Nox Python session runner

START HERE

`nox -l`
List sessions

`nox -p 3.12`
Only one Python version

`nox`
Run default sessions

`nox -t <tag>`
Run sessions by tag

`nox -s <session>`
Run one session

`nox -R`
Reuse envs, skip installs

`nox -s tests -- -x`
Pass args to the session

`nox -s <session> -r`
Reuse the venv

noxfile.py @nox.session → `nox -s` → **Virtualenv** .nox/<name> → `session.install` → **Dependencies** pip / uv → `session.run` → **Commands** tests, lint, docs

CONCEPTS

Sessions are plain Python functions in noxfile.py; each gets its own fresh virtualenv under .nox/.

Without -s, every session in nox.options.sessions (or all, if unset) runs in file order.

Run as python -m nox to use the exact interpreter you want for Nox itself.

RUNNING

`nox`
Run all default sessions

`nox -s <session> [<session>...]`
Run named sessions
Also -e or --session; env var NOXSESSION takes a comma list

`nox -l`
List sessions and their Pythons
--json gives machine-readable output

`nox -- <args>`
Pass args to sessions as posargs
Everything after -- lands in session.posargs

`nox -p <version>`
Only sessions using this Python
Env var NOXPYTHON; several versions allowed

`nox -f <file>`
Use another noxfile

`nox -t <tag> [<tag>...]`
Run sessions with a tag

`nox -k "<expr>"`
Select by keyword expression
pytest-style: -k "tests and not lint"

`nox -s "tests(dj='5')"`
Run one parametrized variant
Plain -s tests runs every variant

`nox -x`
Stop after the first failure

`nox --usage <session>`
Show a session's docstring

`nox --extra-pythons <version>`
Add Pythons beyond the noxfile

`nox -P <version>`
Force this Python for all sessions
Works even for sessions with no python set

`nox -v`
Show output of silent commands

`nox --report <file.json>`
Write a JSON result report

RUNNING (cont.)

`nox --envdir <dir>`
Put virtualenvs elsewhere than .nox

`nox --nocolor / --forcecolor`
Control colored output

`nox -ts`
Add timestamps to the log

`nox -j <N|auto>`
Run sessions in parallel
Sessions must opt in with allow_parallel=True

`nox --allow-parallel`
Let every session run in parallel

`nox --error-on-missing-interpreters`
Fail instead of skipping
Already the default when CI is set

`nox --error-on-external-run`
Fail on non-venv programs
Needs external=True per call to allow

`nox --non-interactive`
Force session.interactive off

ENVIRONMENTS & EXECUTION

`nox -r`
Reuse existing virtualenvs
Alias for --reuse-venv=yes; default recreates each run

`nox -R`
Reuse venvs and skip installs
Makes session.install a no-op

`nox --install-only`
Install deps, skip session.run
Handy for warming caches in CI

`nox --no-install`
Skip install calls on reused venv

`nox -N`
Undo reuse set in the noxfile

`nox --reuse-venv always|never`
Override per-session reuse

`nox -db uv`
Default venv backend (uv, virtualenv...)
Fallback chain works: uv|virtualenv

`nox -fb <backend>`
Force a backend for all sessions

`nox --no-venv`
Run in the current interpreter
Alias for -fb none

`nox --download-python auto|never|always`
Fetch missing Pythons

ENVIRONMENTS & EXECUTION (cont.)

`tox-to-nox`
Draft a noxfile from tox.ini
Experimental; review the output

`eval "$(register-python-argcomplete nox)"`
Enable shell completion (bash/zsh)

NOXFILE

`import nox`
Load the library

`@nox.session`
Define a session function

`@nox.session(python=["3.11", "3.12"])`
Run once per Python

`session.install("<pkg>")`
Install into the session venv

`session.install("-e", ".")`
Install the project editable

`session.run("<cmd>", "<arg>")`
Run a command in the venv

`session.posargs`
Args given after --

`session.run("pytest", *session.posargs)`
Forward posargs to a tool

`@nox.session(name="<name>")`
Set a name different from the function

`@nox.session(tags=["<tag>"])`
Group sessions for -t

`@nox.parametrize("<arg>", [<v1>, <v2>])`
Run once per value

`@nox.session(venv_backend="uv")`
Choose backend per session

`@nox.session(reuse_venv=True)`
Reuse this session's venv

`session.notify("<session>")`
Queue another session after this one

`session.skip("<reason>")`
Skip the session with a message

`session.chdir("<dir>")`
Change directory for later commands

`session.run("<cmd>", external=True)`
Allow a program outside the venv

NOXFILE (cont.)

`session.run(..., env={"K": "V"})`
Set env vars for one command

`session.run_install("<cmd>")`
Run a setup command, skipped with -R

`session.run_always("<cmd>")`
Run even with --install-only

`session.create_tmp()`
Make a temp dir inside the env

`session.error("<msg>")`
Fail the session with a message

`session.python`
Python version of this run

`nox.options.sessions = [...]`
Default sessions when none given

`nox.options.default_venv_backend = "uv"`
Set backend for the whole file

`@nox.session(requires=["<session>"])`
Run another session first

`@nox.session(default=False)`
Exclude from the default run

`@nox.session(python=False)`
Run without a virtualenv

RECIPES

Test matrix with uv

```
import nox

nox.options.
default_venv_backend = "uv"

@nox.session(python=["3.11",
"3.12", "3.13"])
def tests(session):
    session.install("-e", ".",
"pytest")
    session.run("pytest",
*session.posargs)
```

Lint and format sessions

```
@nox.session(tags=["style"])
def lint(session):
    session.install("ruff")
    session.run("ruff",
"check", ".")

# nox -t style
```

Fast local loop

```
nox -s tests -p 3.12 -R -- -x -
k parser

# first time: nox -s tests -p
3.12 -r
```

Warm environments in CI

```
nox --install-only
nox -R --error-on-missing-
interpreters
```