www.logicalimagination.com
800.657.1494

AI Foundations for Software Developers

Course #: AI-300 **Duration:** 1 day

Prerequisites

Professional experience in software development and familiarity with modern development workflows. No prior AI or machine learning experience is required.

Details

This course provides software developers with a practical foundation for understanding how AI behaves inside modern development tools. Rather than focusing on machine learning theory, the course explains how large language models generate code, where they fail, and how those failures differ from traditional software defects.

Participants learn how AI-assisted coding changes responsibility, review practices, and professional judgment. The course prepares developers to use tools like GitHub Copilot effectively by understanding their strengths, limitations, and risks before relying on them in real projects.

After attending this course, students should be able to:

Explain how AI code generation differs from traditional programming tools
Recognize common failure modes in AI-generated code
Evaluate AI-generated code critically rather than accepting it at face value
Understand how AI affects software quality, maintainability, and security
Apply professional judgment when using AI in development workflows

This course is designed for software developers who are beginning to use AI-powered development tools and want to do so responsibly and professionally. This course assumes participants can read, write, and reason about code, but it does not require prior AI knowledge.

Software Needed

No AI tools, programming environments, or software installations are required for this course, though participants should be comfortable reading and reasoning about code.

Outline

AI Foundations for Software Developers

- **Why AI Changes Software Development**
  - From autocomplete to probabilistic code generation
  - How AI differs from compilers, linters, and IDE tools
  - Why AI-generated code "looks right" even when it's wrong
  - Shifts in responsibility and accountability
- **How AI Generates Code**

- High-level overview of large language models
  - Tokens, context windows, and pattern completion
  - Why AI does not “understand” code
  - What training data means for code generation
- **Strengths of AI-Assisted Coding**
  - Accelerating routine and boilerplate code
  - Exploring unfamiliar APIs and libraries
  - Generating examples and scaffolding
  - Supporting learning and experimentation
- **Failure Modes in AI-Generated Code**
  - Logical errors and incorrect assumptions
  - Edge cases and missing constraints
  - Outdated or deprecated APIs
  - Overconfidence and fabricated explanations
- **Reading and Reviewing AI-Generated Code**
  - Treating AI output as a draft, not an answer
  - Techniques for validating correctness
  - Recognizing “convincing nonsense”
  - Knowing when to rewrite instead of patch
- **AI and Software Quality**
  - Impact on readability and maintainability
  - Consistency with existing codebases
  - Avoiding accidental architectural drift
  - Long-term cost of unchecked AI usage
- **Security and Risk Considerations**
  - Common insecure patterns introduced by AI
  - Dependency and licensing risks
  - Data leakage and prompt misuse
  - Why AI-generated code must be reviewed for security
- **Professional Responsibility and Ethics**
  - Accountability for AI-assisted code
  - Attribution and intellectual property considerations
  - Using AI responsibly in team environments
  - Setting personal and team boundaries
- **When Not to Use AI**
  - High-risk or safety-critical code
  - Performance-sensitive systems
  - Security and cryptography
  - Regulatory and compliance constraints
- **Preparing for AI-Augmented Development**
  - Developing healthy skepticism and discipline
  - Integrating AI into professional workflows
  - Preparing for tool-specific training (e.g., Copilot)
  - Next steps in AI-augmented software development