and Queues ............................................... 62  
  Exercise Challenge ........................................... 62  
  Breaking It .................................................... 64  
  Further Study ................................................. 64

**Exercise 16**  
Bubble, Quick, and Merge Sort ................................. 66  
  Exercise Challenge ........................................... 66  
    Study Bubble Sort ........................................ 68  
    Merge Sort .................................................. 69  
    Merge Sort Cheat Mode ................................... 70  
    Quick Sort .................................................. 71  
    Study Drills ............................................... 71

**Exercise 17**  
Dictionary ....................................................... 74  
  Exercise Challenge ........................................... 74  
  Doing a “Code Master Copy” ................................ 74  
  Copy the Code ............................................... 75  
  Annotate the Code .......................................... 78  
  Summarize the Data Structure ................................ 78  
  Memorize the Summary ...................................... 79  
  Implement from Memory ..................................... 80  
  Repeat .......................................................... 80  
  Study Drills ................................................. 81  
  Break It ....................................................... 81

**Exercise 18**  
Measuring Performance .......................................... 82  
  The Tools ..................................................... 82  
  timeit ......................................................... 82  
  cProfile and profile ....................................... 83
CONTENTS

Analyzing Performance ............................................. 84
Exercise Challenge .................................................. 86
Study Drills ............................................................. 86
Breaking It .............................................................. 86
Further Study .......................................................... 86

Exercise 19 Improving Performance ............................... 88
Exercise Challenge .................................................. 89
Further Study .......................................................... 90

Exercise 20 Binary Search Trees ................................... 92
BSTree Requirements ................................................ 92
Deleting ................................................................. 93
Exercise Challenge .................................................. 94
Study Drills .............................................................. 94

Exercise 21 Binary Search ............................................. 96
Exercise Challenge .................................................. 96
Study Drills .............................................................. 96
Further Study .......................................................... 97

Exercise 22 Suffix Arrays ............................................. 98
Exercise Challenge .................................................. 99
Study Drills .............................................................. 99
Further Study .......................................................... 99

Exercise 23 Ternary Search Trees ................................. 100
Exercise Challenge .................................................. 100
Study Drills .............................................................. 102

Exercise 24 Fast URL Search ........................................ 104
Exercise Challenge .................................................. 104
Study Drills .............................................................. 105
Further Study .......................................................... 105

PART IV Intermediate Projects ................................. 106
Tracking Your Defects ............................................... 107

Exercise 25 xargs ....................................................... 108
Exercise Challenge .................................................. 108
Study Drills .............................................................. 108

Exercise 26 hexdump .................................................... 110
Exercise Challenge .................................................. 111
Study Drill ............................................................... 112
Further Study .......................................................... 112
| Exercise 27 | tr | 114 |
| Exercise Challenge | 114 |
| A Criticism of 45-Minute Blocks | 115 |
| Study Drills | 115 |

| Exercise 28 | sh | 116 |
| Exercise Challenge | 116 |
| Study Drill | 117 |
| Further Study | 117 |

| Exercise 29 | diff and patch | 118 |
| Exercise Challenge | 118 |
| Study Drill | 119 |
| Further Study | 119 |

| PART V | Parsing Text | 120 |
| Introducing Code Coverage | 120 |

| Exercise 30 | Finite State Machines | 122 |
| Exercise Challenge | 123 |
| Study Drills | 125 |
| Further Study | 125 |

| Exercise 31 | Regular Expressions | 126 |
| Exercise Challenge | 127 |
| Study Drills | 128 |
| Further Study | 128 |

| Exercise 32 | Scanners | 130 |
| Puny Python Scanner | 131 |
| Exercise Challenge | 133 |
| Study Drills | 133 |
| Further Study | 133 |

| Exercise 33 | Parsers | 136 |
| Recursive Descent Parsing | 137 |
| BNF Grammars | 138 |
| Quick Demo Hack Parser | 140 |
| Exercise Challenge | 142 |
| Study Drill | 142 |
| Further Study | 142 |

| Exercise 34 | Analyzers | 144 |
| Visitor Pattern | 144 |
| A Short Puny Python Analyzer | 145 |
CONTENTS

Parser versus Analyzer ........................................ 148
Exercise Challenge ............................................ 148
Study Drills ..................................................... 149
Further Study ................................................. 149

Exercise 35 Interpreters ....................................... 150
Interpreters versus Compilers ............................ 150
Python Is Both ............................................... 151
How to Write an Interpreter ............................... 151
Exercise Challenge ............................................ 152
Study Drills ..................................................... 152
Further Study ................................................. 152

Exercise 36 Simple Calculator .............................. 154
Exercise Challenge ............................................ 154
Study Drills ..................................................... 155
Further Study ................................................. 155

Exercise 37 Little BASIC ...................................... 156
Exercise Challenge ............................................ 156
Study Drills ..................................................... 157

PART VI SQL and Object Relational Mapping ........ 158
Understanding SQL Is Understanding Tables .... 158
What You’ll Learn .............................................. 159

Exercise 38 Introduction to SQL ......................... 160
What Is SQL? ................................................... 160
The Setup ....................................................... 161
Learning SQL Vocabulary .................................. 162
SQL Grammar .................................................. 163
Further Study ................................................. 163

Exercise 39 Creating with SQL ............................ 164
Creating Tables ............................................... 164
Creating a Multi-table Database ....................... 165
Inserting Data .................................................. 165
Insert Referential Data ...................................... 166
Exercise Challenge ............................................ 166
Further Study ................................................. 167

