Python Developer's Guide Documentation

Brett Cannon

Sep 10, 2021

CONTENTS

This guide is a comprehensive resource for *contributing* to Python – for both new and experienced contributors. It is *maintained* by the same community that maintains Python. We welcome your contributions to Python!

CHAPTER

ONE

QUICK REFERENCE

Here are the basic steps needed to get *set up* and contribute a patch. This is meant as a checklist, once you know the basics. For complete instructions please see the *setup guide*.

1. Install and set up *Git* and other dependencies (see the *Git Setup* page for detailed information).
2. Fork the CPython repository to your GitHub account and *get the source code* using:

```
git clone https://github.com/<your_username>/cpython
cd cpython
```

3. Build Python, on UNIX and Mac OS use:

```
./configure --with-pydebug && make -j
```

   and on Windows use:

```
PCbuild\build.bat -e -d
```

   See also *more detailed instructions*, *how to install and build dependencies*, and the platform-specific pages for *UNIX*, *Mac OS*, and *Windows*.

4. *Run the tests*:

```
./python -m test -j3
```

   On *most* Mac OS X systems, replace `./python` with `./python.exe`. On Windows, use `python.bat`.

5. Create a new branch where your work for the issue will go, e.g.:

```
git checkout -b fix-issue-12345 main
```

   If an issue does not already exist, please create it. Trivial issues (e.g. typo fixes) do not require any issue to be created.

6. Once you fixed the issue, run the tests, run `make patchcheck`, and if everything is ok, commit.
7. Push the branch on your fork on GitHub and *create a pull request*. Include the issue number using `bpo-NNNN` in the pull request description. For example:

```
bpo-12345: Fix some bug in spam module
```

8. Add a News entry into the `Misc/NEWS.d` directory as individual file. The news entry can be created by using blurb-it, or the blurb tool and its `blurb add` command. Please read more about `blurb` in *documentation*.

Note: First time contributors will need to sign the Contributor Licensing Agreement (CLA) as described in the *Licensing* section of this guide.

CHAPTER

TWO

QUICK LINKS

Here are some links that you probably will reference frequently while contributing to Python:

- Issue tracker
- Buildbot status
- *Where to Get Help*
- PEPs (Python Enhancement Proposals)
- *Git Bootcamp and Cheat Sheet*

CHAPTER

THREE

STATUS OF PYTHON BRANCHES

Branch	Schedule	Status	First release	End-of-life	Release manager
main	**PEP 664**	features	*2022-10-03*	*2027-10*	Pablo Galindo Salgado
3.10	**PEP 619**	prerelease	*2021-10-04*	*2026-10*	Pablo Galindo Salgado
3.9	**PEP 596**	bugfix	2020-10-05	*2025-10*	Łukasz Langa
3.8	**PEP 569**	security	2019-10-14	*2024-10*	Łukasz Langa
3.7	**PEP 537**	security	2018-06-27	*2023-06-27*	Ned Deily
3.6	**PEP 494**	security	2016-12-23	*2021-12-23*	Ned Deily

Dates in *italic* are scheduled and can be adjusted.

The main branch is currently the future Python 3.11, and is the only branch that accepts new features. The latest release for each Python version can be found on the download page.

Status:

features new features, bugfixes, and security fixes are accepted.

prerelease feature fixes, bugfixes, and security fixes are accepted for the upcoming feature release.

bugfix bugfixes and security fixes are accepted, new binaries are still released. (Also called **maintenance** mode or **stable** release)

security only security fixes are accepted and no more binaries are released, but new source-only versions can be released

end-of-life release cycle is frozen; no further changes can be pushed to it.

See also the *Development Cycle* page for more information about branches.

By default, the end-of-life is scheduled 5 years after the first release, but can be adjusted by the release manager of each branch. All Python 2 versions have reached end-of-life.

CHAPTER

FOUR

CONTRIBUTING

We encourage everyone to contribute to Python and that's why we have put up this developer's guide. If you still have questions after reviewing the material in this guide, then the Core Python Mentorship group is available to help guide new contributors through the process.

A number of individuals from the Python community have contributed to a series of excellent guides at Open Source Guides.

Core developers and contributors alike will find the following guides useful:

- How to Contribute to Open Source
- Building Welcoming Communities

Guide for contributing to Python:

New Contributors	Documentarians	Triagers	Core Developers
Getting Started	*Helping with Documentation*	*Issue Tracking*	*How to Become a Core Developer*
Where to Get Help	*Documenting Python*	*Triaging an Issue*	*Developer Log*
Lifecycle of a Pull Request	*Style guide*	*Helping Triage Issues*	*Accepting Pull Requests*
Running & Writing Tests	*reStructuredText Primer*	*Experts Index*	*Development Cycle*
Fixing "easy" Issues (and Beyond)	*Translating*		*Core Developer Motivations and Affiliations*
Following Python's Development			*Core Developers Office Hours*
Git Bootcamp and Cheat Sheet			

Advanced tasks and topics for once you are comfortable:

- *Silence Warnings From the Test Suite*
- Fixing issues found by the *buildbots*
- *Coverity Scan*
- Helping out with reviewing open pull requests. See *how to review a Pull Request*.
- *Fixing "easy" Issues (and Beyond)*

It is **recommended** that the above documents be read as needed. New contributors will build understanding of the CPython workflow by reading the sections mentioned in this table. You can stop where you feel comfortable and begin contributing immediately without reading and understanding these documents all at once. If you do choose to skip

around within the documentation, be aware that it is written assuming preceding documentation has been read so you may find it necessary to backtrack to fill in missing concepts and terminology.

CHAPTER

FIVE

PROPOSING CHANGES TO PYTHON ITSELF

Improving Python's code, documentation and tests are ongoing tasks that are never going to be "finished", as Python operates as part of an ever-evolving system of technology. An even more challenging ongoing task than these necessary maintenance activities is finding ways to make Python, in the form of the standard library and the language definition, an even better tool in a developer's toolkit.

While these kinds of change are much rarer than those described above, they do happen and that process is also described as part of this guide:

- *Adding to the Stdlib*
- *Changing the Python Language*