www.logicalimagination.com
800.657.1494

Building MCP-Based AI Systems with Python

Course #: AI-203 **Duration:** 3 days

Prerequisites

Completion of AI Foundations & Risk for IT Professionals, Designing Reliable AI Workflows & Interactions, and AI Architecture & Agentic Systems, or equivalent experience designing AI workflows and agentic system architectures. Strong working knowledge of Python is required.

Details

This course teaches participants how to design, build, and deploy AI systems using the **Model Context Protocol (MCP)** and Python. Rather than building fragile, prompt-driven demos, participants learn how to expose structured tools and backend services that AI systems can use safely and reliably.

Participants implement an MCP server, define schema-driven tools, integrate data access layers, and apply best practices for validation, error handling, and deployment. The course emphasizes **structure, safety, and maintainability**, enabling teams to move from experimental AI use to production-ready systems.

After attending this course, students should be able to:

Explain the role of MCP in agentic AI systems
Build an MCP server using Python
Define schema-driven tools with clear input and output contracts
Integrate backend data sources and services safely
Apply validation, error handling, and idempotency strategies
Connect MCP servers to AI clients such as Claude Desktop
Package, version, and distribute MCP-based systems

This course is designed for technical professionals responsible for implementing AI systems that integrate safely and reliably with real organizational data and services. This course is hands-on and implementation-focused and assumes prior experience with Python development.

Software Needed

Participants must have a laptop or desktop computer (Windows, macOS, or Linux) with Python 3.10 or later installed, the ability to create virtual environments, install Python packages, and run local development servers. Access to an MCP-compatible AI client (such as Claude Desktop) is required. Full system requirements are provided prior to the course.

Outline

Building MCP-Based AI Systems with Python

- **MCP and Agentic Architecture Fundamentals**
  - Why MCP exists

    - LLMs as system actors rather than chat interfaces
    - How MCP changes AI system design
    - Real-world examples of tool-using agents

- **MCP Core Components in Depth**
    - MCP servers, tools, and resources
    - Schema-driven interfaces
    - JSON Schema vs dynamic data structures
    - Why explicit schemas matter in Python systems

- **Structuring Data with Pydantic**
    - Using Pydantic for schema definition and validation
    - Enforcing input and output contracts
    - Handling optional and constrained fields
    - Preventing runtime ambiguity

- **Setting Up the MCP Server (Python)**
    - MCP server lifecycle
    - Virtual environments and dependency management
    - Minimal, maintainable project structure
    - Logging and debugging strategies

- **Designing and Implementing MCP Tools**
    - Tool functions vs tool classes
    - Mapping tools to backend capabilities
    - Input validation and output enforcement
    - Exception handling and error reporting
    - Idempotency in dynamic systems

- **Building a Backend Domain Model**
    - Domain modeling in Python
    - Separating business logic from MCP interfaces
    - Designing for testability and reuse

- **Data Access and Persistence Options**
    - In-memory data structures
    - SQLite for lightweight persistence
    - SQLAlchemy integration
    - Asynchronous data access considerations

- **Exposing CRUD Operations via MCP**
    - Tool vs resource design decisions
    - Read-only vs mutating operations
    - Guardrails for destructive actions
    - Aligning CRUD exposure with governance

- **Prompting and Sampling for Reliable Agents**
    - Prompt definitions as files or constants
    - Sampling strategies and helper functions
    - Managing multi-step agent workflows
    - Reducing variability and unintended behavior

- **Integrating with AI Clients**
    - Connecting MCP servers to Claude Desktop
    - CLI execution and configuration
    - Environment variables and secrets management
    - Debugging runtime behavior

- **Testing, Safety, and Reliability**
    - Testing MCP tools and schemas
    - Failure injection and edge cases
    - Observability and logging
    - Safe iteration during development

- **Packaging, Distribution, and Versioning**
    - Python packaging with `pyproject.toml`
    - Semantic versioning strategies
    - Publishing to PyPI or internal repositories
    - Managing breaking changes

- **Operational Considerations**
  - Deployment models
  - Configuration management
  - Monitoring and maintenance
  - Supporting MCP-based systems over time
- **Summary and Next Steps**
  - Key implementation principles
  - From prototype to production
  - Aligning MCP systems with governance
  - Preparing for continued evolution of AI systems