Exercise 40 Reading with SQL ............................ 168
Select across Many Tables ................................. 168
<table>
<thead>
<tr>
<th>Exercise</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exercise Challenge</strong></td>
<td>169</td>
</tr>
<tr>
<td>Further Study</td>
<td>170</td>
</tr>
<tr>
<td><strong>Exercise 41</strong> Updating with SQL</td>
<td>172</td>
</tr>
<tr>
<td>Updating Complex Data</td>
<td>172</td>
</tr>
<tr>
<td>Replacing Data</td>
<td>173</td>
</tr>
<tr>
<td>Exercise Challenge</td>
<td>173</td>
</tr>
<tr>
<td>Further Study</td>
<td>174</td>
</tr>
<tr>
<td><strong>Exercise 42</strong> Deleting with SQL</td>
<td>176</td>
</tr>
<tr>
<td>Deleting Using Other Tables</td>
<td>176</td>
</tr>
<tr>
<td>Exercise Challenge</td>
<td>177</td>
</tr>
<tr>
<td>Further Study</td>
<td>178</td>
</tr>
<tr>
<td><strong>Exercise 43</strong> SQL Administration</td>
<td>180</td>
</tr>
<tr>
<td>Destroying and Altering Tables</td>
<td>180</td>
</tr>
<tr>
<td>Migrating and Evolving Data</td>
<td>181</td>
</tr>
<tr>
<td>Exercise Challenge</td>
<td>182</td>
</tr>
<tr>
<td>Further Study</td>
<td>182</td>
</tr>
<tr>
<td><strong>Exercise 44</strong> Using Python's Database API</td>
<td>184</td>
</tr>
<tr>
<td>Learning an API</td>
<td>184</td>
</tr>
<tr>
<td>Exercise Challenge</td>
<td>185</td>
</tr>
<tr>
<td>Further Study</td>
<td>185</td>
</tr>
<tr>
<td><strong>Exercise 45</strong> Creating an ORM</td>
<td>186</td>
</tr>
<tr>
<td>Exercise Challenge</td>
<td>186</td>
</tr>
<tr>
<td>Further Study</td>
<td>187</td>
</tr>
<tr>
<td><strong>PART VII</strong> Final Projects</td>
<td>188</td>
</tr>
<tr>
<td>What Is Your Process?</td>
<td>189</td>
</tr>
<tr>
<td><strong>Exercise 46</strong> blog</td>
<td>190</td>
</tr>
<tr>
<td>Exercise Challenge</td>
<td>190</td>
</tr>
<tr>
<td>Study Drills</td>
<td>191</td>
</tr>
<tr>
<td><strong>Exercise 47</strong> bc</td>
<td>192</td>
</tr>
<tr>
<td>Exercise Challenge</td>
<td>192</td>
</tr>
<tr>
<td>Study Drill</td>
<td>193</td>
</tr>
<tr>
<td><strong>Exercise 48</strong> ed</td>
<td>194</td>
</tr>
<tr>
<td>Exercise Challenge</td>
<td>194</td>
</tr>
<tr>
<td>Study Drills</td>
<td>195</td>
</tr>
<tr>
<td><strong>Exercise 49</strong> sed</td>
<td>196</td>
</tr>
<tr>
<td>Exercise Challenge</td>
<td>197</td>
</tr>
<tr>
<td>Study Drill</td>
<td>197</td>
</tr>
</tbody>
</table>
CONTENTS

Exercise 50  vi  ......................................................... 198
  Exercise Challenge .................................................. 198
  Study Drills .......................................................... 199

Exercise 51  lessweb .................................................. 200
  Exercise Challenge .................................................. 200
  Breaking It ......................................................... 200
  Study Drills .......................................................... 201

Exercise 52  moreweb .................................................. 202
  Exercise Challenge .................................................. 202
  Breaking It ......................................................... 203
  Further Study ...................................................... 203

Index ................................................................. 204
Preface

Process, creativity, and quality. Burn these three words into your mind while you read this book. Process. Creativity. Quality. This book may be full of exercises that teach important topics every programmer should know, but the real knowledge you’ll gain from the book is these three words. My goal in writing this book on programming is to teach you what I’ve known to be the three most important constants in software. Without process you’ll flounder around wondering how to get started and have problems maintaining progress on long projects. Without creativity you’ll be unable to solve the problems you’ll encounter every day as a programmer. Without quality you’ll have no idea if anything you’re doing is any good.

Teaching you these three concepts is easy. I could simply write three blog posts and say, “There ya go, now you know what those three words mean.” That isn’t going to make you a better programmer and definitely not a person who can work on their own as a developer for the next 10 or 20 years. Simply knowing about process doesn’t mean you can actually apply it in real practice. Reading a blog post about creativity doesn’t help you find out how you are creative with code. To really understand these complex topics you’ll want to internalize them, and the best way to do that is to apply them to simple projects.

As you work through the exercises in this book I will tell you which of the three you’ll be working on. This is a change from my other books where I try to be sneaky and have you learn concepts without your realization. I’m going to be explicit this time because it’s important that you keep the concept firmly in your mind so you can practice it throughout the exercise. You will then evaluate how well your attempt at applying the practice worked and what you can do to improve the next time. A key component of this book is the ability to reflect on your own capabilities objectively and improve yourself. You do this best by being focused on one technique or practice at a time while accomplishing some other goal.

In addition to process, creativity, and quality you’ll also learn what I consider six important topics a modern programmer needs to function. These may change in the future, but they’ve been essential for decades now, so unless there’s a drastic shift in technology they’ll still apply. Even something like SQL in Part VI is still relevant because it teaches you how to structure data so that it doesn’t logically fall apart later. Your secondary educational goals are the following:

1. Getting Started: You learn quick hacks to start a project.
2. Data Structures: I don’t teach every single data structure, but I get you started down the path to learning them more completely.
3. Algorithms: Data structures are fairly pointless without a way to process them.
4. Parsing Text: The foundation of computer science is parsing, and knowing how to do that helps you learn programming languages as they become popular.
5. Data Modeling: I use SQL to teach you the basics of modeling stored data in a logical way.

6. Unix Tools: Command line tools are used throughout the book as projects for you to copy, and you then also learn advanced Unix command line tools.

At each part of the book you’ll focus on one or two of the three practices at a time until finally, in Part VII, you’ll apply them all as you build a simple website. The final projects aren’t sexy. You won’t learn how to create your next startup, but they are nice little projects that will help you apply what you know while learning Django.

It’s All Personal

Many other books are designed to teach you these three concepts in the context of a team. When these books teach you about process it’s all about how you work with another person on a project to maintain code. When they teach creativity it’s all about how you go to meetings with your team to ask customers questions. Sadly most of these “professional” books don’t really teach quality. This is all fine, but there’s two problems with these team-style books for most beginners:

1. You don’t have a team, so you can’t practice what they’re teaching. The team-oriented books are designed for junior programmers who already have jobs and need to work on the team they just joined. Until that happens to you, any team-oriented book is fairly useless to you.

2. What’s the point of learning how to work on a team if your own personal process, creativity, and quality is a total mess? Despite what the fans of “team players” say, the vast majority of programming tasks are done solo, and your evaluation of your skills is usually done solo. If you work on a team, but your code is always low quality and you constantly have to ask team members for help, you get a low review from your boss. For all their talk of how awesome teams are, they never blame the team when a junior programmer can’t work alone. They blame the junior programmer.

