uv Quick Reference

This reference provides essential uv commands organized by workflow. For comprehensive documentation, visit the uv docs. Need help getting started? Check out how to install uv and create your first Python project.

Project Creation

COMMAND	DESCRIPTION
`uv init`	Initialize new Python project in current directory with pyproject.toml
`uv init myproject`	Create new project directory and initialize with project structure
`uv init --package`	Create project to be built as Python package
`uv init --app`	Create application project
`uv init --lib`	Create library project
`uv init --script`	Create single-file script
`uv init --bare`	Create only pyproject.toml file
`uv init --python 3.12`	Specify Python version

Dependency Management

COMMAND	DESCRIPTION
`uv add requests`	Add production dependency to project and update lockfile
`uv add --dev pytest`	Add development dependency for testing and development tools
`uv add -r requirements.txt`	Import from requirements file (migration guide)
`uv remove requests`	Remove dependency

COMMAND	DESCRIPTION
`uv sync`	Update project environment and refresh lockfile with latest compatible versions
`uv lock`	Generate or update lockfile
`uv lock --upgrade`	Upgrade all dependencies in lockfile
`uv tree`	View dependency tree

Script Management

COMMAND	DESCRIPTION
`uv init --script myscript.py`	Create single-file script with inline metadata
`uv add --script myscript.py click`	Add dependency to script inline metadata
`uv run myscript.py`	Execute Python script in managed virtual environment with project dependencies
`uv run --with requests myscript.py`	Run script with temporary dependencies without adding to project

Want to learn about self-contained scripts? See how to write self-contained Python scripts.

Tool Management

COMMAND	DESCRIPTION
`uvx pytest`	Run tool in ephemeral environment
`uv tool install ruff`	Install tool globally for system-wide use
`uv tool list`	List installed tools
`uv tool upgrade ruff`	Update specific tool

COMMAND	DESCRIPTION
`uv tool upgrade --all`	Update all tools

Python Version Management

COMMAND	DESCRIPTION
`uv python list`	Show available Python versions
`uv python install 3.12`	Download and install specific Python version for project use
`uv python pin 3.12`	Pin project to specific Python version
`uv run --python 3.12 python`	Run Python using specific version

Learn more about managing Python versions in uv projects and adding Python to your system PATH.

Building & Publishing

COMMAND	DESCRIPTION
`uv build`	Build distribution packages
`uv publish`	Upload to PyPI
`uv publish --publish-url https://test.pypi.org/legacy/`	Upload to TestPyPI

Ready to publish? Follow our complete PyPI publishing guide.

Version Management

COMMAND	DESCRIPTION
`uv version`	Show current version
`uv version --bump patch`	Increment patch version

COMMAND	DESCRIPTION
`uv version --bump minor`	Increment minor version
`uv version --bump major`	Increment major version

Virtual Environments

COMMAND	DESCRIPTION
`uv venv`	Create virtual environment
`uv venv --python 3.12`	Create with specific Python

New to virtual environments? Read why you should use a virtual environment.

Utility Commands

COMMAND	DESCRIPTION
`uv --help`	Show help
`uv help sync`	Help for specific command
`uv self update`	Update uv itself
`uv cache clean`	Clear package cache

Common Workflows

New Project Setup

```
uv init myproject
cd myproject
uv add requests pandas
uv add --dev pytest ruff
uv run python main.py
```

Existing Project

```
git clone <repo>
cd <repo>
uv sync
uv run pytest
```

Need help with testing? Check out setting up testing with pytest and uv and how to run tests using uv.

Quick Script

```
uv init --script analyze.py
uv add --script analyze.py pandas matplotlib
uv run analyze.py
```

Configuration

pyproject.toml Essentials

```
[project]
name = "myproject"
version = "0.1.0"
dependencies = ["requests>=2.28.0"]

[tool.uv]
dev-dependencies = [
    "pytest>=7.0",
    "ruff>=0.1.0",
]
```

Environment Variables

- `UV_CACHE_DIR` - Set cache directory
- `UV_PYTHON_INSTALL_DIR` - Python installation location

- `UV_INDEX_URL` - Default package index

Tips

- Most commands accept `--python 3.x` to specify version
- Use `--with package` to add temporary dependencies
- `uv sync --locked` enforces exact lockfile versions
- `uv run` automatically manages virtual environments
