An Overdue Apology

2 min read • 462 words

Themes: Programming

Dear Python friends,

I hope this blog post can serve as an official response to Why I'm Not Collaborating with Kenneth Reitz.

I owe you an apology. Reflecting on the Requests III fundraiser from years past, I realize the importance of transparency — a virtue I regrettably overlooked.

> The Requests III fundraiser raised significant community funds for developing an async-capable version of the popular HTTP library, but faced challenges around communication and delivery timelines.

From chaos comes clarity, and I see now that I should have handled things in a different manner.

Perhaps you're curious about the fate of Requests III. My aim was to craft a product of exceptional quality, and initially, everything seemed promising. However, as I delved deeper into integrating H2 support, it became apparent that the dependencies I was relying on—or considering for use—simply didn't meet the necessary standards. A key factor behind the success of Requests has always been its thoughtful design.

> Requests became Python's most downloaded package by prioritizing developer experience with its "HTTP for Humans" philosophy, making complex HTTP operations simple and intuitive.

Yet, the asynchronous landscape within Python's ecosystem failed to meet my expectations, leading me to conclude that Requests should retain its synchronous nature and I (again, quietly) licensed the codebase **CC0**.

I sincerely hope that we can continue to work together towards a common goal, to help make the world a little bit of a better place — through our code, our thoughts, and our actions.

Questions & Answers

Please submit questions via email or Twitter.

Much Love, Kenneth Reitz

What work was done on Requests III by you?

- Adding type annotations to all public interfaces within the library.
- Refactoring all the namespaces, to be much more declarative.
- Adapting the existing architecture to be compatible with an asynchronous programming model.

- Testing different async systems, deciding which one to "vet" for general usage by the masses.
- Collecting community feedback, pain points, and desired changes.
- Designing the primary "requests3.core" API, which the rest of requests3 sits upon.
- Orchestrating with the rest of the Python ecosystem, and attempting to find a suitable solution for the asynchronous low-level requirements.
- Some very, very nasty merge conflict resolution :)

What was left to do?

- Integration with a low-level HTTP library ready for the task.
- Product documentation (the hardest part).
- Beta & Release

I ultimately decided (again, not transparently) that supporting async fully was not feasible, as I could not find anyone capable or willing to work on the project with me. The psychological weight of getting everything perfect was also just too much.

> This acknowledgment of perfectionism and psychological pressure reflects common challenges faced by open source maintainers, particularly those stewarding widely-used libraries with millions of dependents.

I'm sorry I didn't finish this project without taking the time to be transparent.