

COURSE CURRICULUM

Production-Ready Agentic AI

with Python, LangChain, LangGraph, MCP and LangSmith

4 DAYS | 32 HOURS | INSTRUCTOR-LED | HANDS-ON

From model integration to tool-enabled, observable and controlled enterprise agents

Table of Contents

Course Snapshot

Course Name: Production-Ready Agentic AI with Python, LangChain, LangGraph, MCP and LangSmith

Duration: 4 Days | 32 Hours

Delivery: Instructor-led with guided demonstrations and practical labs

Audience: Developers, solution engineers, AI engineers and technical professionals building enterprise AI applications

Course Overview

A focused, build-first course that takes participants from LLM integration to production-oriented AI agents. Participants use LangChain for the agent harness, LangGraph for controlled workflows, MCP for interoperable tools and context, and LangSmith for tracing and evaluation.

Course Objectives

- Connect Python applications to an Azure- or AWS-hosted LLM and use LangChain core components.
- Build tool-enabled agents with structured output, context, state, memory, and enterprise knowledge.
- Orchestrate reliable workflows with LangGraph, middleware, guardrails, and human approval.
- Create an MCP server and make MCP capabilities available as tools to an agent.
- Secure agents with identity and access controls, governance guardrails, and an agentic security attack-and-defense lab.
- Trace, test, and evaluate agent behavior using LangSmith and production-readiness practices.

Prerequisites

- Basic knowledge of Python programming.
- Familiarity with REST APIs and JSON is helpful.
- Basic understanding of Generative AI is beneficial but not mandatory.

Platform and Access Requirements

- Access to Python, with permission to install and run the required packages.
- A suitable code editor and internet or network access for the labs.
- A client-provided Azure or AWS LLM endpoint, API key, or equivalent credentials.
- Access to LangSmith is recommended for the observability and evaluation exercises.

Day 1 | Models, LangChain Foundations and Tools

Outcome: Participants build a structured LLM application and convert Python functions into tools for a basic LangChain agent.

MODULE 01

Generative AI and Agentic AI Foundations

Establish the role of LLMs, tools and agents in modern AI applications.

- Explain LLM requests, tokens, context windows, hallucinations, and common limitations in practical terms.
- Distinguish a direct LLM application, a deterministic workflow and an autonomous agent.
- Introduce the agent loop as model reasoning supported by prompts, tools, and runtime controls.

Hands-on: Call the client-provided LLM from Python and inspect the request-response cycle.

MODULE 02

Python Essentials for AI Applications

Prepare the minimum Python foundation needed for the remaining course.

- Use functions, classes, type hints, JSON, environment variables, exceptions, and asynchronous calls in an AI project.
- Organize a simple Python application and manage credentials through environment configuration.
- Create reusable Python functions that can later be exposed as agent tools.

Hands-on: Set up the course project and create reusable business functions.

MODULE 03

LangChain Core Components

Introduce LangChain in the same progression used to assemble an agent harness.

- Work with chat models, messages, prompt templates, streaming, and provider configuration.
- Return validated structured output through schemas rather than relying only on free-form text.
- Understand how LangChain, LangGraph, and LangSmith serve different parts of the agent lifecycle.

Hands-on: Build a message-based LLM application that returns validated structured output.

MODULE 04

Tools and the First LangChain Agent

Move from model invocation to a model that can select and use tools.

- Define focused tools with clear descriptions, input schemas, outputs, and error handling.
- Build an agent using a model, system prompt, tools, and response format.
- Observe the model-tool-result loop, termination behavior, and execution limits.

Hands-on: Create a tool-enabled customer service agent with two or three Python tools.

Day 2 | Context, RAG, State, Memory and Middleware

Outcome: Participants ground an agent in enterprise knowledge and control its context and behaviour through state, memory and middleware.

MODULE 05

Context Engineering and Agent State

Teach participants to provide the right information to the model at the right time.

- Differentiate among messages, runtime context, custom agent state, and external knowledge.
- Use dynamic prompts and selective tool availability to shape agent behavior.
- Control context growth and keep business data separate from conversation history.

Hands-on: Add customer or session context and a custom state field to the agent.

MODULE 06

Retrieval-Augmented Generation

Build a compact RAG pipeline for grounded enterprise responses.

- Ingest, clean, chunk, and index documents with useful metadata.
- Use embeddings, vector search, and retrievers to supply relevant context to the model.
- Compare RAG with tools and memory, and address weak or missing retrieval results.

Hands-on: Build a company policy assistant and connect retrieval to the agent.

MODULE 07

Short-Term and Long-Term Memory

Introduce memory as controlled persistence rather than unlimited conversation history.

- Apply thread-level conversation memory and resume an earlier interaction.
- Distinguish short-term state, long-term user information and document retrieval.
- Use summarization or retention rules to manage expanding context responsibly.

Hands-on: Add session memory and retain one selected user preference across interactions.

MODULE 08

Agent Middleware, Governance and Guardrails

Use middleware to control and observe events throughout the agent loop.

- Apply hooks around agent, model, and tool execution for logging, prompt changes, and validation.
- Use retries, fallbacks, call limits, context summarization, and tool-selection controls.
- Introduce human approval and permission checks before sensitive actions.
- Compare deterministic (rule-based) and model-based guardrails, and determine where each fits.
- Apply built-in PII-detection middleware with redaction, masking, hashing, and blocking strategies.
- Place guardrails before the agent starts, after it completes, and around model and tool calls.

