PERSONAL AI LEARNING CHART

Sam Park

Developer Camp · 4 weeks · 3× per week, 45 min per session

Name	Sam Park
Track	Developer Camp
Start	2026-06-13
Duration	4 weeks
Sessions	3 per week
Commitment	45 min per session

Lighthouse Chart — Sam Park

1. Opening / Bearings

You have been writing Python and running your own server for four years, and now you want the AI tools to actually keep up with you — not slow you down with suggestions you have to undo. Three specific things are on the table: catching your own bugs before a PR lands, getting real test coverage on a module you already know works but cannot prove it, and shipping one portfolio project that demonstrates you can take something from blank repo to live URL. That last one matters more than it might look on the surface, because it is the artifact that will do the talking when college applications open.

Four weeks, three sessions a week, 45 minutes a session. That is 12 sessions total — enough to build real habits without burning out on evenings you do not have. The plan below treats your time as the constraint it is.

One thing worth naming at the start: you already know the craft. The AI tools here are not teaching you to code — they are giving you a faster feedback loop on code you already know how to write.

That framing matters, because it changes how you use the tools. You are not asking Claude or Copilot to think for you; you are asking them to be a second pair of eyes that never gets tired.

---

2. Track Context

Your track is Developer Camp — specifically the flavor of it that is about integrating AI into a solo engineering workflow rather than learning to build AI systems from scratch. That distinction is worth being clear about, because the two paths look similar from the outside but pull in different directions.

The developer-camp track for someone at your level is about three things:

Closing the feedback loop. Right now, the gap between writing code and knowing whether it is correct is filled by running it, reading it again later, or having someone else review it. AI shrinks that gap. A well-prompted Claude review of a diff takes about two minutes and surfaces the class of bugs — off-by-one errors in date ranges, missing None checks, SQL edge cases — that are easy to miss when you wrote the code five minutes ago.

Generating the scaffolding you already know how to write. Copilot is best used not for the interesting parts of your code but for the parts you could write yourself in ten minutes but would rather not: boilerplate test fixtures, repetitive route handlers, standard error-handling patterns. Let it generate those so you spend your 45 minutes on the parts that require judgment.

Building a paper trail. For university applications, "I used AI tools" is less interesting than "here is a live project, here is the test suite, here is the commit history." The goal of this plan is to produce exactly that paper trail — concrete, public, and yours.

The secondary track here is teammate-developer, which reflects the open-source contribution habit you already have. The PR review workflow you build in week one will transfer directly to the quality of the PRs you submit to external repos.

---

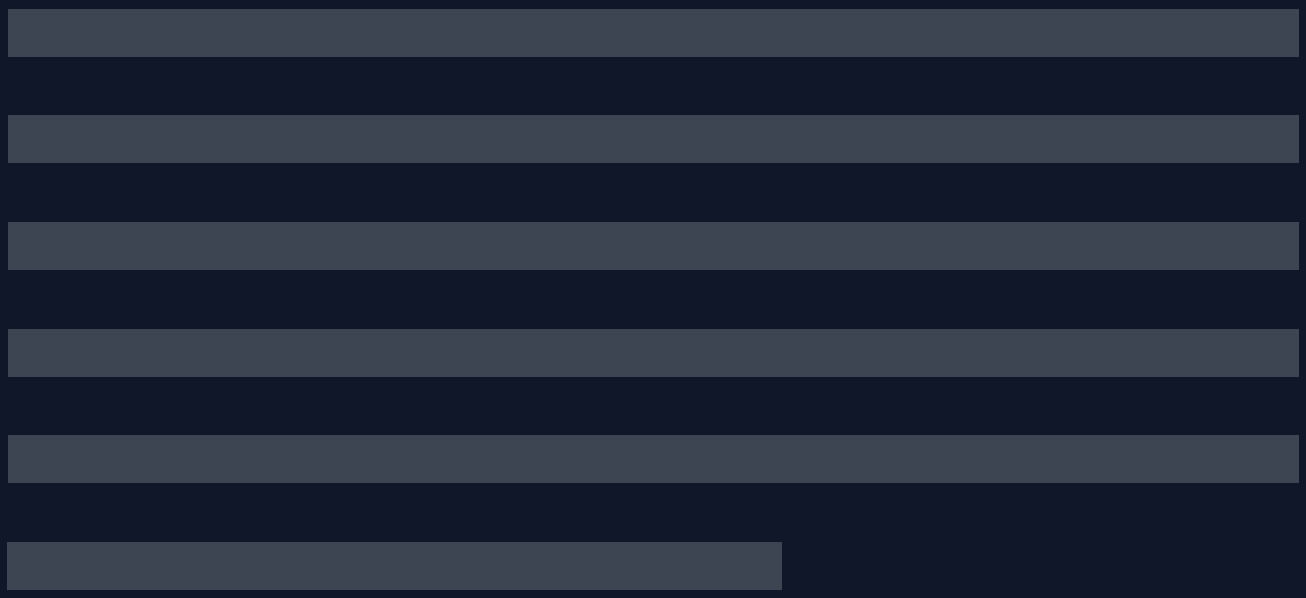
3. Goals and Success Metrics

Your three outcomes, in priority order:

1. Use AI code tools to review and improve your own pull requests before merging. The metric is five PRs reviewed with AI before submitting. This is the highest-leverage habit in the plan because it compounds — every project you work on for the rest of your life will have PRs, and a reliable

The rest of this plan is redacted.

You are seeing the first two pages. The full plan is yours when you create one.



Create yours at lighthouse.dom.net