Next-Gen Python Environments

Package Management made simple with Pixi

Data Science Tutorial 2026-01

Why Python Environments?

Projects need many Python (and non-Python) packages

Problem 1: Dependencies inside a project can conflict!

Problem 2: Dependencies between projects can conflict!

Solution: (Virtual Python) Environments

The Classic: venv + pip

python venv creates an environment in a directory (project scoped)

pip installs python dependencies in the environment

Problem 1: Only one Python version per project

Problem 2: Only Python dependencies are let in!

The Modern Solution: Conda

conda creates globally available environments (namespace scoped)

modern alternatives have made it fast(er) (mamba)

Manages many non-Python dependencies with a large ecosystem!

Problem: Environments are irreproducible and break!

The Modern Solution: Conda

Problem: Environments are irreproducible and break!

$ conda list

General (has every package possible installed, never update so it works!)
Python2.7 with IRAF (don’t update it breaks)!
Python3.8 for that one project (breaks with my other code!)
Legacy Astropy install before they changed that behavior
…

Losing environments should not mean you can’t run code!

Environments should not be general purpose!

Why Project Scoped over Namespace Scoped

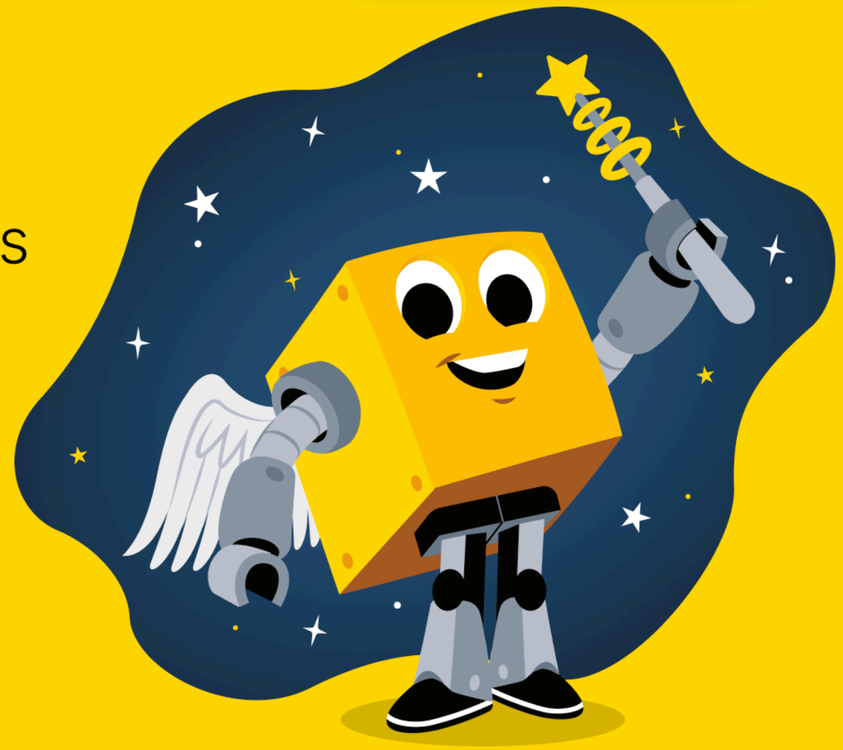
- Require each project / directory to have all of the environments and packages in needs to accomplish research tasks. Doing this we get:
  - Updating project packages is independent and doesn’t affect others.
  - Version control of packages alongside research code.
- It would be great to have a tool that makes this easy that…
  - is fast and is able to produce / reproduce environments quickly.
  - can handle Python / non-Python dependencies.
  - is built for (cross-platform) reproducibility from the ground up .
  - can track additional project requirements (envars, tasks, etc)

Pixi

prefix.dev

Fast conda & PyPI Package Manager

- Reproducible workflows with lockfiles and tasks
- Can replace apt-get, brew, and winget
- Supports Windows, macOS and Linux



LICENSE
BSD-3

The Goal for Today: Get Started with Pixi and learn its features

I tried out pixi alongside conda until I found myself not using conda!

Step 0: Install pixi

Go to pixi.prefix.dev (Note! Evaluate installer with your default shell)

- Pixi pulls from conda compatible channels as well as PyPI (and more!).
- Uses the uv installer under the hood for PyPI.
- Pixi can be used to develop other projects (C, Rust, etc).
- Pixi is written in Rust, extremely performant and fast.
- Note for HPC users, pixi also has a cache (PIXI_CACHE_DIR)