Afterword

> *Python is a language for consenting adults.*
>
> —Alan Runyan, cofounder of Plone

Alan's pithy definition expresses one of the best qualities of Python: it gets out of the way and lets you do what you must. This also means it doesn't give you tools to restrict what others can do with your code and the objects it builds.

At age 30, Python is still growing in popularity. But of course, it is not perfect. Among the top irritants to me is the inconsistent use of `CamelCase`, `snake_case`, and `joinedwords` in the standard library. But the language definition and the standard library are only part of an ecosystem. The community of users and contributors is the best part of the Python ecosystem.

Here is one example of the community at its best: while writing about *asyncio* in the first edition, I was frustrated because the API has many functions, dozens of which are coroutines, and you had to call the coroutines with `yield from`—now with `await`—but you can't do that with regular functions. This was documented in the *asyncio* pages, but sometimes you had to read a few paragraphs to find out whether a particular function was a coroutine. So I sent a message to python-tulip titled "Proposal: make coroutines stand out in the *asyncio* docs". Victor Stinner, an *asyncio* core developer; Andrew Svetlov, main author of *aiohttp*; Ben Darnell, lead developer of Tornado; and Glyph Lefkowitz, inventor of *Twisted*, joined the conversation. Darnell suggested a solution, Alexander Shorin explained how to implement it in Sphinx, and Stinner added the necessary configuration and markup. Less than 12 hours after I raised the issue, the entire *asyncio* documentation set online was updated with the *coroutine* tags you can see today.

That story did not happen in an exclusive club. Anybody can join the python-tulip list, and I had posted only a few times when I wrote the proposal. The story illustrates a community that is really open to new ideas and new members. Guido van Rossum used to hang out in python-tulip and often answered basic questions.

Another example of openness: the Python Software Foundation (PSF) has been working to increase diversity in the Python community. Some encouraging results are already in. The 2013–2014 PSF board saw the first women elected directors: Jessica McKellar and Lynn Root. In 2015, Diana Clarke chaired PyCon North America in Montréal, where about one-third of the speakers were women. PyLadies became a truly global movement, and I am proud that we have so many PyLadies chapters in Brazil.

If you are a Pythonista but you have not engaged with the community, I encourage you to do so. Seek the PyLadies or Python Users Group (PUG) in your area. If there isn't one, create it. Python is everywhere, so you will not be alone. Travel to events if you can. Join live events too. During the Covid-19 pandemic I learned a lot in the "hallway tracks" of online conferences. Come to a PythonBrasil conference—we've had international speakers regularly for many years now. Hanging out with fellow Pythonistas brings real benefits besides all the knowledge sharing. Like real jobs and real friendships.

I know I could not have written this book without the help of many friends I made over the years in the Python community.

My father, Jairo Ramalho, used to say "Só erra quem trabalha," Portuguese for "Only those who work make mistakes," great advice to avoid being paralyzed by the fear of making errors. I certainly made my share of mistakes while writing this book. The reviewers, editors, and early release readers caught many of them. Within hours of the first edition early release, a reader was reporting typos in the errata page for the book. Other readers contributed more reports, and friends contacted me directly to offer suggestions and corrections. The O'Reilly copyeditors will catch other errors during the production process, which will start as soon

as I manage to stop writing. I take responsibility and apologize for any errors and suboptimal prose that remains.

I am very happy to bring this second edition to conclusion, mistakes and all, and I am very grateful to everybody who helped along the way.

I hope to see you soon at some live event. Please come say hi if you see me around!

Further Reading

I will wrap up the book with references regarding what it its to be "Pythonic"—the main question this book tried to address.

Brandon Rhodes is an awesome Python teacher, and his talk "A Python Æsthetic: Beauty and Why I Python" is beautiful, starting with the use of Unicode U+00C6 (`LATIN CAPITAL LETTER AE`) in the title. Another awesome teacher, Raymond Hettinger, spoke of beauty in Python at PyCon US 2013: "Transforming Code into Beautiful, Idiomatic Python".

The "Evolution of Style Guides" thread that Ian Lee started on Python-ideas is worth reading. Lee is the maintainer of the `pep8` package that checks Python source code for PEP 8 compliance. To check the code in this book, I used `flake8`, which wraps `pep8`, `pyflakes`, and Ned Batchelder's McCabe complexity plug-in.

Besides PEP 8, other influential style guides are the *Google Python Style Guide* and the *Pocoo Styleguide*, from the team that brought us Flake, Sphinx, Jinja 2, and other great Python libraries.

The Hitchhiker's Guide to Python! is a collective work about writing Pythonic code. Its most prolific contributor is Kenneth Reitz, a community hero thanks to his beautifully Pythonic `requests` package. David Goodger presented a tutorial at PyCon US 2008 titled "Code Like a Pythonista: Idiomatic Python". If printed, the tutorial notes are 30 pages long. Goodger created both reStructuredText and `docutils`—the foundations of Sphinx, Python's excellent documentation system (which, by

the way, is also the official documentation system for MongoDB and many other projects).

Martijn Faassen tackles the question head-on in "What is Pythonic?" In the python-list, there is a thread with that same title. Martijn's post is from 2005, and the thread from 2003, but the Pythonic ideal hasn't changed much—neither has the language, for that matter. A great thread with "Pythonic" in the title is "Pythonic way to sum n-th list element?", from which I quoted extensively in the "Soapbox".

PEP 3099 — Things that will Not Change in Python 3000 explains why many things are the way they are, even after the major overhaul that was Python 3. For a long time, Python 3 was nicknamed Python 3000, but it arrived a few centuries sooner—to the dismay of some. PEP 3099 was written by Georg Brandl, compiling many opinions expressed by the *BDFL*, Guido van Rossum. The "Python Essays" page lists several texts by Guido himself.