

A Python experiment you can recreate next week

An October look at uv, explicit dependencies, and the small environment record that makes AI tooling easier to debug.



By Edwin Rodriguez

Published Oct 21, 2026 · 4 min read

October launch collection · Story context 2026-10-14



An experiment becomes harder to understand once you can no longer say which environment produced the result. That happens quickly with AI tooling: a framework update, a new wheel, a different Python interpreter, and suddenly yesterday's working command has a different outcome.

uv is a useful open-source tool to evaluate for managing Python projects and environments. Its project workflow centers on explicit metadata and a lockfile. For this October walkthrough, let's use it to make a small experiment easier to repeat.

The practical question is whether another developer can recreate your small experiment without reverse-engineering a long terminal history.

Give the experiment its own directory

Start with a dedicated folder and keep the purpose narrow. A tiny compatibility probe doesn't need to inherit every dependency in your main application.

The [current uv project guide](#) documents a straightforward starting sequence:

```
uv init compatibility-probe
cd compatibility-probe
uv add httpx
uv run python -c "import httpx; print(httpx.__version__)"
```

This example creates a small HTTP-library project. Check the current documentation for your chosen package and platform before adapting it to a GPU framework.

Keep the project metadata and lockfile under version control. Review their changes just as you would review source changes. A new transitive dependency can explain a changed result even when your own script stayed the same.

Record the environment outside Python too

A lockfile is useful, but native packages also interact with the operating system, driver, accelerator runtime, and available hardware. Record those details alongside the experiment.

Include the exact model revision when model weights are involved. "The small model" won't help a future reader distinguish a quantized artifact from a full-precision checkpoint with a similar name.

A short experiment note can answer:

```
Question being tested:
Source revision:
Python and dependency lock state:
OS and relevant native runtime:
Input or model revision:
Command used:
Observed result and remaining limit:
```

Keep secrets and private prompts out of that note. Use a small synthetic input that somebody else can share and inspect.

| Separate updating from reproducing

There are times to upgrade dependencies and times to reproduce an existing result. Make that intention explicit in the command and review process.

uv's documentation distinguishes locking and syncing behavior, including a locked mode that checks whether the lockfile is consistent. For an established project, `uv sync --locked` is a useful command to understand before using it in a repeatable build.

Read the tool's actual behavior rather than assuming every invocation leaves the lockfile untouched. If a command changes dependencies automatically, that may be helpful during development and surprising while reproducing an earlier result.

| Keep a small acceptance probe

Installation is one stage of the experiment. Add a check that exercises the capability you need: an import, a tiny request, a model load, or a bounded inference.

For a native package, include an operation that reaches the relevant backend. An import can succeed while the first actual accelerator operation fails. Save the failure with the exact command that produced it. Once the intended operation passes, keep that command as your quick check.

Keep the probe short enough to rerun after a dependency change. It should tell you whether the narrow capability still works without starting an expensive end-to-end workload every time.

| Review the result before expanding the environment

Once the probe works, grow toward the real task one step at a time. Add the next dependency, rerun the check, and record any new constraint. If a change breaks the environment, you now have a small difference to inspect.

This discipline is useful with any competent environment manager. uv brings a convenient project workflow worth exploring, while the quality of the experiment still depends on your inputs, notes, and validation.

Next week, you should be able to open the folder, recreate the environment, and run the same small test. If it fails, the lockfile and experiment note give you somewhere concrete to start. That's a good return for a few minutes of setup today.

| Sources & further reading

Checked 2026-10-14 · Story context 2026-10-14

1. [uv project guide](#) (2026-10-14)
2. [uv locking and syncing](#) (2026-10-14)
3. [uv source repository](#) (2026-10-14)

[Read the current article online →](#)

Support the workshop

Explore software and learning offers at [StackSocial](#). Affiliate link: ProgreTech may earn a commission from purchases. Check current price, license and renewal terms before buying.

Find our projects at [progretech.com](#).

ProgreTech LLC · Build / Learn / Improve / Together

This file contains its text, illustrations and fonts. No account, scripts or network connection required to read. External reference links need internet access.