PyTexas Meetup Nov 2023

PEP Talk

by Mariatta

https://fosstodon.org/@mariatta

PEP 8: Style Guide for Python Code

PEP 8: Style Guide for Python Code

PEP 8: Style Guide for Python Code

"This document gives coding conventions for the Python code comprising **the standard library in the main Python distribution.**"

PEP != Style Guide Documentation

PEP != Style Guide Documentation

So … what's a PEP?

PEP 1: PEP Purpose and Guidelines

"PEP stands for **Python Enhancement Proposal**.

A PEP is a design document providing information to the Python community, or describing a new feature or its processes or environment. The PEP should provide a concise technical specification of the feature and a rationale for the feature."

What needs a PEP?

What doesn’t need a PEP?

Making Changes to Python

Bugs, crashes, docs issues, feature requests, etc

File them on GitHub: https://github.com/python/cpython

Making Changes to Python

Changes to the language itself, grammar, syntax, adding a new module, new keyword, new operator: PEP required

Example PEPs

PEP 498: Literal String Interpolation (f-strings)
PEP 572: Assignment Expressions (the “walrus” operator)
PEP 654: Exception Group and excepts
PEP 492: Coroutines with async and await
Typing PEPs: PEP 482, 483, 484, 526, 561, etc
Packaging PEPs

Governance PEPs

PEP 13: Python Language Governance
PEP 609: PyPA Governance
PEP 8000: Python Language Governance Proposal Overview
PEP 8016: The Steering Council Model

PEP 810x: Steering Council elections

Core Workflow and Infrastructure PEPs

PEP 512: Migrating from hg.python.org to GitHub
PEP 581: Using GitHub Issues for Python
PEP 545: Python Documentation Translations
PEP 676: PEP Infrastructure Process

Release PEPs

PEP 719: Python 3.13 Release Schedule
PEP 693: Python 3.12 Release Schedule
PEP 664: Python 3.11 Release Schedule
…

Also at https://devguide.python.org/versions/#versions

PEP Lifecycle

Start with an idea for Python

PEP Lifecycle

Start with an ~~idea~~ for Python

PEP Lifecycle

problem

Start with a~~n idea~~ for Python

PEP Lifecycle

Discuss your ~~idea~~ problem with core devs and contributors

- open an issue on GitHub[1]
- start a thread on Python Discourse[2]
- discuss at Python Language Summit at PyCon US

1: https://github.com/python/cpython
2: https://discuss.python.org/c/core-dev/23

PEP Lifecycle

If it can’t be solved by third party library on PyPI,

If it requires changes to the language itself

You’ll need a PEP (and a PEP sponsor)

PEP Lifecycle

Write the PEP

Champion your ~~idea~~ solution to your problem

Document discussions, questions, pushbacks, reasonings, etc

PEP Lifecycle

Submit the PEP for Steering Council's review

PEP Lifecycle

PEP accepted? Congrats!!! 🎉

It can be included in Python

PEP Lifecycle

PEP is a proposal doc, not feature documentation

The feature should be documented at docs.python.org

PEP Lifecycle

Could take months, and in some cases: years

PEP 525 (async generator): 2 months
PEP 557 (data classes): 9 months
PEP 572 (the walrus): 18 months
PEP 517 (build system/Packaging): 2 years

Most time are spent in discussions and consensus building

Your Role

PEPs will affect how you use Python (things could break)

Python core devs have blind spots: Your input matters

Bring up your concerns during PEP discussion period

Case study:

PEP 563 (2017)
vs
PEP 649, Pydantic, FastAPI (2021)

Case study:

PEP 563 (2017)
vs
PEP 649, Pydantic, FastAPI (2021)

P.S. PEP 649 was accepted May 2023

Your Role

Keep up to date with PEPs!

New PEPs are announced on Discourse: discuss.python.org/c/peps

Test your code against Python pre-releases: (alpha, beta, release candidate): check the Release Schedule

“The steering council shall work to: …

Seek consensus among contributors and the core team…”,

PEP 8016

"A PEP is a design document **providing information to the Python community**",

PEP 1

Thank You

PEP Talk
by Mariatta

https://fosstodon.org/@mariatta

PyTexas Meetup Nov 2023