Lab Project Overview

AI-Assisted Programming for PhD Researchers

Days 1 and 2 run on a single shared project: a field-study analysis pipeline you inherit, fix, and rebuild across five labs. This page is the map for that project: what it is, how to get it, what you build in each lab, and how to catch up if you fall behind.

The Lab Project

You inherit `legacy_analysis.py`, one script from a PhD student who has left the group, plus a folder of raw measurements. It runs on nobody's machine but theirs, mixes units without checking them, and treats missing values as real readings. Over five labs, you turn that inherited mess into a tested, documented, CLI-driven pipeline: a package with a cleaning module, a statistics module, plots, a report, tests, and a command-line entry point.

This arc mirrors real research life more than any toy exercise could. You will inherit code from a labmate, a collaborator, or your own past self; it will not run out of the box; and the fastest way through is not to rewrite everything from scratch but to understand it, fix it in small verified steps, and leave it better documented than you found it. Every convention you practice here (explain before you accept, verify before you trust, small commits that tell a story) is the same discipline you will need on your own data.

Get It

Clone the starter repository:

```
git clone https://github.com/beyondsimulations/aiprog-lab-starter
cd aiprog-lab-starter
```

A fresh clone lands you on the **inherited starting state**: the same mess described above, on the `main` branch. The lab checkpoints are not separate branches; they are git **tags** (`lab-01-done` … `lab-05-done`) that mark the reference end-state after each lab. You will create those states yourself as you work through the labs; the tags exist so you can check your progress or catch up if you fall behind (see below).

The Arc

Lab	You build	Tag
Lab 1	Understanding the inherited script + a project `AGENTS.md` + the path fix that makes it run	`lab-01-done`

Lab	You build	Tag
Lab 2	A written spec, then the cleaning module (`src/pipeline/io.py`)	`lab-02-done`
Lab 3	Tests and a statistics module	`lab-03-done`
Lab 4	Plots and a report, plus a CLI you build through structured review	`lab-04-done`
Lab 5	A references list via MCP, installed writing skills, and a git worktree exercise	`lab-05-done`

Catching Up

If you fall behind, or want to compare your work against the reference state, fetch the tags and check one out on a new branch:

```
git fetch --tags && git checkout tags/lab-0N-done -b catchup-0N
```

Replace `0N` with the lab number, for example `lab-03-done` and `catchup-03`. Catching up is normal. The goal of any lab is not to have written every line yourself; it is to arrive at the next exercise with a working repository and enough understanding to keep going.

Final Architecture

By the end of Lab 5, the repository looks like this:

```
aiprog-lab-starter/
├── AGENTS.md
├── pyproject.toml
├── src/pipeline/__init__.py
├── src/pipeline/io.py
├── src/pipeline/stats.py
├── src/pipeline/plots.py
├── src/pipeline/report.py
├── src/pipeline/cli.py
├── tests/test_io.py
├── tests/test_stats.py
├── docs/references.md
└── capstone/liter/sample.bib
```

`legacy_analysis.py` is gone by Lab 2, replaced by the `src/pipeline/` package it was hiding inside. `capstone/liter/sample.bib` is not part of the pipeline; it is a sample dataset for the Literature Manager capstone option, shipped alongside the rest of the repository.

Bibliography