COURSE SYLLABUS · COURSE 03 — PRINCIPLED AI CODING MASTERY

Course 03 — Principled AI Coding Mastery

A 6-modules CyberG7 system — research-backed, build-as-you-go.

THE COURSE AT A GLANCE

The only AI-coding course framed as a working discipline (P^8) rather than a prompt library — each principle is a named behavioral inversion of a documented 2026 failure mode, drilled on real repos so engineers ship 3-5x faster on code they can defend in a postmortem.

Who This Is For

The only AI-coding course framed as a working discipline (P^8) rather than a prompt library — each principle is a named behavioral inversion of a documented 2026 failure mode, drilled on real repos so engineers ship 3-5x faster on code they can defend in a postmortem.

This course is built for:

- Junior-to-mid software engineers who use AI assistants daily but firefight the output as much as they write code
- Mid-to-senior engineers who are already fast with AI tools but can feel comprehension debt accumulating in their codebase
- Senior+ engineers and tech leads who need team norms and defensible inversions, not prompting help
- Engineering managers and platform owners responsible for AI-coding hygiene, code quality, and audit readiness

You'll feel right at home if any of these sound familiar:

- AI output that's locally correct but globally wrong — wiring overhead, duplicated logic, hallucinated imports
- Comprehension debt: merged code nobody on the team can explain, surfacing during incidents and audits
- Velocity illusion — feeling faster with AI while quality, reuse, and review trust quietly erode
- Security exposure from over-trusted agents: credential leaks, unsanitized input, unexpected code execution

What You'll Build

By the end, you won't just understand the ideas — you'll have assembled a working system, module by module. Across the course you'll develop:

- Why Single-File Thinking Is a Trap
- The 5-File Context Bundle
- Running Multi-File Edits Across Tools
- The 4-Lens Accept Gate
- Glossary
- The Shape of an AI Hallucination
- Building a Verification Harness
- Grounding, Retrieval, and Truthfulness
- Stale Knowledge Is a Hallucination Too

Course Outline

Module 1 · Multi-File Mind: Orchestrate AI Across the Whole Codebase

Most engineers prompt AI one function at a time, a habit inherited from an era of tiny context windows. That single-file stance is the root cause of logic duplication, hallucinated imports, and answers that are correct in isolation but wrong in the system. Principle 1 (P^1) makes the file-bundle, not the function, the unit of AI interaction. The constraint is no longer the model's memory — it is your discipline in deciding what to put in front of it.

Key themes:

- Why Single-File Thinking Is a Trap
- The 5-File Context Bundle
- Running Multi-File Edits Across Tools

Module 2 · Know Your IDKs: Hallucination Defense Before Code Hits Main

AI assistants state fabricated APIs and stale knowledge with the same fluent confidence they use for correct answers. Principle 2 (P^2) is about knowing your I-Don't-Knows: detecting, naming, and mitigating hallucinations before they land in your repo. The defense is not vigilance by willpower — it is a verification harness that runs automatically, before human review.

Key themes:

- The Shape of an AI Hallucination
- Building a Verification Harness
- Grounding, Retrieval, and Truthfulness

Module 3 · Anti-Pattern Avoidance: The 7 AI-Coding Failure Modes and Their Inversions

Every AI-coding disaster traces back to one of seven recognizable anti-patterns. Principle 3 (P^3) is about naming all seven by their behavioral signature, installing the corresponding inversions as default workflow reflexes, and triaging existing AI-generated code into keep, rewrite, or delete. The 2026 OWASP Top 10 for Agentic Applications now lists vibe-coded systems as a primary attack surface — these are not style preferences, they are risk controls.

Key themes:

- The Seven Anti-Patterns
- Why Now: Comprehension Debt and Ownership Debt
- Triaging Existing AI Code

Module 4 · Spec-First Development: 10-Line Specs That Beat 100-Line Prompts

The single highest-leverage technique in AI-assisted coding is writing a short spec before you prompt — output quality follows input quality. Principle 4 (P^4) replaces the improvised one-line prompt with a 10-30 line behavioral spec that encodes constraints, non-goals, and edge cases the model will honor, then converts that spec into reviewable, atomic, test-backed patches. The cost is five minutes of writing; the payoff is typically 30-60 minutes of saved iteration, a 6-12x return on routine features.

Key themes:

- Why a Spec Beats a Prompt
- Spec Anatomy: Six Sections
- Non-Goals Are the Secret Weapon

Module 5 · Tooling Mastery: The Right Tool for the Shape of the Task

The leverage in AI coding is not in any one tool — it is in matching the tool to the shape of the task and configuring the automation seam so the code can write itself, principally. Principle 5 (P^5) covers Aider for headless work, Cursor for interactive editing, and Claude Code for agentic loops, plus the graduated-autonomy boundaries that let you scale safely toward production.

Key themes:

- Match Tool to Task
- Configuring the Automation Seam
- Graduated Autonomy

Module 6 · Compute Commander: The Director Pattern, AI Workflows, and the P^8 Synthesis

The capstone fuses the final three principles into a single working stance. Principle 6 (P^6 — the Director Pattern) stops you being the typist and makes you the director of compute. Principle 7 (P^7 — AI Developer Workflows) composes tools into end-to-end loops that run on git, CI, or cron triggers. Principle 8 (P^8 — Compute Commander) is the synthesis: wielding AI as principled leverage across teams and time. The leverage was never in typing speed — it is in the stance.

Key themes:

- P^6: The Director Pattern
- P^7: AI Developer Workflows (ADWs)
- P^8: Compute Commander — The Synthesis

Outcomes

Complete the course and you'll be able to:

- Before prompting, open the ticket and list the five files it will touch — routes, model, persistence, pattern test, utilities.

- Add "cite the file path for every symbol you use" to your standard code prompt; reject uncited imports on sight.
- Memorize the seven anti-patterns by behavioral signature so you can name them in code review on the spot.
- Replace your next one-line prompt with a 10-30 line spec covering intent, inputs, outputs, edge cases, non-goals, and a pattern reference.
- Map each task to a tool by shape: Aider for headless/batch, Cursor for interactive, Claude Code for agentic loops.
- Run a manual three-role session today — spec writer, planner, implementer — handing off through committed files.

ENROLL

Enroll in P[8] Start *Course 03 — Principled AI Coding Mastery* today — the full module-by-module system lives at https://principled-ai-coding-mastery-edu.cyberg7.com.sg.