

# Foundations of Agile Python Development



Jeff Younker

## **Foundations of Agile Python Development**

**Copyright © 2008 by Jeff Younker**

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-59059-981-5

ISBN-10 (pbk): 1-59059-981-0

ISBN-13 (electronic): 978-1-4302-0636-1

ISBN-10 (electronic): 1-4302-0636-5

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Tom Welsh

Technical Reviewer: Will McGugan

Editorial Board: Clay Andres, Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell,

Jonathan Gennick, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Frank Pohlmann,

Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Project Manager: Susannah Davidson Pfalzer

Copy Editor: Damon Larson

Associate Production Director: Kari Brooks-Copony

Production Editor: Elizabeth Berry

Compositor: Dina Quan

Proofreaders: Nancy Bell, April Eddy

Indexer: John Collin

Artist: Kinetic Publishing Services, LLC

Cover Designer: Kurt Krames

Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail [orders-ny@springer-sbm.com](mailto:orders-ny@springer-sbm.com), or visit <http://www.springeronline.com>.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail [info@apress.com](mailto:info@apress.com), or visit <http://www.apress.com>.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at <http://www.apress.com/info/bulksales>.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at <http://www.apress.com>.

# Contents at a Glance

About the Author .....	xiii
About the Technical Reviewer.....	xv
Acknowledgments .....	xvii
Introduction.....	xix
■ <b>CHAPTER 1</b> What Is Agile Development? .....	1
■ <b>CHAPTER 2</b> The IDE: Eclipsing the Command Line .....	21
■ <b>CHAPTER 3</b> Revision Control: Subverting Your Code .....	41
■ <b>CHAPTER 4</b> Setuptools: Harnessing Your Code .....	81
■ <b>CHAPTER 5</b> A Build for Every Check-In .....	103
■ <b>CHAPTER 6</b> Testing: The Horse and the Cart.....	139
■ <b>CHAPTER 7</b> Test-Driven Development and Impostors .....	175
■ <b>CHAPTER 8</b> Everybody Needs Feedback.....	233
■ <b>CHAPTER 9</b> Databases.....	263
■ <b>CHAPTER 10</b> Web Testing .....	309
■ <b>CHAPTER 11</b> Functional Testing.....	339
■ <b>INDEX</b> .....	369



# Contents

About the Author .....	xiii
About the Technical Reviewer.....	xv
Acknowledgments .....	xvii
Introduction.....	xix
<b>■ CHAPTER 1    What Is Agile Development? .....</b>	<b>1</b>
Why More Methodologies? .....	1
A Little History .....	3
Planning and Agile Development .....	4
What Are Agile Methods?.....	4
Pair Programming .....	5
User Stories .....	7
The System Metaphor.....	8
On-Site Customers .....	8
Unit Tests .....	9
Test-Driven Development.....	10
Refactoring .....	11
Simple Design .....	12
Collective Code Ownership .....	12
Short Iterations.....	13
Continuous Reflection .....	15
Continuous Integration .....	16
Documentation.....	17
Summary.....	18
<b>■ CHAPTER 2    The IDE: Eclipse the Command Line .....</b>	<b>21</b>
Installing Eclipse .....	23
Installing Plug-Ins .....	25
Installing and Configuring Pydev.....	31
Your First Project.....	32
Looking Under the Hood.....	38
Paying for More Functionality .....	39
Summary .....	40

<b>CHAPTER 3</b>	<b>Revision Control: Subverting Your Code</b>	41
	Revision Control Phylum	42
	What Subversion Does for You	43
	Getting Subverted	44
	Working with Your Subverted Code	47
	Examining Files	49
	Adding Files	50
	Copying and Moving Files	51
	Deleting Files	52
	Reverting Changes	53
	Modifying a File	53
	Updating Your Working Copy	54
	Conflicting Changes	55
	Subverting Eclipse	59
	Sharing Your Subverted Project	59
	Importing from Subversion	60
	Working with a Subverted Eclipse	64
	The Team Repository View	65
	Adding a File	68
	Committing Changes	70
	Editing a File	71
	Reverting Changes	72
	Resolving Conflicts	73
	Deleting Files	76
	Moving Files	77
	Renaming Files	77
	Copying Files	78
	Reverting Moves, Renames, and Copies	79
	Summary	79
<b>CHAPTER 4</b>	<b>Setuptools: Harnessing Your Code</b>	81
	The Project: A Simple RSS Reader	81
	Python Modules	82
	The Old Way	83
	The New Way: Cooking with Eggs	84
	Some Notes About Building Multiple Versions	85
	Installing Setuptools	86
	Getting Started with Setuptools	87
	Building the Project	88

