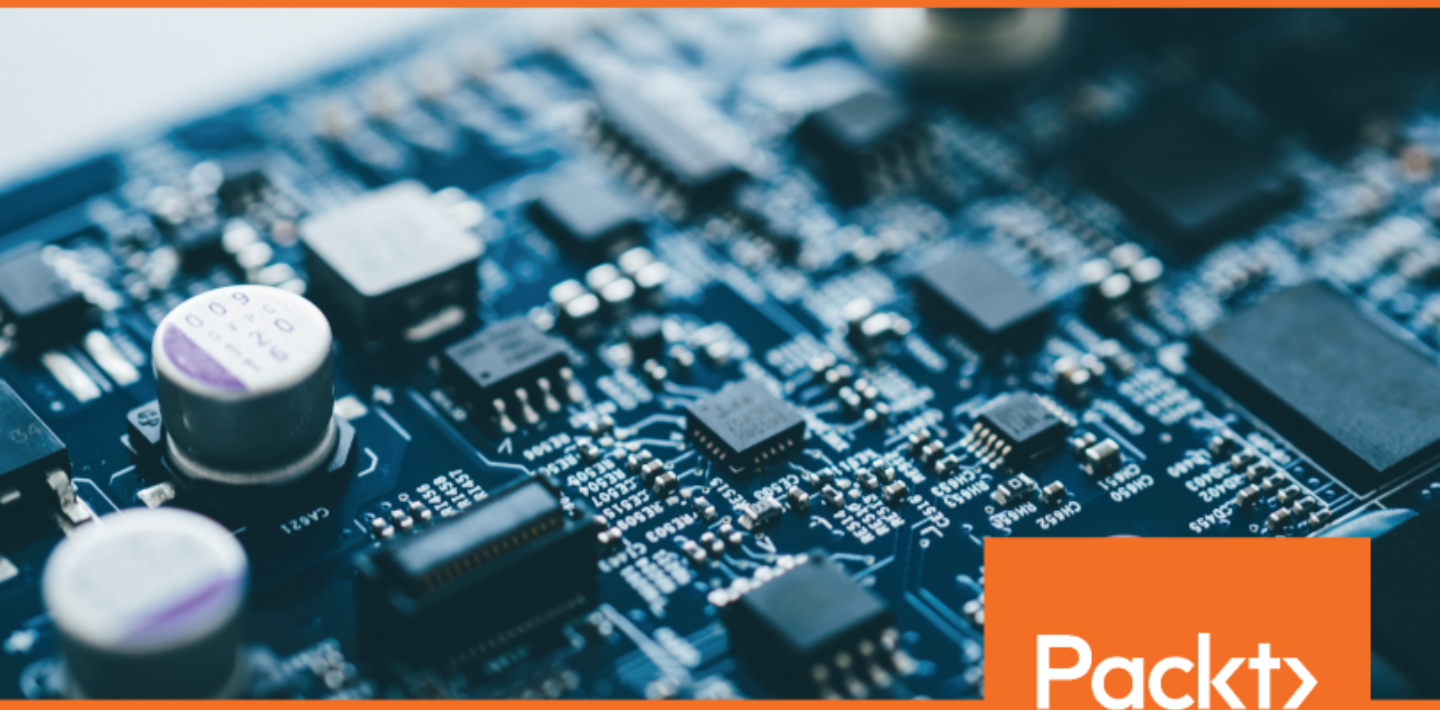
Clean Code in Python

Refactor your legacy code base



Packt>

www.packt.com

By Mariano Anaya

Clean Code in Python

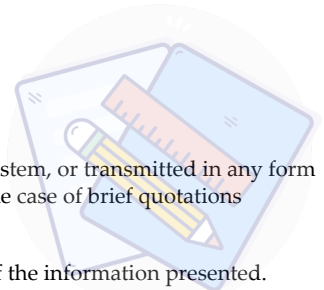
Refactor your legacy code base

Mariano Anaya

Packt

BIRMINGHAM - MUMBAI

Clean Code in Python



Commissioning Editor: Merint Mathew
Acquisition Editor: Denim Pinto
Content Development Editor: Priyanka Sawant
Technical Editor: Gaurav Gala
Copy Editor: Safis Editing
Project Coordinator: Vaidehi Sawant
Proofreader: Safis Editing
Indexer: Rekha Nair
Graphics: Jason Monteiro
Production Coordinator: Shantanu Zagade

First published: August 2018

Production reference: 1270818

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78883-583-1

www.packtpub.com

To my family and friends, for their unconditional love and support.

– Mariano Anaya

mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

PacktPub.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Contributors

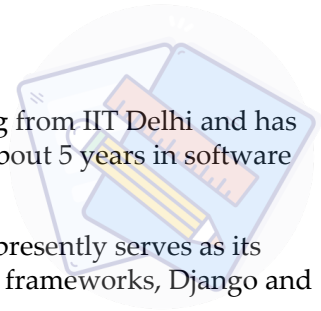
About the author

Mariano Anaya is a software engineer who spends most of his time creating software with Python and mentoring fellow programmers. Mariano's main areas of interests besides Python are software architecture, functional programming, distributed systems, and speaking at conferences.

