Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Building AI Agents with Python and MCP

Class Duration

35 hours of live training delivered over 5 days.

Student Prerequisites

- Professional Python development experience
- Familiarity with LLM APIs: prompts, tool/function calling, and structured outputs (see *LLM Application Development with Python*)
- Comfort with async programming (`asyncio`) and Git
- No prior agent framework experience required

Target Audience

Python developers ready to move beyond single request/response LLM calls to autonomous, multi-step systems. Ideal for engineers building internal agents that plan work, call tools, and act on company data and APIs — and for platform teams standardizing how agents connect to internal systems via the Model Context Protocol. Includes a dedicated module on running open-weight models locally for organizations with data residency, privacy, or AI usage restrictions.

Description

Agents are the defining LLM application pattern of 2026: systems that take a goal, plan steps, call tools, observe results, and iterate until done. This five-day intensive teaches Python developers to build them for production. Day one builds an agent from scratch — a plain-Python agent loop with tool dispatch, working memory, and stop conditions — so students understand exactly what frameworks abstract away. Day two introduces the frameworks: LangGraph for graph-structured workflows with state, checkpoints, and human-in-the-loop interrupts, and Pydantic AI for type-safe, dependency-injected agents, with guidance on when each fits. Day three is the Model Context Protocol: building MCP servers in Python that expose internal tools, data sources, and APIs; consuming them from agents and from coding assistants like Claude Code; and designing tool interfaces that models use reliably. Day four covers multi-agent orchestration — planner/worker architectures, hand-offs, and failure recovery — plus the evaluation discipline

agents demand: trajectory evals, tool-call accuracy, and regression suites. Day five addresses production realities: security (prompt injection, tool sandboxing, least-privilege credentials), observability and cost control, deployment, and a module on running open-weight models locally with Ollama and vLLM for restricted environments where data cannot leave the network. Students finish with a capstone agent built end to end.

Learning Outcomes

- Explain the agent loop — plan, act, observe, iterate — and implement it from scratch in Python.
- Design tool interfaces, schemas, and error contracts that models invoke reliably.
- Build stateful agent workflows in LangGraph with checkpoints and human-in-the-loop interrupts.
- Build type-safe agents with Pydantic AI, and choose the right framework for a given problem.
- Build and test MCP servers in Python that expose internal tools, data, and APIs.
- Connect MCP servers to agents and to coding assistants like Claude Code and Cursor.
- Design multi-agent systems: planner/worker patterns, hand-offs, and failure recovery.
- Evaluate agents with trajectory evals, tool-call accuracy metrics, and regression suites.
- Apply security guardrails: prompt-injection defenses, tool sandboxing, and least-privilege credentials.
- Run open-weight models locally with Ollama and vLLM, and design agents for restricted environments.
- Deploy agents with observability, cost tracking, and graceful failure handling.

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Software Requirements

Python 3.13+, an Anthropic API key (instructions for OpenAI and Gemini also provided), Ollama installed locally, VS Code or an editor of choice, Docker or Podman, and Git.

Training Topics

Agent Fundamentals from Scratch

- The agent loop: plan, act, observe, iterate
- Tool dispatch and structured tool results
- Working memory and conversation state
- Stop conditions, budgets, and runaway prevention
- What frameworks abstract — and what they cost you

Tool Design for Agents

- Tool schemas models use reliably
- Granularity: few powerful tools vs. many narrow ones
- Error contracts and recoverable failures
- Idempotency and side-effect safety

Agent Frameworks: LangGraph

- Graph-structured workflows: nodes, edges, and state
- Checkpointing and resumable runs
- Human-in-the-loop interrupts and approvals
- Streaming agent progress to clients

Agent Frameworks: Pydantic AI

- Type-safe agents with dependency injection
- Structured outputs as agent results
- Testing agents with overridden dependencies
- Framework selection: LangGraph vs. Pydantic AI vs. plain Python

Building MCP Servers in Python

- MCP architecture: hosts, clients, and servers
- Tools, resources, and prompts
- Building servers with the Python SDK and FastMCP
- Transports: stdio and streamable HTTP
- Testing and debugging MCP servers

Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Consuming MCP

- Connecting MCP servers to your own agents
- MCP in coding assistants: Claude Code and Cursor
- Authentication and credential scoping
- Versioning and maintaining tool contracts

Multi-Agent Orchestration

- Planner/worker and supervisor architectures
- Agent hand-offs and shared context
- Parallel agents and result aggregation
- Failure recovery and partial-result handling

Evaluating Agents

- Trajectory evals: judging the path, not just the answer
- Tool-call accuracy and efficiency metrics
- Golden tasks and regression suites in CI
- LLM-as-judge for agent transcripts

Security and Guardrails

- Prompt injection via tool results and retrieved content
- Tool sandboxing and least-privilege credentials
- Approval gates for destructive actions
- Audit trails for agent actions

Local and Restricted Environments

- Open-weight models: the current landscape
- Serving locally with Ollama and vLLM
- Tool calling with local models: capabilities and limits
- Architecting agents for data-residency and compliance constraints

Production Operations and Capstone

- Tracing, token budgets, and cost attribution
- Deploying agents as services
- Capstone: build, evaluate, and demo a tool-using agent with a custom MCP server
- Q&A session