Faysal Aziz

Modernizing Software Development using AI

Empower your Python toolkit with prompt engineering, agentic automation & real execution - 3 days

FA-0326 · 3 days

Agentic & Generative AI · Software Development

Introduction

The days of coding in isolation are over. AI has shifted from helpful sidekick to essential teammate, fundamentally changing how we build, ship, and scale software. This isn't another surface-level overview—we're diving deep into the intersection where battle-tested development practices meet the raw power of modern AI tooling.

You'll master prompt engineering that actually works, supercharge your workflow with advanced Copilot techniques, and build autonomous agents that handle real work. We're talking production-ready skills: API integration, deployment pipelines, and the kind of AI-enhanced development that companies are scrambling to adopt.

Skip the theory lectures and marketing hype. This is about getting your hands dirty with the tools and techniques that are reshaping our industry right now.

Learning outcomes

By the end of this course, learners will:

- Design effective Python prompts for AI coding assistants
- Compare coding performance and workflows across ChatGPT, Gemini, Claude, and Manus
- Build and integrate agentic AI workflows using tools like Manus with task decomposition
- Implement Python-based automated code generation, execution & MCP integration

- Set up backend API interactions to support AI-driven modules
- Deploy small AI-powered services end-to-end using real tooling

Prerequisites

- Comfortable with intermediate to advanced Python
- Familiarity with Git/GitHub workflows
- Basic understanding of REST APIs
- Willingness to explore new AI-powered tools and SDKs

Training outline

1. 1. AI-Enhanced Coding: Prompt Engineering & Copilot

Fundamentals of prompt patterns for code generation

- Context framing, function signatures, expected outputs

- Prompt sharpening for debugging, refactoring, test generation
- **Hands-on lab**: Refine prompts to solve live coding tasks

GitHub Copilot integration

- Deep prompts within VS Code: "Implement BFS", "Write pytest tests"
- Compare outputs vs. manual coding
- Best practices: code suggestions, acceptance, overrides

2. 2. Model Comparison: ChatGPT vs Gemini vs Claude vs Manus

Performance benchmarks from 2025 data:

- **ChatGPT**: Swiss-army tool—fast and versatile
- **Gemini 2.5 Pro/Flash**: Large context, cost-effective
- **Claude 4 Sonnet/Opus**: Highest accuracy in code tasks
- **Manus**: Full-stack agentic execution

Side-by-side live demos:

- Write a utility function, generate tests, catch bugs
- Measure speed, correctness, cost

3. 3. Agentic Workflows: Task Decomposition & Autonomous Tools

- Architectures of agentic AI (task planners, executors, sandboxes)
- Manus in action: multi-agent orchestration
- **Hands-on lab**: Decompose a task into sub-tasks and orchestrate via a Python agent
- Introduce MCP concept, libraries, server patterns

4. 4. MCP: Model Context Protocol for Code Execution

- What is MCP? Standardizing LLM-to-tool links

  Building MCP servers in Python
  - Registering functions with @mcp.tool()

  Executing LLM-generated code in safe environments
  - Use cases: file read/write, API access, computation

5. 5. Building a Python MCP Code Executor

- Walkthrough: mcp_code_executor GitHub project
- Support for incremental code execution, dependency installs
- Lab: Build a mini MCP executor, connect ChatGPT/Claude agents

6. 6. Autonomy at Work: End-to-End Code Generation & Execution

- Full pipeline from prompt → code → execution → output
- Safety strategies: sandboxing, test wrappers, execution logs
- **Hands-on**: Build an AI-driven mini-service (e.g., currency converter or basic scraper)

7. 7. Backend Integration: API Calls & Deployment

- Crafting effective API-call prompts
- Embed AI-generated code in Flask/FastAPI backends
- Deploy via Docker or simple managed services
- Testing & validation workflows

8. 8. Automating with MCP Agents & GitHub

- GitHub Copilot + MCP: commit scripts, CI interactions
- Scheduled agentic tasks: run tests, lint, deploy
- Agent logs and iterative debugging

9. 9. Putting It All Together: Final Project

- Idea: A Python-based AI agent that reads a spec, writes code, executes tests, logs results, and deploys
- Stepwise task decomposition, MCP integration, model selection - Wrap-up: Tech stack recap, tool pros/cons, recommended AI workflows

---

Instructor Profile

Your instructor brings **30+ years** of real-world software engineering experience, pairing foundational architecture and systems insight with the latest in AI-driven development. Expect industry-demanded skills, not academic hypotheticals.

© 2026 Faysal Aziz · Kuala Lumpur