This book is not about being a good worker drone at Mega Enterprise, LLC. This book is about helping you improve your personal skills so that when you get a job you can work alone. If you improve your personal process then it makes sense that you’ll be a stronger contributor on a team. It also means you can start and develop your own ideas, which is where the vast majority of projects start.

Using the Included Videos

*Learn More Python 3 the Hard Way* has an extensive set of videos demonstrating how the code works, debugging, and, most importantly, solutions to the challenges. The videos are the perfect place to demonstrate many common errors by breaking the Python code on purpose and showing you how to fix it. I also walk through the code using debugging and interrogation tricks and techniques. The videos
are where I show you how to “stop staring and ask” the code what’s wrong. You can watch these videos online at informit.com/title/9780134123486.

Register your copy of Learn More Python 3 the Hard Way on the InformIT site for convenient access to updates and corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account. Enter the product ISBN (9780134123486) and answer the simple proof-of-purchase question. Then look on the Registered Products tab for an Access Bonus Content link next to this product, and follow that link to access the bonus materials.
find

Hopefully you are discovering the various ways you sabotage yourself even before you begin to work. Maybe it’s not that dramatic, but you should at least be identifying things you can improve in your environment that are making it difficult for you to start working. These little exercises are a good way for you to focus on the beginning since they are not that important and fit into a time scale that you can analyze. If these projects were hours long, you’d get bored reviewing what you’ve done and making improvements. A short 45-minute project is something you can take notes about (or record) and review very quickly.

This is a pattern I use throughout my studies. I’ll identify something that I need to improve on, such as how I get started, or how I handle a tool. Then I’ll devise an exercise that simply focuses on that. When I was learning to paint I struggled with going outside to paint trees. I sat down and looked at the problems, and the first thing I identified was I simply dragged too much stuff with me. I also kept all my things in random places around my apartment. I purchased a specific bag just for my painting supplies and kept that bag ready to go. When I wanted to paint outside I grabbed this bag and walked to one of a few places, rather than planning elaborate painting hikes. I practiced just grabbing my bag, walking to one of two places, setting up, doing a painting, then walking home until the process was smooth as silk. After that I watched Bob Ross to figure out how to paint trees because that guy can crank out some trees.

This is what you should be doing. One place many people waste time and effort is in their work area. Do you have a dedicated place to work that never changes? I ditched my laptop and now just use a desktop machine so that I can have a consistent place to do my work. This also saved my back and neck from hauling around that chunk of metal and gave me a bigger screen to work with, all improving my ability to work. In this exercise, I want you to focus on your work area and make sure that it’s ready to go before you begin:

1. Do you have enough light? Do you need less light?
2. How’s your chair? Do you need a better keyboard?
3. What other tools are getting in the way? Are you trying to do Unix-like things on a Windows machine? Trying to do Mac things on Linux? Don’t go buy a new computer, but consider it for your next big purchase if you find there’s just too much friction for what you want to do.
4. How’s your desk? Do you even have one? Do you hack in cafés all day with terrible chairs and too much coffee?
5. How about music? Do you listen to music with words? I find that if I listen to music without words it’s easier for me to focus on the voice in my head that helps me write or code.
6. Do you work in an open plan office and your coworkers are annoying? Go buy yourself a pair of big over-the-ear headphones. When you wear them it’s obvious you’re not paying attention,
so people will leave you alone and they'll feel it's less rude than if you're plugged in and they can't see. This will also block out distractions and help you focus.

Spend this exercise thinking about topics like this and trying to simplify and enhance your environment. One thing, though: Don't go buying crazy contraptions and spending tons of money. Just identify problems, and then try to find ways to fix them.

Exercise Challenge

In this challenge you are implementing a basic version of the find tool for finding files. You run find like this:

```bash
find . -name "*.txt" -print
```

That will search the current directory for every file ending in .txt and print it out. find has an insane number of command line arguments, so you are not expected to implement them all in one 45-minute session. The general format of find is the following:

1. The directory to start searching in: . or /usr/local/
2. A filter argument like -name or -type d (files of type directory)
3. An action to do with each found file: -print

You can do useful things like execute a command on every found file. If you want to delete every Ruby file in your home directory you can do this:

```bash
find . -name "*.rb" -exec rm {} \;
```

Please don't run this without realizing it will delete all the files that end in .rb. The -exec argument takes a command, replaces any instance of {} with the name of the file, and then stops reading the command when it hits a ; (semicolon). We use \; in the preceding command because bash and many other shells use ; as part of their language, so we have to escape it.

This exercise will really test your ability to use either argparse or sys.argv. I recommend you run man find to get a list of arguments, and then try using find to figure out exactly what arguments you'll implement. You only have 45 minutes, so you probably can't get too many, but -name and -type are both essential as well as -print and -exec. The -exec argument will be a challenge though, so save it for last.

When you implement this, try to find libraries that can do the work for you. You'll definitely want to look at the subprocess module and also the glob module. You will definitely want to look at os more carefully as well.
Study Drills

1. How much of `find` did you get implemented?
2. What are the libraries you found to improve this implementation?
3. Did you count finding libraries as part of your 45 minutes? You could say that research before you start hacking doesn’t count, and I’d be alright with that. If you want the extra challenge, then include your research in the 45 minutes.

Further Study

How much of `find` can you implement in more 45 minute hacks? Maybe make this your hacking warmup challenge for the next week to see what you can get done. Remember that you should be trying to slap together the best ugly hack you can. Don’t worry, I won’t tell the Agile people you are just having fun.
This page intentionally left blank
Reading with SQL

Out of the CRUD matrix you only know create. You can create tables and you can create rows in those tables. I'll now show you how to read, or in the case of SQL, SELECT:

```
1. SELECT * FROM person;
2. SELECT name, age FROM pet;
3. SELECT name, age FROM pet WHERE dead = 0;
4. SELECT * FROM person WHERE first_name != "Zed";
```

Here’s what each of these lines does:

- **Line 1**: This says “select all columns from person and return all rows.” The format for SELECT is `SELECT what FROM tables(s) WHERE (tests)`, and the WHERE clause is optional. The `*` (asterisk) character is what says you want all columns.

- **Line 3**: In this one I’m only asking for two columns, name and age, from the pet table. It will return all rows.