Installing Executables . . . . .	91
Dependencies . . . . .	92
Think Globally, Install Locally . . . . .	94
Removing an Existing Package: Undoing Your Hard Work . . . . .	95
Installing from the Local Copy . . . . .	96
Fixing Options with setup.cfg . . . . .	97
Bootstrapping Setuptools . . . . .	97
Subverting Subversion: What Shouldn't Be Versioned . . . . .	98
The Easy Way with Eclipse . . . . .	100
Checking in Changes: Not Losing It . . . . .	100
Working in Development Mode . . . . .	100
Summary . . . . .	102
 <b>CHAPTER 5   A Build for Every Check-In . . . . .</b>	 103
Buildbot Architecture . . . . .	104
Installing Buildbot . . . . .	104
Configuring the Build System . . . . .	106
Mastering Buildbot . . . . .	107
Enslaving Buildbot . . . . .	112
Hooking Up Source Control . . . . .	116
Using the Source . . . . .	119
Subversion to Buildbot, Over . . . . .	121
A Python for Every Builder . . . . .	122
Finally, a Real Build Succeeds . . . . .	124
Installing the Build . . . . .	125
Supporting Python 2.4 Builds . . . . .	128
Ensuring Local Dependency Processing . . . . .	132
Keeping Up Appearances . . . . .	134
Summary . . . . .	136
 <b>CHAPTER 6   Testing: The Horse and the Cart . . . . .</b>	 139
Unit Testing . . . . .	141
The Problems with Not Unit Testing . . . . .	142
Pessimism . . . . .	143
Test-Driven Development . . . . .	146
Knowing Your Unit Tests . . . . .	147
unittest and Nose . . . . .	148
A Simple RSS Reader . . . . .	149
The First Tests . . . . .	151

Finding Tests with Nose .....	159
Skipping Slow Tests .....	160
Integrating the Tests into the Environment .....	162
Running Tests After Every Change .....	163
Running the Complete Test Suite in Development .....	167
Buildbot with Unit Tests .....	171
Summary .....	173

## ■ CHAPTER 7    **Test-Driven Development and Impostors** ..... 175

Moving Beyond Acceptance Tests .....	175
Renaming .....	183
Overriding Existing Methods: Monkeypatching .....	185
Monkeypatching and Imports .....	186
The Changes Go Live .....	188
Using Data Files .....	189
Isolation .....	190
Rolling Your Own .....	192
Python Quirks .....	193
Mocking Libraries .....	193
Aggregating Two Feeds .....	194
A Simple pMock Example .....	195
Implementing with pMock .....	196
Test: Defining combine_feeds .....	196
Test: Defining add_single_feed .....	197
Refactoring: Extracting AggregateFeed .....	198
Refactoring: Moving add_single_feed .....	199
Test: Defining create_entry .....	200
Test: Ensuring That AggregateFeed Creates a FeedEntry Factory .....	200
Test: Defining add .....	201
Test: AggregateFeed.entries Is Always Initialized to a Set .....	201
Test: Defining FeedEntry.from_parsed_feed .....	202
Test: Defining feed_entry_listing .....	202
Test: Defining feeds_from_urls .....	203
Test: AggregateFeed Initializes the FeedParser Factory .....	203
Test: Defining from_urls .....	204
Refactoring: Reimplementing from_urls .....	204
Refactoring: Condensing Some Tests .....	206



Test: Formatting Feed Entry Listings . . . . .	207
Test: Defining print_entry_listings . . . . .	208
Test: FeedWriter Initializes the stdout Attribute . . . . .	209
Test: Empty AggregateFeeds Generate No Output. . . . .	209
Test: Defining is_empty . . . . .	210
Test: Defining new_main . . . . .	210
Test: The Application Initializes Dependencies. . . . .	211
Refactoring: Making new_main the New main2 . . . . .	212
A Simple PyMock Example . . . . .	212
Monkeypatching. . . . .	214
Saying the Same Thing Differently . . . . .	214
Implementing with PyMock . . . . .	215
Test: from_urls and Mocking External Modules. . . . .	216
Test: Defining add_single_feed . . . . .	217
Refactoring: Moving Methods to a New Object . . . . .	218
Refactoring: Moving add_single_feed . . . . .	218
Refactoring: Moving from_urls() . . . . .	219
Test: create_entry() and Mocking Class Constructors. . . . .	220
Tests: Defining add and AggregateFeed.__init__ . . . . .	221
Test: Defining FeedEntry.__init__ . . . . .	222
Test: Defining listing . . . . .	222
Test: entry_listings Should Be Sorted . . . . .	223
Test: Defining print_entry_listings . . . . .	224
Test: print_entry_listings Should Do Nothing with Empty Feeds . . . . .	225
Test: is_empty and the Unproven Test . . . . .	226
Test: new_main, Hooking It All Together. . . . .	226
Test: RSReader Initialization . . . . .	227
Finishing Up: Activating the New Functionality . . . . .	227
Other pMock and PyMock Features . . . . .	228
Raising Exceptions with pMock. . . . .	228
Raising Exceptions with PyMock. . . . .	228
Playback Counts with pMock. . . . .	229
Playback Counts with PyMock. . . . .	229
Mocking Attribute Setters with PyMock. . . . .	229
Mocking Generators with PyMock . . . . .	230
Using PyMock with unittest . . . . .	230
Summary . . . . .	231