He was a speaker at Euro Python 2016 and 2017. To know more about him, you can refer to his GitHub account with the username rmariano.

His speakerdeck username is rmariano.

About the reviewer



Nimesh Kiran Verma has a dual degree in Maths and Computing from IIT Delhi and has worked with companies such as LinkedIn, Paytm and ICICI for about 5 years in software development and data science.

He co-founded a micro-lending company, Upwards Fintech and presently serves as its CTO. He loves coding and has mastery in Python and its popular frameworks, Django and Flask.

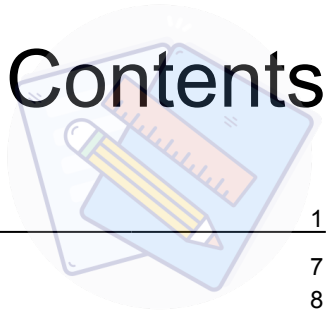
He extensively leverages Amazon Web Services, design patterns, SQL and NoSQL databases, to build reliable, scalable and low latency architectures.

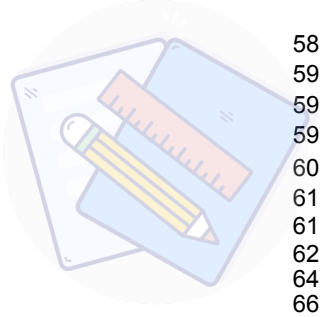
> *To my mom, Nutan Kiran Verma, who made me what I am today and gave the confidence to pursue all my dreams. Thanks Papa, Naveen, and Prabhat, who motivated me to steal time for this book when in fact I was supposed to be spending it with them. Ulhas and the entire Packt team's support was tremendous. Thanks Varsha Shetty for introducing me to Packt.*

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packtpub.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents





Preface

This is a book about software engineering principles applied to Python.

There are many books about software engineering, and many resources available with information about Python. The intersection of those two sets, though, is something that requires action, and that's the gap this book tries to bridge.

It would not be realistic to cover all possible topics about software engineering in a single book because the field is so wide that there are entire books dedicated to certain topics. This book focuses on the main practices or principles of software engineering that will help us write more maintainable code, and how to write it by taking advantage of the features of Python at the same time.

A word to the wise: there is no single solution to a software problem. It's usually about trade-offs. Each solution will have upsides and downsides, and some criteria must be followed to choose between them, accepting the costs and getting the benefits. There is usually no single best solution, but there are principles to be followed, and as long as we follow them we will be walking a much safer path. And that is what this book is about: inspiring the readers to follow principles and make the best choices, because even when facing difficulties, we will be much better off if we have followed good practices.

And, speaking of good practices, while some of the explanations follow established and proven principles, other parts are opinionated. But that doesn't mean it has to be done in that particular way only. The author does not claim to be any sort of authority on the matter of clean code, because such a title cannot possible exist. The reader is encouraged to engage in critical thinking: take what works the best for your project, and feel free to disagree. Differences of opinions are encouraged as long as they yield an enlightening debate.

My intention behind this book is to share the joys of Python, and idioms I have learned from experience, in the hope that readers will find them useful to elevate their expertise with the language.

The book explains the topics through code examples. These examples assume the latest version of Python at the time of this writing is used, namely Python 3.7, although future versions should be compatible as well. There are no peculiarities in the code that bind it to any particular platform, therefore with a Python interpreter, the code examples can be tested on any operating system.

In most of the examples, with the goal of keeping the code as simple as possible, the implementations and their tests are written in plain Python using just the standard libraries. In some chapters, extra libraries were needed, and in order to run the examples of those cases, instructions have been provided along with the respective `requirements.txt` file.

Throughout this book we will discover all the features Python has to offer to make our code better, more readable, and easier to maintain. We do so not only by exploring the features of the language, but also by analyzing how software engineering practices can be applied in Python. The reader will notice that some of the reference implementations differ in Python, other principles or patterns change slightly, and others might not be even applicable all along. Understanding each case represents an opportunity to understand Python more deeply.

Who this book is for

This book is suitable for all software engineering practitioners who are interested in software design or learning more about Python. It is assumed that the reader is already familiar with the principles of object-oriented software design and has some experience writing code.

In terms of Python, the book is suitable for all levels. It's good for learning Python because it is organized in such a way that the content is in increasing order of complexity. The first chapters will cover the basics of Python, which is a good way to learn the main idioms, functions, and utilities available in the language. The idea is not just to solve some problems with Python, but to do so in an idiomatic way.

Experienced programmers will also benefit from the topics in this book, as some sections cover advanced topics in Python, such as decorators, descriptors, and an introduction to asynchronous programming. It will help the reader discover more about Python because some of the cases are analyzed from the internals of the language itself.

It is worth emphasizing the word *practitioners* in the first sentence of this section. This is a pragmatic book. Examples are limited to what the case of study requires, but are also intended to resemble the context of a real software project. It is not an academic book, and as such the definitions made, the remarks made, and the recommendations given are to be taken with caution. The reader is expected to examine these recommendations critically and pragmatically rather than dogmatically. After all, practicality beats purity.