- **Line 5**: Now I’m looking for the same columns from the pet table, but I’m asking for only the rows where `dead = 0`. This gives me all the pets that are alive.

- **Line 7**: Finally, I’m selecting all columns from `person` just like in the first line, but now I’m saying only if they do *not* equal “Zed.” That WHERE clause is what determines which rows to return or not.

Select across Many Tables

Hopefully you’re getting your head around selecting data out of tables. Always remember this: SQL ONLY KNOWS TABLES. SQL LOVES TABLES. SQL ONLY RETURNS TABLES. TABLES. TABLES. TABLES! I repeat this in this rather crazy manner so that you will start to realize that what you know in programming isn’t going to help. In programming you deal in graphs, and in SQL you deal in tables. They’re related concepts, but the mental model is different.

Here’s an example of where it becomes different. Imagine you want to know what pets Zed owns. You need to write a SELECT that looks in person and then “somehow” finds my pets. To do that you have to query the `person_pet` table to get the id columns you need. Here’s how I’d do it:
Now, this looks huge, but I'll break it down so you can see it's simply crafting a new table based on data in the three tables and the WHERE clause:

Line 1  I only want some columns from pet, so I specify them in the select. In the last exercise you used * to say “every column” but that's going to be a bad idea here. Instead, you want to be explicit and say what column from each table you want, and you do that by using table.column as in pet.name.

Line 2  To connect pet to person I need to go through the person_pet relation table. In SQL that means I need to list all three tables after the FROM.

Line 3  Start the WHERE clause.

Line 4  First, I connect pet to person_pet by the related id columns pet.id and person_pet.id.

Line 5  AND I need to connect person to person_pet in the same way. Now the database can search for only the rows where the id columns all match up, and those are the ones that are connected.

Line 6  AND I finally ask for only the pets that I own by adding a person.first_name test for my first name.

Exercise Challenge

1. Write a query that finds all pets older than 10 years.
2. Write a query to find all people younger than you. Do one that's older.
3. Write a query that uses more than one test in the WHERE clause using the AND to write it. For example, WHERE first_name = "Zed" AND age > 30.
4. Do another query that searches for rows using three columns and uses both AND and OR operators.
5. This may be a mind-blowing, weird way to look at data if you already know a language like Python or Ruby. Take the time to model the same relationships using classes and objects, then map it to this setup.
6. Do a query that finds your pets you’ve added thus far.
7. Change the queries to use your person.id instead of the person.name like I’ve been doing.
8. Go through the output from your run, and make sure you know what table is produced for which SQL commands and how they produced that output.

Further Study

Continue to deep dive into SQLite3 by reading the documentation for the SELECT command at https://sqlite.org/lang_select.html and also read the documentation for the EXPLAIN QUERY PLAN feature at https://sqlite.org/eqp.html. If you ever wonder why SQLite3 did something, EXPLAIN is your answer.
This page intentionally left blank
Symbols

. (period), accept single character input, 126
| (pipe), 117
+ (plus sign), accept if regex has one or more of
  previous character, 126
$ (dollar sign), anchor end of string, 126
( ) (parentheses), capture enclosed part of, 126
* (asterisk), accept or skip character repeatedly,
  126
? (question mark), previous part of regular
  expression is optional, 126
[X-Y], class or range of characters from
  X to Y, 126
^ (caret), anchor beginning of string, 126
> (less than), function of, 27

Numbers

3P (Personal Process Practice), 38
45-minute time limit
  benefits of, 24
  counting/not counting research time in, 30
  criticism of, 115
  failure and, 25
  practicing creativity, 21–22
  recording/annotating, 28
  in review of project strategy, 42
  time management and, 32
  warm up hack, 26

A

ABNF. See Augmented BNF (ABNF)
Agile, example of Team Process, 10
Algebraic calculator, 154–155
Algorithms
  binary search. See Binary search algorithm
bugs persisting over long time periods, 90
formal study of, 48
further study, 105
improving performance using, 88
memorize, attempt, check process, 48–49
for sorting. See Sorting algorithms
suffix array, 98–99
suffix tree, 98
ALTER TABLE
  destroying/altering tables, 180–181
  exercise administering databases, 182
analyze() methods, 148
Analyzers
  comparing with parsers, 148
  creating simple calculator, 154–155
  example, 145–148
  exercise challenge, 148
  overview of, 144
  study drills and further study, 149
  visitor pattern use with, 144–145
Annotate, CASMIR
  Dictionary code, 78
  overview of, 74
API, database API, 184–185
argparse package
  exercise using find command, 29
  exercise working with command line
  arguments, 24–25
Arguments. See Command line arguments
Arrays, suffix array, 98–99
ASCII, viewing file contents when not in text
  format, 111
assert calls, repetitive use, 60
asyncio module, 202
Atom text editor, 6
INDEX 205

Auditing
conducting basic code audits, 55–56
implanting xargs, 108–109
reviewing code critically, 47
tracking defects, 107

Augmented BNF (ABNF)
applying BNF grammars, 138–141
creating interpreter, 157
creating simple calculator, 154–155
studying RFC 7230, 202

Automation, how to proactive creativity, 21

Back ache, focusing on work habits and behaviors, 35

Backus-Naur Form (BNF)
augmented. See Augmented BNF (ABNF)
creating simple calculator, 154–155

bash, running programs from Terminal, 116

BASIC, implementing BASIC interpreter, 156–157
bc language
creating simple calculator, 154
exercise challenge, 192–193
further study, 192–193
overview of, 192

Behavior-driven development, 115

Binary search algorithm
exercise challenge, 96
overview of, 96
study drills and further study, 96–97

Binary search trees (BSTree)
binary search algorithm compared with, 96
creating tree from characters, 136–137
deleting nodes, 93
exercise challenge, 94
overview of, 92–93
study drills, 94

Suffix tree compared with, 98
ternary search trees compared with, 100
walking the tree, 144

blog tool
exercise challenge, 190–191

Bubble Sort
cProfile applied to, 83–84
exercise challenge, 66–68
overview of, 66
reasons for avoiding, 88
studying, 68–69
timeit module applied to, 82–83

Bugs
auditing SingleLinkedList, 55–56
benefit of external review, 16–17
cleaning up code, 43
invariant use in testing, 60
lurking in algorithms over long time periods, 90
“off by one” error, 64
printing debugging output, 51
quality as low defect rate, 46–47
separating debugging from performance analysis, 85
tracking defects, 107

