Pybites Foundations Cohort

Build Your First Real Python Project: Developer Journal App

Duration: 6 weeks
Format: Self-paced lessons + weekly milestones, project work and community support
Final project: A developer journal CLI app you can run, extend, and show in your portfolio.

Program Overview

In this beginner-friendly cohort you'll learn Python *by building a real tool you'll actually use*: a command-line Developer Journal.

Step by step you'll go from:

- basic scripts → a structured CLI app with commands
- loose functions → dataclasses and classes with clear responsibilities
- "does it run?" → automated tests, coverage, and nicer terminal UX
- local script → something you can package, export, and (optionally) publish.

You'll practise core skills every professional Python dev uses: Git, GitHub, uv, ruff, Typer, pytest, Rich, JSON, basic packaging and testing.

What You'll Build

By the end of the cohort you'll have:

- A **Dev Journal CLI** that lets you:
    - add, list, search and tag entries
    - store entries on disk using JSON
    - optionally filter by tag, show stats and limit to recent entries
- A **small test suite** (pytest + pytest-cov) with useful coverage
- **Polished CLI UX** using Rich (colors, formatting)
- A **clean project layout** (modules, class to encapsulate storage)
- Export options (e.g. **PDF / HTML / Markdown** export of your journal)
- Optional stretch goals:
    - NiceGUI frontend
    - Makefile and end-to-end CLI tests
    - Publish your package to **PyPI**

Who This Is For

- Beginners who know a little Python but haven't built and shipped a full project yet – we take you from tutorial follower to **shipping a real CLI app with tests, Git, and packaging**, supported by access to our Python Platform.
- Self-taught devs who want to learn the *workflow*: Git, CLI tools, tests, packaging.
- Anyone who wants a structured 6-week path to "I ship real code."

Tech & Tools You'll Use

- **Python:** modern Python with type hints and dataclasses
- **uv:** project + dependency management
- **Git & GitHub:** branches, pull requests, basic collaboration
- **Code quality:** ruff + pre-commit (optional but encouraged)
- **CLI framework:** Typer
- **Testing:** pytest, pytest-cov
- **Terminal UX:** Rich
- **Packaging & automation (optional):** package structure, Makefile, PyPI
- **Frontend (optional):** NiceGUI for a simple GUI wrapper

Course Structure & Weekly Breakdown

Intro (Immediately after enrollment)

Focus: Onboarding & mindset

- Access the **GitHub repo** and get your local environment ready
- Join the first group call and introduce yourself
- Walk through the **app mindmap / cohort structure**
- Learn how we'll work with **Git, branches and pull requests**
- Expectations for **collaboration** with other cohort members
- Start a **"wins" file** to track your progress
- Short lesson: *"Should I use AI tools?"* – how to use AI responsibly in this cohort

---

Week 1 – Setup & Simple MVP

Theme: From zero to a tiny working journal

You'll learn

- Using **uv** to create and manage a Python project
- Optional: setting up **pre-commit + ruff** for instant feedback
- **Dataclasses** for modelling a journal entry
- Reading/writing data with **JSON serialization**

Project work

- Define a `JournalEntry` dataclass
- Implement functions to **save / load entries** from a JSON file
- Implement a minimal MVP:
  - `add` an entry (using `input()`)
  - `list` and **print all** entries

Outcome: A simple script that lets you add and list journal entries from the command line, stored in a JSON file.

Week 2 – Typer CLI & Improvements

Theme: Turn the script into a real CLI app

You'll learn

- Intro to **Type Hints in Python**
- Using **Typer** to build an elegant CLI (`list`, `add`, …)
- Basics of **validation and exception handling**

Project work

- Wrap your journal logic in a **Typer CLI** (at least `add` + `list`)
- Add better **input validation** and clear error messages
- Improve **entry display formatting** (dates, truncation, etc.)
- Optional: add a command to **seed fake data** for testing
- Optional: limit `list` output to the **most recent N entries**

Outcome: A usable CLI tool with subcommands and more robust behaviour.

Week 3 – More Features

Theme: Make the journal actually useful for everyday dev life

You'll learn

- How to extend a data model with **tags**
- Basic **search** patterns over in-memory and JSON data
- Thinking about simple **stats / analytics** on your data

Project work

- Add **tags** to `JournalEntry` + commands to add/list them
- Implement a **search command** (by keyword in title/content)
- Optional:
    - Filter entries by one or more **tags**
    - Basic **journal stats** (entry count, entries per tag, etc.)
    - Delete entries, fuzzy search, and other niceties

Outcome: A richer journal app that you can actually use to track your learning.

Week 4 – Package & Testing

Theme: From "script" to "small codebase"

You'll learn

- **Modularising** your code into multiple files
- Encapsulating data storage in a **class**
- Intro to **pytest** and writing your **first unit tests**

Project work

- Break the script into **modules** (e.g. `models.py`, `storage.py`, `cli.py`, …)
- Create a **storage class** to handle all file/JSON operations
- Write your first **pytest tests** for key functions / methods
- Optional: introduce a simple **package structure** for the project

Outcome: A cleaner codebase that's easier to reason about, with your first automated tests in place.

Week 5 – More Testing & Rich

Theme: Confidence + polish

You'll learn

- Measuring **test coverage** with `pytest-cov`
- Using **Rich** to make CLI output more readable and friendly
- (Optional) basic **end-to-end tests** for the CLI
- (Optional) using a **Makefile** to automate common tasks

Project work

- Expand your test suite to hit a sensible **coverage target**
- Use **Rich** to:
  - style headings and section separators
  - create nicer tables / lists of entries
- Optional:
  - Add simple **end-to-end CLI tests** (e.g. via subprocess or Typer testing utilities)
  - Create a **Makefile** for commands like `make test`, `make run`, `make lint`

Outcome: A tested, pleasant-to-use CLI that feels more like a real tool than a toy script.

Week 6 – Final Polish & Wrap-Up

Theme: Ship it and show it off

You'll learn

- How to write a concise, user-centric **README**
- Exporting data as **Markdown / HTML / PDF**
- Basics of **sharing and distributing** your project

Project work

- Write a clear **README** (what it does, how to install, how to use)
- Add a **PDF (or HTML/MD) export** of your journal entries
- Share your project with the group / publicly (GitHub repo link)
- Optional:
  - Build a simple **NiceGUI frontend** on top of your CLI / core logic
  - Package and **publish to PyPI**

Outcome: A small but complete Python project: documented, tested, polished and ready to show in your portfolio.

End-of-Cohort Outcomes

By the end of Foundations you will:

- Have **shipped** a real Python project end-to-end
- Be comfortable working with **Git, GitHub and uv**
- Know how to structure code into **modules, dataclasses and classes**
- Have written and run **pytest tests** and looked at **coverage**
- Understand how to **package, document and share** a small project
- Have a concrete example of professional workflow you can build on in future cohorts.