APPLIED AND GENERATIVE AI

Foundations of AI Coding Assistants (GitHub Copilot and Claude Code)

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

Overview

AI coding assistants have moved from novelty to standard equipment faster than any developer tool in memory, and most developers are now expected to use one. Yet the difference between developers who get real leverage from these tools and those who fight them is not talent, it is technique: knowing what the tool can actually see, how to ask, and how to judge what comes back. Used badly, an assistant produces plausible code you cannot trust; used well, it changes what one developer can do in a day.

This is a hands-on, foundation course. It builds a working practice from the ground up: first a plain mental model of what these tools are and why they behave the way they do, then real daily work in GitHub Copilot and Claude Code, then the skills that make the difference, asking well, applying the tools beyond code generation, and reviewing output with appropriate skepticism. The course deliberately focuses on a small set of habits that transfer across tools rather than touring every feature. Every module includes a lab, and each module builds on the one before it.

Who Should Attend

- Developers who have not yet used an AI coding assistant, or have only dabbled
- Teams adopting GitHub Copilot or Claude Code and wanting a consistent baseline
- Engineering managers who want first-hand understanding of how these tools change daily work

  Developers already using these tools daily should consider *Advanced AI Coding Agent Techniques* instead.

Prerequisites

- Working proficiency in at least one programming language
- Comfort with an editor or IDE and basic use of git
- No prior experience with AI tools is required

What You Will Learn

- Explain what an AI coding assistant is, what it can see, and why it sometimes gets things wrong
- Use GitHub Copilot effectively for completions, chat, and everyday coding tasks
- Use Claude Code to make and review changes to a real project from the terminal
- Write requests that give the assistant the context it needs to do good work
- Apply assistants beyond code generation: tests, debugging, documentation, and understanding unfamiliar code
- Review AI-generated code with sound judgment and safe habits

Course Outline

Day one: getting productive

- What an AI Coding Assistant Really Is
  - How these tools work, in plain terms: prediction, context, and why wording matters
  - The spectrum of assistance: autocomplete, chat, and agents that take actions
  - What the assistant can and cannot see, and where hallucination comes from
  - Lab: give the same task to Copilot and Claude Code and compare what each did and why
- Everyday Coding with GitHub Copilot
  - Inline suggestions: accepting, rejecting, and steering them
  - Copilot Chat: asking about code, generating functions, and making edits
  - Small habits that raise suggestion quality: names, comments, and open files
  - Lab: build a small feature end to end with Copilot doing the typing
- Everyday Coding with Claude Code
  - Working with an agent in the terminal: describing a change and letting it work
  - Reviewing what the agent did: reading diffs before accepting them
  - Asking Claude Code to explain, fix, and iterate
  - Lab: use Claude Code to add a feature to an existing project and review its diff

Day two: working well with the tools

- Asking Well: Context Is Everything
  - Why the same request succeeds or fails depending on what the tool can see
  - Being specific: constraints, examples, and expected behavior
  - Iterating instead of accepting the first answer
  - Lab: take a vague request that produced bad code and refine it until the output is right
- Beyond Writing Code
  - Generating and improving tests
  - Debugging with an assistant: explaining errors and narrowing causes
  - Understanding unfamiliar code and generating documentation
  - Lab: use an assistant to diagnose a failing test and explain a legacy module
- Judgment, Review, and Safe Habits
  - Trust but verify: reading AI code as carefully as a junior developer's pull request
  - Common failure patterns: confident nonsense, subtle bugs, and outdated APIs
  - Security, licensing, and sensitive data, and where team norms belong
  - Lab: review a set of AI-generated changes, find the planted defects, and fix them

Extended Version

The three-day version keeps the same gradient and adds depth and a fuller piece of work:

- Customizing the tools: project instructions, custom configuration, and team conventions
- A first look at delegating larger tasks, as a bridge to *Advanced AI Coding Agent Techniques*
- Assistant-supported refactoring and code quality work on a real codebase
- A capstone that builds and reviews a complete small feature with an assistant doing most of the work