Hands-on: Add retry, approval and PII-redaction middleware to a sensitive tool call.

Day 3 | LangGraph, Controlled Workflows and MCP

Outcome: Participants build a stateful workflow, create an MCP server and expose MCP capabilities to an agent.

MODULE 09

LangGraph Fundamentals

Introduce LangGraph where explicit workflow control is required beyond a standard agent loop.

- Model a workflow with state, nodes, edges, and conditional routing.
- Combine deterministic business steps with agentic decisions and tool execution.
- Use streaming, retry paths, and subflows to make execution easier to monitor and control.

Hands-on: Create a small stateful workflow with conditional routing and a tool node.

MODULE 10

Persistence and Human-in-the-Loop

Make long-running or sensitive workflows recoverable and reviewable.

- Checkpoint workflow state and resume execution with a thread or session identifier.
- Interrupt execution before a sensitive action and continue after review.
- Use approval, editing, rejection, and response decisions as governance controls for tool calls.
- Design failure paths that avoid repeating non-idempotent business operations.

Hands-on: Pause a workflow for approval, modify the decision if needed and resume execution.

MODULE 11

MCP Architecture and Primitives

Explain MCP as a standard connection between AI applications and external capabilities.

- Understand MCP host, client, and server roles, as well as the connection lifecycle and security boundary.
- Use MCP tools for actions, resources for contextual data, and prompts for reusable interaction templates.
- Compare MCP with direct REST integration, local function calling, and native agent tools.

Hands-on: Inspect a sample MCP server and invoke an exposed capability from a client.

MODULE 12

Build and Connect a Python MCP Server

Create an MCP server and make its capabilities available to the LangChain agent.

- Define MCP tools, resources, and prompts with validated inputs, structured results, and clear errors.
- Run the server locally or through an approved remote connection, and test capability discovery.
- Discover MCP tools through the LangChain MCP integration and add them to the agent harness.
- Secure MCP connections with bearer-token or OAuth authentication and per-server or per-user credentials.

Hands-on: Build a customer operations MCP server and connect its tools to the course agent.

Day 4 | Multi-Agent Patterns, Observability and Capstone

Outcome: Participants trace and evaluate agent behaviour, apply identity, governance and agentic security controls, and complete an integrated capstone solution.

MODULE 13

Multi-Agent Design Patterns

Use multiple agents only when specialization or context separation provides clear value.

- Compare a single capable agent with supervisor, router, specialist, and handoff patterns.
- Share only the required state and tools with each specialist to limit costs and complexity.
- Review when a multi-agent design should be replaced by a simpler workflow.

Hands-on: Create a supervisor that routes a request to two focused specialist agents.

MODULE 14

Observability and Evaluation with LangSmith

Use execution evidence to understand, test and improve the agent.

- Trace model calls, tool calls, retrieval steps, state transitions, latency, and errors.
- Inspect an agent trajectory to identify incorrect tool selection or weak retrieval.
- Create compact evaluation cases for response quality, grounding, tool use, and task completion.

Hands-on: Trace the course agent and compare two runs using a small evaluation set.

MODULE 15

Enterprise Readiness: Agent Identity and Access Management

Apply the essential controls required before moving an agent beyond the prototype stage.

- Address authentication, authorization, secrets, tool permissions, and sensitive data handling.
- Use validation, timeouts, retries, fallbacks, audit information, and cost controls.
- Organize the solution around model access, orchestration, knowledge, MCP capabilities, and operations.
- Distinguish authentication (who the caller is) from authorization (what the caller may do).
- Validate identity-provider tokens and apply resource-level access rules to agent threads and data.
- Pass the user's identity to MCP servers so that tools run with the caller's permissions rather than those of a shared superuser.

Hands-on: Add authentication and owner-only access to the course agent, then perform a production-readiness review.

MODULE 16

Agentic Security Half-Day Module (ASMS)

Understand how agents are attacked and apply layered defences using LangChain and LangGraph controls.

1. Introduction: The Paradigm Shift

- Anatomy of an agent: model, prompts, tools, context, memory, retrieval and identity.
- Traditional AppSec versus agentic security: why model-selected actions create new trust boundaries.

2. The Attack Surface

- Prompt injection: direct (user input) and indirect (documents, tool results, and web content).
- Tool abuse and privilege escalation through overly permissive tools.
- Memory and RAG poisoning through untrusted stored or retrieved content.

3. Defence Guidelines

- Input and output guardrails, with validation through middleware and structured output.
- Tool-level least privilege with tool permissions and tool call limits.
- Human-in-the-loop (HITL) enforcement for sensitive or irreversible actions.
- Memory and context isolation with per-user threads and namespaced long-term memory.

4. Hands-On Lab

Hands-on: Phase 1 – Exploit: run an indirect prompt injection that triggers data exfiltration in a sandboxed agent with synthetic data.

Hands-on: Phase 2 – Defend: patch the agent with HITL middleware and output parsers, then verify the fix in LangSmith traces.

MODULE 17

Capstone Project: Enterprise Customer Service Agent

Integrate the major course components into one demonstrable solution.

- Build an agent that answers policy questions, checks customer or order information and performs an approved business action.
- Use LangChain for the agent harness, LangGraph for routing and approval, and MCP for external tools or context.
- Add memory, middleware, grounded retrieval, tracing, evaluation, authentication, and security guardrails.

Hands-on: Complete and demonstrate the end-to-end agent, including one MCP call, one approval step, one blocked unsafe action and one trace review.