C
C language, handling socket connections and, 203
cal, 44

Calculators
creating algebraic calculator, 154
exercise challenge, 154–155
study drills and further study, 155
CASMIR (copy, annotate, summarize, memorize, implement, repeat)
  annotate Dictionary code, 78
  copy Dictionary code, 75–78
  implement Dictionary code, 80
  memorize Dictionary code, 79–80
  repeat Dictionary code, 80–81
  summarize Dictionary code, 78–79

cat
  creating replica of, 26
  exercise challenge and solution, 26–27
  study drills and further study, 27

Chair
  focusing on work habits and behaviors, 34–35
  making improvements to work area, 28

Challenge Mode
  process of working with exercises in book, 3
  purpose of, 24

Characters. See also Text
  parsing, 136–137
  in regular expressions, 126
  tr tool for translating character streams, 114

cheat mode, merge sort, 70–71

Checklists
  practicing creativity, 21
  process for early coders, 22–23

chroot function, 201

Clients, http.client, 200–202

Code
  auditing, 55–56
  auditing critically, 47
  creating code master copy, 74–75
  erasing and starting over, 21
  performance analysis, 85
  process for early coders, 22–23
  quality as understandable code, 46–47
  removing repetition from code, 43
  testing run time using cProfile, 83
  tracking defects, 107

Code coverage
  overview of, 120–121
  testing, 196

Columns
  comparing SQL to Excel, 161
  exercise administering database, 182

Command line arguments. See also by individual commands
dealing with, 24
  exercise challenge and solution, 24–25
  study drills, 25

Compilers, interpreters compared with, 150

Computers, making improvements to work area, 28

Concatenate. See cat

Concentration, programming and, 115

Controllers
  data structure concepts, 50–51
  SingleLinkedList, 52
  Stack control class, 63

Copy, annotate, summarize, memorize, implement, repeat. See CASMIR (copy, annotate, summarize, memorize, implement, repeat)

Copy command, CASMIR, 75–78

Copying code, from dictionary, 75–78

count(), avoiding unneeded repetitive calculations, 88

CProfile, 83–84

CPUs, performance analysis, 85

CREATE
  creating multi-table database, 164–165
  creating tables, 164
  SQL operation, 161–162

Create, read, update, delete. See CRUD (create, read, update, delete)

CREATE TABLE, evolving schema to new form, 181–182

Creativity
  defined, 2
  exercise challenge, 14–15
  fluidity and relaxation as key to, 46
  how to practice, 21–22
  not letting metrics undermine, 38
  overview of, 14
  starting as enemy of, 20–21
  study drill, 15
Critical thinking
  auditing code, 47
  balancing with creativity, 46
CRUD (create, read, update, delete)
  creating an ORM, 187
  creating tables, 164
  deleting data, 176–178
  reading (selecting) data, 168–169
  updating data, 172–174
csh, running programs from Terminal, 116
curses module, 198
cut
  exercise challenge and study drills, 35
  removing duplicates from list, 40

D
  da Vinci, Leonardo, 16–17
Data
  inserting into database, 165–166
  migrating and evolving, 181–182
  replacing using DELETE/INSERT, 173
  updating complex data, 172–173
Data structures. See also by individual types
  breaking, 64
  further study, 105
  how to study, 48–49
  improving performance by using built-in data structures, 89
  improving performance by using correct data structure, 88
  learning quality through, 47
  overview of, 46–47
Database API
  exercise challenge, 185
  further study, 185
  learning an API, 184–185
  overview of, 184
Databases. See also SQL
  comparing SQL to Excel, 161
  creating multi-table database, 164–165
Debugging
  invariants and, 60
  printing debugging output, 51
  separating from performance analysis, 85
Defects
  auditing SingleLinkedList, 55–56
  benefit of external review, 16–17
  fixing, 47
  invariant use in testing, 60
  lurking in algorithms over long time periods, 90
  “off by one” error, 64
  printing debugging output, 51
  quality as low defect rate, 46–47
  separating debugging from performance analysis, 85
  tracking, 107
DELETE
  replacing data using DELETE/INSERT, 173
  SQL operation, 161–162
delete, binary search tree operations, 93
Desk, making improvements to work area, 28
dict class, 74
Dictionary
  annotate, 78
  binary search tree compared with, 92
  breaking data structures, 81
  copy, 75–78
  creating code master copy, 74–75
  exercise challenge, 74
  fast URL search, 105
  implement, 80
  improving performance by using correct data structure, 88
  learning quality through, 47
  memorize, 79–80
  overview of dict class, 74
  repeat, 80–81
  study drills, 81
  summarize, 78–79
diff command
  exercise challenge, 118–119
  overview of, 118
  study drills and further study, 119
Directories
  searching with find, 29
  unwanted directory traversal, 200–201
Divide and conquer algorithm, 66
Django, 187
Domains, attacking (breaking) web server, 201
Don't Repeat Yourself (DRY), 43
DoubleLinkedList
  binary search, 96–97
  exercise challenge, 60
  improving performance by using correct data structure, 88
  invariant conditions, 59–60
  overview of, 58–59
  queue compared with, 64
  study drills, 61
DROP
  removing tables (DROP TABLE), 180
  SQL operation, 162
DRY (Don't Repeat Yourself), 43
Duplicates, removing from list, 40

E
ed command
  creating curses UI, 198–199
  exercise challenge, 194
  overview of, 194
  study drill, 195
  testing, 196
Edges (pointers or links)
  data structure concepts, 50–51
  in DoubleLinkedList, 58
Effective TCP/IP Programming (Snader), 203
Efficiency, inspecting use of time, 32
Emacs text editor, 6
Ergonomics, focusing on work habits and behaviors, 34–35
Errors
  auditing SingleLinkedList, 55–56
  benefit of external review, 16–17
  cleaning up code, 43
  exercise using analyzers, 148
  invariant use in testing, 60
  lurking in algorithms over long time periods, 90
  “off by one,” 64
  printing debugging output, 51
  quality as low defect rate, 46
  separating debugging from performance analysis, 85
  in studying data structures, 48
  tracking, 107
Events
  as function of subclass, 125
  organizing as set of states, 122–123
Excel, comparing SQL to, 161
exec argument
  executing actions on multiple files, 29
  inspecting use of time, 32
Exercises, process of working with exercises in book, 2–3
Expression-based languages, vs.
  statement-based, 149
eXtreme Programming, example of Team Process, 10

