Extracted from:

Pythonic Programming

Tips for Becoming an Idiomatic Python Programmer

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Pythonic Programming

Tips for Becoming an Idiomatic Python Programmer

Dmitry Zinoviev

Edited by Adaobi Obi Tulton

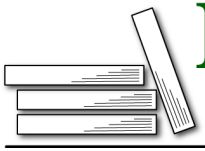
Pythonic Programming

Tips for Becoming an Idiomatic Python Programmer

Dmitry Zinoviev

The Pragmatic Bookshelf

Raleigh, North Carolina



For our complete catalog of hands-on, practical, and Pragmatic content for software developers, please visit *https://pragprog.com*.

The team that produced this book includes:

CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Adaobi Obi Tulton
Copy Editor: Karen Galle
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics
Founders: Andy Hunt and Dave Thomas

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-861-1
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—October 2021

Introduction

Python is a fantastic programming language. It is concise. It is flexible. It is versatile. It is elegant. It is unbelievably popular, firmly holding its position as the number three language in the world since September 2018, according to TIOBE Index.[1] It comes with a great collection of about 200 modules in the standard library, an unmeasurable pile of third-party modules, and a well-documented extension mechanism. Finally, Python is very efficient, despite being an interpreted language. You just have to follow its spirit.

Every programming language and system has its spirit. The spirit of FORTRAN 66/77 is bulky multidimensional arrays of real numbers, uppercase letters, and a lack of recursion. The spirit of C is pointers and the happy sisters malloc() and free(). The spirit of Java is pages-long classes and the Java virtual machine. What is the spirit of Python, then?

This book offers almost one hundred tips that explain how to write Pythonic code in the namesake language. It is hard to explain what "Pythonic" means. Just like Zen that I mention in Tip 2, Import This, on page ? and elsewhere in the book, "pythonicity" (yes, there is such word!) is an epiphany, an enlightenment that is not learned but experienced. Hopefully, after browsing or reading the book, you will become a more Pythonic programmer—and, therefore, a better Python programmer in general.

The tips are grouped into six chapters: documentation tips, data types/data structures tips, safety tips, performance tips, function design tips, and general tips. They embrace different aspects of pythonicity: how to make your programs correct, safe, fast, easy to read, and easy to maintain.

About the Software

All you need is Python. Almost any Python suffices. Ninety-five percent of the tips are compatible with any currently supported version (3.4 and above) and

1. www.tiobe.com/tiobe-index/

with 2.7, which is not supported anymore but is still used in legacy software. Five percent of the tips work only for Python 3.4 and above. And there is only one exception: Tip 64, Format with Formatted Strings, on page ?—that requires Python 3.6.

About the Notation

Each tip in the book comes with a brief "stars-and-numbers" annotation that describes its complexity and compatibility. The number of stars ranges from one ★ (a simple, almost trivial tip) to three ★★★ (an advanced tip). Naturally, the star ratings are subjective, but rest assured that any one-star tip is much simpler than any three-star tip. Most of the tips belong to the middle category.

The number or numbers in the exponent are the Python versions that are compatible with the tip. Most tips work for any Python at or above 2.7, but some require a more recent version. (And if you still have an installation of Python 2.6 that was officially retired in October 2013, you must seriously ask yourself why.)

The most common combination of stars and numbers is ★★$^{2.7,\ 3.4+}$: an intermediate-to-advanced tip that works for any popular version of Python.

About the Reader

The book is primarily for programmers who are already somewhat familiar with the language. You may be a first-year computer science or engineering undergraduate student; a student or researcher in another field, trying to learn programming skills; a seasoned programmer switching to Python from Fortran or C/C++/Java; or merely adding Python to your toolset.

Is that you? Enjoy the book! Are you someone else? Enjoy it all the same!