<b>CHAPTER 8</b>	<b>Everybody Needs Feedback</b>	233
	Measuring Software Quality	235
	Measurements	236
	Quantitative Measurements: How Much Is That Doggie in the Window?	237
	Code Coverage	237
	Complexity Measurements	239
	Velocity: When Are We Done?	242
	Qualitative Measurements: It's a Shih Tzu!	243
	Coding Conventions	244
	Welcome Back to Python	246
	Never Try to Fix a Social Problem with a Technical Solution	248
	Code Reviews	249
	Renaming	250
	Communication	250
	Technological Feedback: Bad Programmer, No Cookie	251
	Coercion at the Keyboard	251
	When Code Is Submitted	256
	Buildbot and Coverage	258
	Summary	261
 <b>CHAPTER 9</b>	 <b>Databases</b>	 263
	A New Religion	263
	Blurring the Boundaries	264
	Concealing Data Access	265
	Object-Relational Mappers	265
	The Active Record Pattern	266
	The Data Mapper Pattern	266
	The Unit of Work Pattern	266
	Python ORMs	267
	SQLObject	267
	SQLAlchemy	283
	Building the Database	296
	Testing	297
	Refactorings	298

Migrations . . . . .	298
The Instructions . . . . .	299
Numbering Migrations and Playing Them Back . . . . .	299
Where to Put the Migration Mechanism . . . . .	300
DBMigrate: A Migration Mechanism . . . . .	300
Summary . . . . .	306
 <b>CHAPTER 10 Web Testing . . . . .</b>	 309
Really Simple Primer . . . . .	309
HTML . . . . .	310
CSS . . . . .	311
XML . . . . .	311
URI and URL . . . . .	311
HTTP . . . . .	312
JavaScript . . . . .	312
Web Servers and Web Applications . . . . .	312
WSGI . . . . .	314
Using the write Callback . . . . .	315
WSGI Middleware . . . . .	316
Testing Web Applications . . . . .	316
Graphics and Images . . . . .	317
Markup . . . . .	317
Testing JavaScript . . . . .	320
Using JsUnit . . . . .	321
Running a Test . . . . .	322
How It Works . . . . .	326
Connoisseur of the Undefined . . . . .	327
Adding a Little More Realism . . . . .	328
Manipulating the DOM . . . . .	328
Aggregating Tests . . . . .	335
Running Tests by URL . . . . .	336
Summary . . . . .	337

- CHAPTER 11 Functional Testing . . . . . 339
  - Running Acceptance Tests . . . . . 339
  - PyFit. . . . . 340
    - Writing Requirements . . . . . 341
    - A Simple PyFit Example . . . . . 344
    - Giving the Acceptance Tests a Home. . . . . 346
    - Your First FIT. . . . . 346
  - FIT into Buildbot. . . . . 353
    - Preparing the Slave . . . . . 353
    - Run New Builder, Run! . . . . . 354
    - Making the Reports Available . . . . . 358
    - Getting Regular Builds . . . . . 366
    - What’s Left? . . . . . 367
  - Summary . . . . . 367
- INDEX . . . . . 369

# About the Author



**JEFF YOUNKER** is chief engineer of Data-Pipes ([www.data-pipes.com/](http://www.data-pipes.com/)). His educational background carefully avoided computers, but he was drawn in anyway. Most of his misguided adulthood has been spent in large installation systems administration, tool smithing, and release engineering, with a peculiar obsession involving both monitoring and rapid deployment. Over the last several years, he's had the pleasure of working with Python full time. Having escaped Texas nearly a decade ago, he now lives in gray and rainy Northern California. When not suffering monitor-induced radiation burns, Jeff likes to do anything that doesn't involve a roof, unless the roof has been top-roped or covers a machine shop.



