Typing Summit

At PyCon US 2024

10 am – 1 pm

Schedule

- 10 am: Welcome
- 10:20 am: Typing Council
- 11:05 am: Alex Waygood: Rust and Python
- 11:30 am: break
- 11:45 am: Carl Meyer: Theory of Type Hints, Revisited
- 12:15 pm: David Foster: TypeForm
- 12:45 pm: Lightning talks
  - Sign up at https://hackmd.io/@jellezijlstra/typing-summit-2024
- 1 pm: End

Notes

https://hackmd.io/@jellezijlstra/typing-summit-2024

- Anyone is free to add notes
- Also for lightning talk signups
- Be respectful (CoC applies)

Introductions

- Name
- How you’re involved in Python typing
- What’s a missing feature from the Python type system that would have helped you recently (if any)?

Typing Council

Rebecca Chen
Shantanu Jain
Guido van Rossum
Eric Traut
Jelle Zijlstra

Agenda

- What is the Typing Council?
- Review of the first 6 months
- Statements from Council members
- Questions

Wait, what?

- The type system was created through PEPs: 484, 526, 544, 561, 563, 585, 586, 591, …
- But many gaps in coverage
- Type checkers had to fill in the gaps
- PEP 729: Typing governance process
  - Created September 2023
  - Accepted November 2023

The Typing Council

- Structure: 5 members
  - Indefinite terms
  - Replacements appointed by fiat
- Mandate: Make the type system
  - Useful
  - Usable
  - Stable

Typing Council: The first six months

Roles:

- PEP review
- Specification
- Conformance tests
- User-facing guide

PEP review

The Council advises on PEPs before they go to the Steering Council.

- PEP 696 (type parameter defaults): accepted
- PEP 705 (ReadOnly in TypedDict): accepted
- PEP 724 (stricter TypeGuard): no consensus, PEP withdrawn
- PEP 742 (TypeIs): accepted
- PEP 728 (TypedDict extra items): ?

Typing specification

- Initial spec created by copy-pasting PEPs
- Expansions
  - Tuples; NamedTuples; constructors; “type expression” definition; context managers
- Changes
  - Allow NoReturn outside of functions; unary + in Literals; TypeVar variance in functions; treatment of triple-quoted string annotations; behavior of @no_type_check; runtime use of Annotated; old positional-only syntax; TypeGuard/bool compatibility; Final/dataclasses interaction
- Assessment
  - Lots of progress
  - More work to do in clearing up and formalizing the spec

Conformance tests

- Set of tests that check whether type checkers implement the spec
- Now covers ~all of the spec
- Checked against mypy, pyre, pyright, pytype
- Room for expansion

User-facing reference

- PEP 729 listed a user-facing reference for the type system as a goal
- https://typing.readthedocs.io
- Sorry :(
- Help welcome

The road ahead

- Ensure type checkers implement the spec
- Continue to improve the spec
- Make the type system even better

What would you like to see in the type system?

- TypeForm
- Intersection
- A Map operator
- Higher-kinded types
- Shorthand syntax for Callable
- Shorthand syntax for TypedDict
- … Something we haven’t listed

Typing Council

Rebecca Chen
Shantanu Jain
Guido van Rossum
Eric Traut
Jelle Zijlstra

Questions?