F
false, applying 45-minute hacks, 44
Fast URL search
  exercise challenge, 104–105
  overview of, 104
  study drills and further study, 105
Fear, blocking learning, 27
Files
  dumping content to screen, 26
  finding .txt files, 29
  searching for text patterns using regular expressions, 32–33
  viewing file contents when not in text format, 111
Filters, using in searches, 29
find
  exercise challenge, 29
  implanting xargs, 108
  pattern for locating improvements, 28–29
  study drills and further study, 30
find_all
  suffix array searches, 99
ternary search trees (TSTree), 101
**find_longest**
- suffix array searches, 99
- ternary search trees (TSTree), 101

**find_part**, ternary search trees (TSTree), 101

**find_shortest**
- suffix array searches, 99
- ternary search trees (TSTree), 101

**Finite state machines (FSMs)**
- exercise challenge, 123–125
- handling modal nature of ed command, 194
- overview of, 122–123
- study drills and further study, 125

**fish**, running programs from Terminal, 116

**Flags**, exercise working with command line arguments, 24

**Flash cards**, learning an API, 184

**Formats**, viewing file contents when not in text format, 111

**Friction. See also Problem identification**
- eliminating in projects, 21
- solving problems creatively, 46

**FROM**, SQL operation, 162

**FSM. See Finite state machines (FSMs)**

**FSMRunner class**, 125

**Functions**
- converting loose hack into set of, 43
- defining for bc language, 192
- exercise using analyzers, 148
- os module, 201

**G**

**get**, binary search tree operations, 92

**GET**, unwanted HTTP requests, 201

**git**, setup requirements for book exercises, 7

**glob module**, exercise using find command, 29

**Grammars**
- analyzing semantics, 144
- Backus-Naur Form (BNF), 138–139
- creating parser for, 137
- creating simple calculator, 154–155
- parser enforcing, 136
- SQL, 163
- studying RFC 7230, 202

**Graphs**, tracking defects, 107

**grep**
- exercise challenge, 32–33
- searching for text patterns using regular expressions, 32
- study drills and further study, 33

**H**

**Habits**, inspecting use of time, 32

**Hacks/hacking**
- benefit of external review, 16–17
- determining what your process is, 189
- keeping it loose and flowing, 22
- making clean beginning, 42–43
- process for early coders, 22–23
- working with command line arguments, 25

**Healthy habits**, 34–35

**Help**, exercise working with command line arguments, 24

**hex function**, viewing file contents when not in text format, 111

**hexadump**
- exercise challenge, 111
- overview of, 110
- study drills and further study, 112

**history**
- printing out list of commands that you’ve run, 40
- tr tool for determining frequency of word use, 114

**HTML pages**, template for, 190

**HTTP**
- attacking (breaking) web server, 201
- parsers for, 202–203
- requests, 202
- http.client, 200–201
- http.server, 200–203

**I**

**IDE**, text editors compared with, 6

**if-statement**
- defining for bc language, 192
- handling branching, 122
Implement command, CASMIR
  Dictionary code, 80
  overview of, 75
IN, SQL operation, 162
“input triggers,” events as, 122–123
INSERT
  inserting data, 165–166
  replacing data using DELETE/INSERT, 173
  SQL operation, 162
inspect module, viewing Python objects and
classes, 125
int function, viewing file contents when not in
text format, 111
interpret method, 151–152
Interpreters
  compilers compared with, 150–151
  creating simple calculator, 154–155
  exercise challenge, 152
  how to write, 151–152
  implementing BASIC interpreter, 156–157
  overview of, 150
  study drills and further study, 152
  using regular expressions to match tokens, 130
Invariant conditions, DoubleLinkedList,
  59–60
J
Journal. See Lab Journal
K
KCachegrind, performance analysis tool, 86
Keyboard, making improvements to work area, 28
L
Lab Journal
  improving mental attitude by observing fears, 27
  process of working with exercises in book, 3
  reviewing process in, 196
  setup requirements for book exercises, 7
lessweb
  attacking (breaking) web server, 200–201
  creating web server, 200
  exercise challenge, 200
  study drills, 201
lex tool, researching, 193
Libraries
  asyncio module, 202
  exercise using find command, 29–30
  researching charting libraries, 41
Lighting, making improvements to work area, 28
Linked lists
  double. See DoubleLinkedList
  single. See SingleLinkedList
Links (pointers or edges)
  data structure concepts, 50–51
  in DoubleLinkedList, 58
  list operation, binary search, 93, 96
Lists. See also Checklists
  binary search, 96
  sorting, 66
Loops, avoiding loops inside loops, 88
ls
  applying 45-minute hacks, 43
  sorting text, 38–39
M
man
  accessing manual pages, 108
  investigating tr command, 115
Markdown, as blogging format, 190
Master copy
  creating code master copy, 74–75
  of ed command, 194
match function, of recursive parser, 138
Math operators, defining for bc language, 192
Memorize, attempt, check, in studying data
  structures, 48–49
Memorize command, CASMIR
  Dictionary code, 79–80
  overview of, 75
Merge sort
  binary search tree compared with, 92
  cheat mode, 70–71
  exercise challenge, 66–68
improper use of recursion, 89
improving performance by using correct algorithm, 88
overview of, 66
performance analysis, 84–86
study drills, 71–72
studying, 69–70

Metrics
building Personal Process Practice (3P), 38
determining what your process is, 189
evaluating and improving work, 36
improving performance, 89
for quality, 47
in review of project strategy, 42

Microsoft Excel, comparing SQL to, 161
Migration, evolving database schema to new form, 181–182
mkdir, applying 45-minute hacks, 43
moreweb
attacking (breaking) web server, 203
exercise challenge, 202
further study, 203
overview of, 202

Music, making improvements to work area, 28

N
Nodes
comparing stacks and queues, 62
data structure concepts, 50–51
deleting, 93
in DoubleLinkedList, 58

O
Object Oriented Programming (OOP), 187
Object Relational Mapper (ORM)
creating, 159, 186
exercise challenge, 186–187
further study, 187
oct function, viewing file contents when not in text format, 111
od command, exercise reusing hexdump code, 112
“off by one” errors, 64

OOP (Object Oriented Programming), 187
OpenSSL project, bugs lurking in algorithms over long time periods, 90
Operators, defining for bc language, 192
Options, exercise working with command line arguments, 25
ord function, viewing file contents when not in text format, 111
ORM. See Object Relational Mapper (ORM)