# About the Technical Reviewer



**WILL McGUGAN** is a software developer and author currently working in London on a social networking site for games built with Django. See his blog at [www.willmcgugan.com/](http://www.willmcgugan.com/) for more information on Will's work and open source projects.





# Acknowledgments

I'd like to thank Apress and Jason Gilmore for giving me the opportunity to write this book. Thanks to Tom Welsh for the unfaltering criticism that has made it something more than my incoherent ravings. I thank Susannah Davidson Pfalzer for playing midwife to Jason's child after he left the Apress family. This has been a huge undertaking. If I had really understood the magnitude, I might not have started to begin with, but Susannah kept everything on track and graciously coped with the events in my life impinging on the schedule.

Thanks to Will McGugan for the many improvements he made in the code, and for his meticulous attention to detail. Damon Larson did an amazing job turning my technobabble into coherent English. It was wonderful working with Liz Berry in the last few weeks of madness as the disjointed word-processed files turned into something that looks suspiciously like a book.

A whole slew of friends came out of the woodwork as they discovered that I was writing this book, and I was astonished to discover that they actually wanted to help. The last half of this book is much richer for the efforts of Matt Ho, Jacob Hoffman-Andrews, Nancy Huntingford, Rachel McConnell, and Erik Ziko.

In particular, Matt Ho's assistance with Chapter 11 was indispensable. I don't know if I could have finished this book on time without the days we spent in his living room. At a point where I was completely lost, his advice made the path clear. I wish we'd talked much earlier in the process.

My business partner David Birkhead bought time with our customers to give me an opportunity to finish this project, and I owe him a huge debt for that. He's been hugely supportive outside of our business relationship, too. I have to thank our customers Cynthia Walston and David Alban for being so supportive and accommodating during the last month of this process.

I owe a huge debt of gratitude to my father, William James Younker, who passed away somewhere around Chapter 5. No matter how poor my family was, he always saw to it that I had the tools to follow my curiosity. I wish I could show him this book you're reading. At least I'll get to show it to my mother, Hallie Younker, who did an astoundingly good job of bringing me up given the circumstances that enveloped our lives.

I'd like to thank my teachers, too—I had some amazing ones. I'd like to thank Adréa Shaw for giving me such wonderful anecdotes about Baylor College of Medicine. I remember the soft glow of the electron microscope fondly. I'd like to thank David Raikow. He writes more than anyone I know, and his advice kept me moving along when I got stuck. I'd like to give my thanks to Ethyl the Dog, too. She was remarkably good at alleviating writer's block.

I've received support in other ways, as well. Roxanne Williams and Sören Ragsdale lent me technical assistance when necessary. Blair Miller kept me from going insane and from time to time made sure that I stayed fed. Scott Calvert, Gwen Perry, and David Cutler made sure that I wasn't completely isolated from my friends. Kathryn Keslosky's assistance with revisions was immensely helpful.

My dear friends Jaron and Cherry Rothkop made sure that my time in Toledo was bearable. Clare and John Vrakking put up with my mental absence from our first Christmas together in over a decade. I'm sorry this book didn't let me make Passover this year. Finally, I don't know if I could have completed this book without Amy Woodward's love and support. Her calming voice and compassionate words kept me from coming unglued when I became immobilized by the seemingly unending tide of drafts and revisions. She assisted me emotionally and materially, and she believed in me more than I believed in myself at times. I owe her a debt of gratitude that I don't know if I can ever repay. A thousand love songs do her no justice.

# Introduction

If you're embarking on a Python development project, then you should buy this book—there's nothing quite like it. I know this because I was looking for it last year, and I couldn't find it. This book introduces the tools you'll need to get started on agile projects in Python, and unlike any other book out there, it shows you how to tie them all together.

Sure, there are many good books on agile development. A lot of them cover the development processes in great detail, and this is a good thing. Agile development is very much about human interactions and the environment surrounding software development, but there is a whole ecology of tooling to make everything work at a practical level.

Agile development eschews extensive up-front specification, and it anticipates that the product will constantly change, but it puts in place rigorous checks to compensate for anticipated change. Testing is an integral part of agile development from the very start, and it is pursued with ferocious rigor. You need software tools to facilitate testing.

Agile projects have very short release cycles, and this has implications for tooling, too. There's no way to have two-week release cycles if it takes you days to integrate changes, days to perform QA, and days to package and deploy the software. This means that agile development puts a high value on build and release automation.

