Skip to content

[![logo](../../assets/logo-letter.svg)](../.. "uv")

uv

Declaring dependencies

Initializing search

[uv](https://github.com/astral-sh/uv "Go to repository")

[![logo](../../assets/logo-letter.svg)](../.. "uv") uv

[uv](https://github.com/astral-sh/uv "Go to repository")

  * [Introduction](../..)
  * [Getting started](../../getting-started/)

Getting started

    * [Installation](../../getting-started/installation/)
    * [First steps](../../getting-started/first-steps/)
    * [Features](../../getting-started/features/)
    * [Getting help](../../getting-started/help/)
  * [Guides](../../guides/)

Guides

* [Installing Python](../../guides/install-python/)
* [Running scripts](../../guides/scripts/)
* [Using tools](../../guides/tools/)
* [Working on projects](../../guides/projects/)
* [Publishing packages](../../guides/package/)
* [Integrations](../../guides/integration/)

Integrations

* [Docker](../../guides/integration/docker/)
* [Jupyter](../../guides/integration/jupyter/)
* [GitHub Actions](../../guides/integration/github/)
* [GitLab CI/CD](../../guides/integration/gitlab/)
* [Pre-commit](../../guides/integration/pre-commit/)
* [PyTorch](../../guides/integration/pytorch/)
* [FastAPI](../../guides/integration/fastapi/)
* [Alternative indexes](../../guides/integration/alternative-indexes/)
* [Dependency bots](../../guides/integration/dependency-bots/)
* [AWS Lambda](../../guides/integration/aws-lambda/)

* [Concepts](../../concepts/)

Concepts

* [Projects](../../concepts/projects/)

Projects

* [Structure and files](../../concepts/projects/layout/)
* [Creating projects](../../concepts/projects/init/)
* [Managing dependencies](../../concepts/projects/dependencies/)
* [Running commands](../../concepts/projects/run/)
* [Locking and syncing](../../concepts/projects/sync/)
* [Configuring projects](../../concepts/projects/config/)
* [Building distributions](../../concepts/projects/build/)
* [Using workspaces](../../concepts/projects/workspaces/)

* [Tools](../../concepts/tools/)
* [Python versions](../../concepts/python-versions/)
* [Resolution](../../concepts/resolution/)
* [Caching](../../concepts/cache/)

* [Configuration](../../configuration/)

Configuration

* [Configuration files](../../configuration/files/)
* [Environment variables](../../configuration/environment/)
* [Authentication](../../configuration/authentication/)
* [Package indexes](../../configuration/indexes/)
* [Installer](../../configuration/installer/)

* [The pip interface](../)

The pip interface

* [Using environments](../environments/)
* [Managing packages](../packages/)
* [Inspecting packages](../inspection/)
* Declaring dependencies [Declaring dependencies](./) Table of contents
  * Using pyproject.toml
  * Using requirements.in
* [Locking environments](../compile/)
* [Compatibility with pip](../compatibility/)

* [Reference](../../reference/)

Reference

* [Commands](../../reference/cli/)
* [Settings](../../reference/settings/)
* [Troubleshooting](../../reference/troubleshooting/)

Troubleshooting

* [Build failures](../../reference/troubleshooting/build-failures/)
* [Reproducible examples](../../reference/troubleshooting/reproducible-examples/)

* [Resolver](../../reference/resolver-internals/)
* [Benchmarks](../../reference/benchmarks/)
* [Policies](../../reference/policies/)

Policies

* [Versioning](../../reference/policies/versioning/)

    * [Platform support](../../reference/policies/platforms/)
    * [License](../../reference/policies/license/)

Table of contents

Declaring dependencies

It is best practice to declare dependencies in a static file instead of modifying environments with ad-hoc installations. Once dependencies are defined, they can be [locked](../compile/) to create a consistent, reproducible environment.

Using `pyproject.toml`

The `pyproject.toml` file is the Python standard for defining configuration for a project.

To define project dependencies in a `pyproject.toml` file:

pyproject.toml

```
[project]
dependencies = [
  "httpx",
  "ruff>=0.3.0"
]
```

To define optional dependencies in a `pyproject.toml` file:

pyproject.toml

```
[project.optional-dependencies]
cli = [
  "rich",
  "click",
]
```

Each of the keys defines an "extra", which can be installed using the `--extra` and `--all-extras` flags or `package[<extra>]` syntax. See the documentation on [installing packages](../packages/#installing-packages-from-files) for more details.

See the official [`pyproject.toml`

guide](https://packaging.python.org/en/latest/guides/writing-pyproject-toml/)

for more details on getting started with a `pyproject.toml`.

Using `requirements.in`

It is also common to use a lightweight `requirements.txt` format to declare

the dependencies for the project. Each requirement is defined on its own line.

Commonly, this file is called `requirements.in` to distinguish it from

`requirements.txt` which is used for the locked dependencies.

To define dependencies in a `requirements.in` file:

requirements.in

```
httpx
ruff>=0.3.0
```

Optional dependencies groups are not supported in this format.

August 27, 2024

Made with [Material for MkDocs Insiders](https://squidfunk.github.io/mkdocs-material/)

[ ](https://github.com/astral-sh/uv "github.com") [ ](https://discord.com/invite/astral-sh "discord.com") [ ](https://pypi.org/project/uv/ "pypi.org") [ ](https://x.com/astral_sh "x.com")