P
P-code (pseudo-code), implementing algorithm based on, 66–68

Parsers
BNF grammars, 138–139
comparing analyzer with, 148
creating simple calculator, 154–155
example of recursive descent parser, 140–141
exercise challenge, 142
for HTTP, 202–203
implementing bc language, 192–193
overview of, 136–137
recursive descent parsing, 137–138
study drills and further study, 142

Parsing text
analyzers, 144–149
code coverage, 120–121
creating simple calculator, 154–155
finite state machines, 122–125
implementing BASIC interpreter, 156–157
interpreters, 150–152
overview of, 120
parsers, 136–142
regular expressions, 126–128
scanners, 130–134

patch command
exercise challenge, 118–119
overview of, 118
study drills and further study, 119
peek function, of recursive parser, 138
Performance analysis
analyzing performance, 84–86
binary searches, 96
cProfile and profile, 83–84
data structure exercise, 64
exercise challenge, 86
overview of, 82
study drills and further study, 86
timeit module, 82–83
Performance analyzer, running, 85
Performance improvement/tuning
exercise challenge, 89
further study, 90
overview of, 88–89
Personal Process Practice (3P), 38
Personal Processes
overview of, 10–11
Personal Process Practice (3P), 38
pip, setup requirements for book exercises, 7
Pipe (.), 117
Pointers (edges or links)
data structure concepts, 50–51
in DoubleLinkedList, 58
pop
comparing stacks and queues, 62
inefficiencies of SingleLinkedList, 58
Positional arguments, exercise working with
command line arguments, 25
POSIX, file redirection features in Terminal, 26
POST, HTTP requests, 201
Posture, focusing on work habits and behaviors,
34–35
prev, in DoubleLinkedList, 58
print, printing files, 29
Problem identification
and correction, 28
in review of project strategy, 42
solving problems creatively, 46
Process
defined, 2
determining what your process is, 189
for early coders, 22–23
exercise challenge, 11–12
means of moving through (slogging through) a
project, 20
Personal Processes, 10–11
reviewing how it is working, 196
study drill and further study, 12
Team Processes, 10
Productivity, improving. See also Performance
analysis, 38
profile, 83–84
Programmer done, vs. quality, 46–47
Programming, concentration in, 115
Programming languages
bc language, 192–193
expression-based vs. statement-based, 149
implementing BASIC interpreter, 156–157
Pseudo-code (p-code), implementing algorithm
based on, 66–68
push
comparing stacks and queues, 62
inefficiencies of SingleLinkedList, 58
pyprof2calltree, performance analysis tool,
86
pytest
for Dictionary, 77–78
for sorting algorithms, 68–71
Python 3.6, setup requirements, 6

Q
Quality
balancing creativity with, 46
defined, 2, 46
exercise challenge, 18
learning through data structures, 47
overview of, 16–17
study drills, 18
Query, exercise administering database, 182
Queues
breaking data structures, 64
exercise challenge, 62–63
further study, 64
overview of, 62
Quick sort
exercise challenge, 66–68
improving performance by using correct
algorithm, 88
overview of, 66
study drills, 71–72
studying, 71

