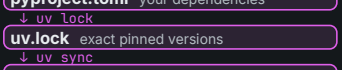
uv — Fast Python package manager

CONCEPTS

uv run and uv add lock and sync automatically, so you rarely need uv lock or uv sync by hand. Commit uv.lock; keep .venv out of git.



Project workflow

uv pip … is a drop-in style interface for existing venvs and requirements files; it does not touch pyproject.toml or uv.lock. Use uv add for projects.

PROJECTS

uv init *<name>*
Create a project in a new directory

uv init
Initialise the current directory

uv add *<pkg>* **...**
Add dependencies and install them

uv add '*<pkg>* **≥** *<ver>***'**
Add with a version constraint

uv add --dev *<pkg>*
Add to the dev dependency group

uv remove *<pkg>* **...**
Remove dependencies

uv sync
Make .venv match the lockfile exactly

uv sync --locked
Fail if uv.lock is out of date (CI)

uv lock --upgrade
Re-resolve to the newest allowed versions

uv lock --upgrade-package *<pkg>*
Upgrade one package only

RUNNING

uv run *<cmd>* **[args]**
Run a command in the project environment

RUNNING (cont.)

uv run python
Python REPL with project deps

uv run *<script.py>*
Run a script (inline metadata is honoured)

uv run --with *<pkg>* *<cmd>*
Add a temporary dependency for this run

TOOLS

uvx *<tool>* **[args]**
Run a tool in a throwaway env

uvx --from *<pkg>* *<cmd>*
Command name differs from package name

uv tool install *<pkg>*
Install a tool onto PATH

uv tool list
List installed tools

PYTHON VERSIONS

uv python install *<ver>*
Download a managed Python

uv python list
Show available and installed Pythons

uv python pin *<ver>*
Write .python-version for this directory

VIRTUAL ENVS AND PIP

uv venv [*<path>***]**
Create a venv (default .venv)

uv venv --python *<ver>*
Create with a specific Python

uv pip install *<pkg>*
Install into the active or local .venv

uv pip install -r requirements.txt
Install from a requirements file

uv pip compile *<in>* **-o requirements.txt**
Resolve to a pinned requirements file

CACHE AND SELF

uv --version
Print the uv version

uv help *<command>*
Show help for a command