APPLIED AND GENERATIVE AI

Advanced AI Coding Agent Techniques

Level: Practitioner • 2 days (expandable to 3) • Virtual, In-person

Overview

Most developers who use an AI coding assistant have learned to prompt it for a function, accept a suggestion, and move on. This course is about everything past that point: the techniques that let you hand a coding agent substantial, multi-step work and trust the result.

This is a hands-on, practitioner course. It is for people who already use these tools daily. Rather than survey every feature of every product, it goes deep on a small number of skills that compound. We begin with a clear mental model of how a coding agent actually works, because every advanced technique is easier once you understand what the agent can see and why it succeeds or fails. From there we layer up one step at a time: engineering the agent's context, encoding repeatable work, steering large multi-file changes, delegating to subagents, and finally the verification habits that make agent-driven development trustworthy. Every module ends with hands-on practice, and later modules build directly on earlier ones.

Who Should Attend

- Developers who already use an AI coding assistant day to day and want to work at a higher level
- Technical leads standardizing how their team uses coding agents
- Engineers moving from occasional prompting toward delegating real work to agents

This is not an introduction to AI coding tools. Learners who have never used one should take *Foundations of AI Coding Assistants* first.

Prerequisites

- Comfortable using an AI coding assistant (Claude Code, GitHub Copilot, Cursor, or similar) for everyday tasks
- Working proficiency in at least one programming language
- Comfortable with git and running a project locally

What You Will Learn

- Explain how a coding agent works: the plan, act, and observe loop, its context, and its tools
- Engineer an agent's context so it produces reliable results in a real codebase
- Encode repeatable tasks as custom commands and reusable workflows
- Plan and steer large, multi-file changes without losing control
- Delegate subtasks to subagents and coordinate more than one agent
- Apply verification and review practices that make agent output trustworthy
- Recognize common agent failure modes and recover from them

Course Outline

Day one: understanding and controlling the agent

- How Coding Agents Actually Work
  - From autocomplete to agent: the plan, act, and observe loop
  - What the agent can and cannot see: context, tools, and memory
  - Why agents succeed, and the common reasons they fail
  - Lab: watch an agent complete a task and narrate what it is doing and why
- Context Engineering
  - Curating what the agent sees: the right files, instructions, and conventions
  - Project-level instructions and persistent memory
  - The cost of too much context and too little; keeping it focused
  - Lab: set up an agent to work effectively in an unfamiliar repository
- Reusable Workflows and Custom Commands
  - Encoding a repeatable task so you stop re-explaining it
  - Building, testing, and sharing custom commands with a team
  - Lab: turn a routine task from your own work into a reusable command

Day two: delegation and trust

- Steering Large Changes
  - Plan first: getting and reviewing a plan before any code is written
  - Working in small, reviewable increments and keeping the agent on track
  - Driving a multi-file refactor without losing the thread
  - Lab: take a multi-file change from plan to finished, reviewed result
- Subagents and Agent Teams
  - Delegating a subtask to a subagent: when it helps and when it adds noise
  - Coordinating more than one agent and dividing the work sensibly
  - Lab: use a subagent to handle part of a larger task in parallel
- Trustworthy Agent-Driven Development
  - Verification as the habit that makes speed safe: tests and human review as guardrails
  - Spotting failure modes early: looping, drift, and overconfidence, and resetting context
  - Team norms: where agents are and are not allowed to work unattended
  - Lab: add guardrails to an agent workflow, then diagnose and recover a stuck agent

Extended Version

The three-day version keeps the same gradient and adds room to go deeper and to practice on a fuller piece of work:

- Spec-driven development: writing a clear specification the agent implements against
- Deeper multi-agent orchestration patterns
- Bringing agents into CI and larger automation, with appropriate limits
- A capstone that carries one real feature from specification to a reviewed, tested implementation