R
RDP. See Recursive descent parser (RDP)
re module, handling regular expressions, 33
READ, SQL operation, 161–162
readline, implementing sh command, 116–117
Recursion, improper use of, 89
Recursive descent parser (RDP). See also
Parsers
example of recursive descent parser, 140–141
implementing bc language, 192–193
overview of, 137–138
Referential data, inserting into database, 166
Regular expressions
altering text using regular expression
replacement pattern, 36–37
exercise challenge, 127
overview of, 126–127
parsers and, 136–137
scanning text for tokens, 130
searching for text patterns using, 32–33
study drills and further study, 128
Repeat command, CASMIR
Dictionary code, 80–81
overview of, 75
REPLACE, replacing data in SQL database, 173
repl(), printing debugging output, 51
Reusability, of software, 196
Review, benefit of external review, 16–17
Review of project strategy
exercise challenge, 42–43
overview of, 42
study drills and further study, 43–44
RFC 7230, 202
Rituals, inspecting use of time, 32
rm, applying 45-minute hacks, 43
rmdir, applying 45-minute hacks, 43
Routing, fast URL search, 104–105
Rows
comparing SQL to Excel, 161
deleting, 176–177
rsync tool, sending blog to server, 190
Run charts
of percentage of features completed, 38
researching charting libraries, 41
for spotting changes in behavior, 37
S
Scanners
brain operating as, 136
combining Scanner class with Parser class, 142
creating simple calculator, 154–155
define example scanning Python code, 131–132
define exercise challenge, 133
define overview of, 130–131
define study drills and further study, 133–134
Schema, evolving database schema to new form, 181–182
Screen recording software, 8
Scripts, exercise working with command line
arguments, 24–25
Scrum, example of Team Process, 10
Searches
binary search, 96–97
binary search trees (BSTree), 92–94
fast URL search, 104–105
search for text patterns using, 32–33
suffix array, 99
ternary search trees (TSTree), 100–102
sed
altering text using regular expression
replacement pattern, 36–37
exercise challenge, 197
overview of, 196
study drill, 197
study drills and further study, 37
SELECT
deleting data from SQL database, 176
reading (selecting) data, 168–169
SELECT (continued)
   SQL operation, 162
   updating complex data, 172–173
Self-criticism techniques, 16–17
Semantics, of grammars, 144
Servers
   creating, 202
   http.server module, 200–201, 202–203
set, binary search tree operations, 93
SET, SQL operation, 162
Setup requirements, for book exercises
   Python version 3.6, 6
   terminal, 6–7
   text editor, 6–8
sh command
   exercise challenge, 116–117
   implementing, 116
   study drills and further study, 117
shift/unshift
   comparing stacks and queues, 62
   inefficiencies of SingleLinkedList, 58
SingleLinkedList
   auditing, 55–56
   controller operations, 52
   exercise challenge, 55–56
   overview of, 50–51
   stack compared with, 64
   study drills, 56–57
   test operations, 53–55
skip function, of recursive parser, 138
SLY Parser Generator, 142, 157
Sockets, handling TCP/IP sockets, 202
Software, reuse, 196
sort
   exercise challenge, 38–39
   ordering text, 38
   removing duplicates from list, 40
   study drills and further study, 39
Sorting algorithms
   exercise challenge, 66–68
   overview of, 66
   study drills, 71–72
studying bubble sort, 68–69
studying merge sort, 69–70
studying quick sort, 71
Space characters, processing text with cut, 35
Speed, improving performance, 89
Spikes
   function of, 116
   implementing sh command, 116–117
   learning an API, 184
   working with command line arguments, 25
Spreadsheets, comparing SQL to Excel, 161
SQL
   administration, 180
   creating multi-table database, 164–165
   creating tables, 164
   deleting data, 176–177
   destroying/altering tables, 180
   exercise challenge for creating tables, 166–167
   exercise challenge for deleting data, 177–178
   exercise challenge for managing database, 182
   exercise challenge for selecting data, 169–170
   exercise challenge for updating data, 173–174
   further study, 163
   further study for creating tables, 167
   further study for deleting data, 178
   further study for making changes to database, 182
   further study for selecting data, 170
   further study for updating data, 174
   grammar constructs of, 163
   inserting data, 165–166
   learning SQL vocabulary, 162
   migrating and evolving data, 181–182
   overview of, 160–161
   reading (selecting) data, 168–169
   replacing data using DELETE/INSERT, 173
   setting up SQLite3, 161–162
   understanding tables, 158–159
   updating complex data, 172
SQL for Smarties (Celko), 187
SQL injection, not having in ORM, 187
SQLite3
  further study, 170, 174
  learning an API, 184–185
  setting up, 161–162
Stacks
  breaking data structures, 64
  comparing queues with, 63
  exercise challenge, 62–63
  further study, 64
  overview of, 62
Standard deviation
  improving accuracy of run chart, 39
  in troubleshooting problems, 40
Starting
  enemy of creativity, 20–21
  GO GO GO, 40
  in review of project strategy, 42
Statement-based languages, vs. expression-based, 149
States, organizing events as set of, 122–123
Strings, regular expressions and, 126
Study Drills, process of working with exercises in book, 3
subprocess module
  implementing find command, 29
  implementing sh command, 116–117
  xargs and, 108
Suffix array
  exercise challenge, 99
  fast URL search, 105
  overview of, 98–99
  study drills and further study, 99
Suffix tree, 98
Summarize command, CASMIR
  Dictionary code, 78–79
  overview of, 74
Symbols, regular expression, 126
sys.argv
  using find command, 29
  working with command line arguments, 24–25
T
  Tables
    comparing SQL to Excel, 161
    creating, 164
    creating multi-table database, 164–165
    destroying/altering, 180
    key to understanding SQL, 158–159
    reading (selecting) data across many,
      168–169
    removing rows, 176–177
  tail, applying 45-minute hacks, 44
TCP/IP
  further study, 203
  sockets, 202
Team Processes, 10
Template, for HTML pages, 190
Tension, focusing on work habits and behaviors, 35
Terminal
  implementing sh command, 116
  POSIX file redirection features, 26
  setup requirements for book exercises, 6–7
Ternary search trees (TSTree)
  creating tree from characters, 136–137
  exercise challenge, 100–102
  fast URL search, 105
  overview of, 100
  study drills, 102
  walking the tree, 144
Test-driven development (TDD)
  code coverage and, 121
  determining what your process is, 189
  implementing sh command, 116–117
  implementing tr command, 114–115
  study drills and further study, 119
Test first
  TDD development style, 114
  “test first” method, 94, 110
Tests
  code coverage and, 120–121
  for data structures, 47
  ed command, 196
Tests (continued)

improving performance, 89
invariants and, 60
pytest for Dictionary, 77–78
pytest for sorting algorithm, 68–71
SingleLinkedList, 53–55
Stack control class, 63
TDD development style, 114–117
"test first" method, 94, 110

Text

finding .txt files, 29
parsing. See Parsing text
processing with cut, 35
scanning for tokens, 130
searching for text patterns using regular expressions, 32–33
sorting, 38–39
t\textit{tr} tool for translating character streams, 114
viewing file contents when not in text format, 111

Text editors

setup requirements for book exercises, 6
Unix, 194–195
\textit{vi}, 198–199

Theme statement, determining what your process is, 189

Time management, 32
t\texttt{ime}it module, applying to bubble sort, 82–83

Timer

45-minute time limit, 40
benefits of setting time limit, 24
how to proactive creativity, 21

TODO lists. See also Checklists

process for early coders, 22–23
turning into TDD test, 116

Tokens

parsing, 136–137
patterns of text, 130

\texttt{tr} tool

exercise challenge, 114–115
study drills, 115
translating character streams, 114

Tracking, determining what your process is, 189

Trees

binary search. See Binary search trees (BSTree)
ternary search. See Ternary search trees (TSTree)
TSTree. See Ternary search trees (TSTree)

Tumblr, creating blog, 190

U

Unicode

processing text with \texttt{cut}, 35
library and, 128

uniq

eXercise challenge, 40–41
removing duplicates from list, 40
study drills and further study, 41

Unix

\texttt{bc} command, 154
shell operations, 27
text editor, 194

UPDATE

exercise administering database, 182
exercise challenge updating data, 173–174
further study updating data, 174
SQL operation, 161–162
updating complex data, 172
URLs, fast URL search, 104–105

V

Variables

creating simple calculator, 154–155
defining for \texttt{bc} language, 192
exercise working with command line arguments, 25
keeping track of variable definitions, 148

\texttt{vi}

eXercise challenge, 198–199
overview of, 198
study drills, 199

Vim, programmer's text editor, 6
virtualenv, setup requirements for book exercises, 7
Visitor pattern
  benefits of, 144–145
  how to write interpreters, 152
Vulnerabilities, OWASP Top 10, 200–201

W
Web servers
  attacking (breaking), 200–201
  creating from scratch, 202–203
  creating using http.server module, 201
  exercise challenge, 200, 202
  study drills and further study, 201, 203
WFM (Works For Me), learning an API, 184
WHERE, SQL operation, 162
Whitespace, regular expression for, 130
Wordpress, creating blog, 190

Work area, making improvements to, 28–29
Work habits, creating healthily habits, 34–35
Works For Me (WFM), learning an API, 184
Wraps, wrapping existing data structure vs. creating new, 64

X
xargs
  exercise challenge, 108
  overview of, 108
  study drills, 108–109

Y
yacc tool, researching, 193
yes, applying 45-minute hacks, 44

Z
zsh, running programs from Terminal, 116