While agile development techniques can be applied to any project, both testing tools and build automation tend to be very language specific. These tools do exist in Python. They're widely available, and by and large they're free, too, but the documentation tends to be . . . um . . . spotty. And while there may be documentation on the individual tools, the documentation telling you how to tie these tools together is usually sparse to nonexistent. This book provides that missing documentation.

## Who This Book Is For

This book is written for a person who knows how to program and is already familiar with Python. If you have some Python under your belt and you're thinking of starting a new project, but you don't know how to get started, then this book is for you. If you're an experienced Python programmer and you want to give this agile stuff a whirl, then this book is for you. If you're a release engineer who has been thrown headlong into the world of Python, then this book is for you, too. If you're brand new to programming or don't really know Python, this is not the best book to start with. There are some wonderful books out there that will introduce you to the language, but this isn't one of them.

# What's Really in Here?

Each chapter in this book addresses a different aspect of tooling in an agile development environment. These are collected roughly into two parts, with the first focusing on basic tooling, and the second focusing on specific practices. If you're already familiar with Subversion, Setuptools, and Buildbot, then you should have no problem jumping between Chapters 6 through 11. If you're not, then you'll want to look at the earlier chapters first.

## Chapter 1: What Is Agile Development?

Chapter 1 provides an overview of the methods that characterize agile development methodologies, with a focus on those not directly related to tooling.

## Chapter 2: The IDE: Eclipsing the Command Line

This book uses the command line throughout, but modern IDEs provide many benefits. This chapter introduces you to Python development using Eclipse and the Pydev plug-in.

## Chapter 3: Revision Control: Subverting Your Code

A revision control system is part of the core infrastructure for any agile development environment. Subversion is an excellent choice. I show you how to use it from the command line and from Eclipse using the Subversive plug-in.

## Chapter 4: Setuptools: Harnessing Your Code

You can't replicate your work for testing purposes without some sort of a framework. In Python, a natural choice is Setuptools, which provides a solid basis for automated builds.

## Chapter 5: A Build for Every Check-In

Automated build systems form the core of a continuous integration system. Here I introduce Buildbot, an excellent system that happens to be written in Python. It ensures that the code you check in builds correctly.

## Chapter 6: Testing: The Horse and the Cart

Unit testing ensures that your code runs as you expect it to, and it prevents regression (reappearance of old bugs) when you change existing code. I introduce the unit-testing packages unittest and Nose, and I show how to use Nose to run tests from within Eclipse and Setuptools. Finally, I show how to link them into Buildbot.

## Chapter 7: Test-Driven Development and Impostors

Test-driven development (TDD) is the practice of writing tests before writing the code they test. Imposters (a.k.a. mock objects) provide a powerful unit-testing technique to isolate units of code. I examine two mock object frameworks, pMock and PyMock, and I work through a sizable example to show how TDD, refactoring, and imposters are used, and how they affect the code that you produce with them.

## Chapter 8: Everybody Needs Feedback

Improving your code requires feedback—useful information that sometimes comes from your coworkers, and sometimes from software. Accurate feedback requires standards. This chapter looks at code coverage, complexity measures, and development velocity. It also examines coding standards, how they can be enforced from within Eclipse, and how you can prevent bad code from reaching your repository by using Subversion pre-commit hooks.

## Chapter 9: Databases

Databases are very widely used these days, and they pose their own special challenges for agile development. This chapter examines the object-relational mappers SQLAlchemy and SQLAlchemy, and then examines how to version databases using the DBMigrate tool.

## Chapter 10: Web Testing

The web is everywhere, and web development has its own set of issues. This chapter examines general approaches to testing web applications, and introduces HTML/XML verification using ElementTree and BeautifulSoup. It also looks into JavaScript unit testing with JsUnit.

## Chapter 11: Functional Testing

This chapter examines functional testing with a particular emphasis on acceptance testing using PyFit. The chapter shows how to use PyFit, and more importantly, how to tie PyFit into Setuptools and Buildbot. (In my view, this alone is worth the price of the book.)

## Contacting Me

Finally, please don't hesitate to give me feedback on the book at any time. This is my first book, my writing ability has improved immensely as the book has progressed, and I now have a much better understanding of what I wanted to say than when I started. I'll try to improve any sections that people find lacking and publish them to this book's web page at <http://www.apress.com/book/view/9781590599815>. Additional materials may be available on my blog ([www.theblobshop.com/blog](http://www.theblobshop.com/blog)) under the tag famip. I'll present more information in these locations as it becomes available. This pertains but is not limited to notes about anything that I've fouled up, new thoughts, and additional materials that I think you may find useful.

