

astral/uv Cheatsheet

The ultra-fast Python package manager, resolver, and installer
github.com/carlonicolini/uv_cheatsheet
January 22, 2026

Project Lifecycle

- `uv init` *Create a new project (app)*
- `uv init -lib` *Create a library project*
- `uv sync` *Install deps and sync lockfile*
- `uv lock` *Generate/update uv.lock*
- `uv tree` *View dependency graph*
- `uv build` *Build sdist and wheel*
- `uv publish` *Upload to PyPI*

Dependency Management

- `uv add <pkg>` *Add dependency*
- `uv add -dev <pkg>` *Add dev dependency*
- `uv add -optional <group>` *Add extra*
- `uv remove <pkg>` *Remove dependency*
- `uv lock -upgrade` *Update all deps*
- `uv export -format requirements-txt`

Python Management

- `uv python install <v>` *Install Python*
- `uv python list` *List available versions*
- `uv python pin <v>` *Pin version for dir*
- `uv python find` *Locate interpreter*
- `uv python uninstall <v>` *Remove ver*

Running & Scripts

- `uv run <file.py>` *Run in project env*
- `uv run <cmd>` *Run tool in env*
- `uv run -with <pkg>` *Ephemeral dep*
- `uvx <tool>` *Install + run tool (ephemeral)*
- `uv tool install <pkg>` *Global install*

Workspaces (Monorepos)

- `uv sync -all-packages` *Sync whole repo*
- `uv run -package <name>` *Run for member*
- `uv tree -package <name>` *Specific tree*

Legacy Pip Support

- `uv pip install -r reqs.txt` *Fast install*
- `uv pip compile Lock reqs.txt`
- `uv pip list` *List env packages*
- `uv pip freeze` *Output installed pkgs*
- `uv venv` *Manual virtualenv creation*

Environment Variables

- `UV_PYTHON` *Set default interpreter*
- `UV_INDEX_URL` *Primary index URL*
- `UV_CACHE_DIR` *Change cache path*
- `UV_OFFLINE=1` *No network access*
- `UV_NO_CACHE=1` *Disable caching*

Advanced Specifiers

- `uv add "pkg">=1.2` *Ver constraint*
- `uv add "git+https://..."` *Git source*
- `uv add "./path"` *Local dependency*
- `uv add -raw-sources` *Direct URLs*

Common Flags

- `-frozen` *Error if lockfile changes*
- `-no-sync` *Skip env sync after lock*
- `-all-extras` *Install all optional deps*
- `-system` *Target system site-pkgs*

Maintenance

- `uv self update` *Update uv binary*
- `uv cache clean` *Clear all cache*
- `uv cache prune` *Remove old cache*
- `uv help <cmd>` *Command documentation*

Project Files: `pyproject.toml` (Meta/Deps) `uv.lock` (Deterministic) `.python-version` (Pinned)

Example: pyproject.toml (Library with CLI)

```
[project]
name = "my-cool-lib"
version = "0.1.0"
description = "my python library"
readme = "README.md"
requires-python = ">=3.10"
dependencies = [
    "requests>=2.31.0",
    "pydantic>=2.0",
]

[project.optional-dependencies]
dev = [
    "pytest>=8.0.0",
    "ruff>=0.3.0",
]

[project.scripts]
my-cli = "my_cool_lib.cli:main"

[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"

[tool.uv]
dev-dependencies = ["pytest-cov"]
```