

The command line

ar022 · 11 July 2026

`demolab` is one command runner wrapping `uv` (Python) and `typst` (publishing); you never call `pip` or `python` directly. Every command finds the lab by walking up from wherever you are, like `git` finding its root, so it works from any subdirectory. Run `demolab` with no arguments for the live catalog; this page mirrors its groups. Most of these are driven by a coding agent following a runbook. The four a human types daily are `dev`, `run`, `build`, and `test`.

Start here

init Start a new lab in the current directory: root files, the content structure, a tests CI workflow, and a first git commit (skipped inside an existing repository or without a git identity). Refuses a non-empty directory and refuses to run inside an existing lab; `--force` overrides both. Typically run once, by the agent, at the top of the GETTING-STARTED runbook.

docs Bare `demolab docs` prints the agent manual plus the menu of runbooks and guides; `demolab docs <NAME>` prints that document's path, and `--print` prints its contents instead. The menu also lists `AGENT`, `CHANGELOG`, `DEMO`, and `STARTERS`. This is the agent's orientation command; a human rarely needs it.

Setup

install Install the lab's Python dependencies (`uv sync`). Rarely needed by hand: the commands that need the venv go through `uv` and create it on demand.

version Print the installed engine version.

Scaffolding

Agent territory: these run inside runbooks, not day to day.

scaffold (Re)lay the working-tree structure (`writings/`, `experiments/`, `tools/`, `artifacts/`, plus config). Non-destructive: it never clobbers an existing file, so it doubles as a repair command. For a worked first experiment, model on a starter rather than installing bulk content: `demolab docs STARTERS` prints the dir (`monte-carlo-pi` is the canonical one).

deploy-setup Opt in to GitHub Pages: copies `deploy.yml` (build main to the `gh-pages` branch) and `preview.yml` (per-PR previews) into `.github/workflows/`, then prints the remaining GitHub-UI clicks, which can't be scripted.

The loop

Running an experiment There is no `run` command — invoke the runner directly. `uv run python experiments/exp000.py` executes `experiments/exp000.py` in the lab's venv; the runner stages figures and a `numbers.json` under `artifacts/data/exp000/`. Daily driver.

new Not a scaffolder: it prints how to start a new experiment, which is to ask your coding agent to model one on an existing runner + writeup pair.

Publishing

dev Serve the site with hot-reload and in-browser build errors. Takes an optional port (default: first free from 3000). `--demo` serves the packaged docs site from a scratch copy under `temp/` instead of your lab; `--landing` (only with `--demo`) previews its landing page too. For your own landing page, put a `landing.typ` at the lab root and plain `demolab dev` renders it. The other daily driver:

`demolab dev 4000 --demo --landing`

build Build the whole bundle once: the website into `artifacts/site/` and the PDFs plus the bound book into `artifacts/pdfs/`. What CI runs.

slides Compile standalone Typst decks (any `writings/*.typ` without the meta+body shape) to `artifacts/pdfs/`.

playground Launch a lab's interactive Streamlit app (`experiments/playground.py`), if it has one, at `http://localhost:8501`.

Quality & housekeeping

test Run the Python test suite (pytest). A fresh lab with no tests yet passes rather than fails. Run it before committing.

clean Delete the regenerable build output: `temp/` and `artifacts/site/`. The committed record under `artifacts/data/` and `artifacts/pdfs/` is untouched.