Cloud Contraptions
TRAINING & CONSULTING

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

# LLM Application Development with Python

## Class Duration

35 hours of live training delivered over 5 days.

## Student Prerequisites

- Professional Python development experience
- Familiarity with REST APIs and async programming (`asyncio`)
- Working knowledge of Git
- No prior LLM API experience required

## Target Audience

Python developers building production LLM-powered features, services, or applications. Equally relevant for backend engineers building LLM API wrappers, orchestration layers, or agentic pipelines, and for data engineers and scientists turning notebook prototypes into deployed services. This is the Python-only, five-day edition of our *LLM Application Development with TypeScript and Python* course, with deeper coverage of Pydantic, FastAPI integration, retrieval, evaluation, and deployment.

## Description

This course teaches end-to-end LLM application development in Python — from first API call to a deployed, observable, cost-controlled production service. Days one and two cover the core development loop: integrating the frontier model APIs (Anthropic, OpenAI, Gemini), designing prompt pipelines that separate templates from application logic, generating structured outputs validated with Pydantic, and building tool/function calling pipelines with multiple tool invocations. Day three turns components into services: streaming responses over SSE and WebSockets from FastAPI endpoints, conversation state and context window budget management, and production reliability patterns — retries, timeouts, fallbacks, and rate-limit handling. Day four covers grounding and quality: retrieval-augmented generation essentials (embeddings, chunking, vector stores) and the evaluation discipline production apps need — golden datasets, regression suites, and LLM-as-judge patterns. Day five completes the production picture: token and cost tracking, caching, observability with tracing, secrets management,

containerized deployment, and a capstone in which students ship a complete LLM-powered service. Students wanting deeper retrieval coverage should follow with *Production RAG Systems*; for autonomous multi-step systems, see *Building AI Agents with Python and MCP*.

## Learning Outcomes

- Integrate frontier model APIs (Anthropic, OpenAI, Gemini) in Python applications using async clients.
- Design prompt pipelines with parameterized, versioned templates separated from application logic.
- Generate and validate structured outputs with Pydantic, including error recovery for malformed responses.
- Implement tool/function calling pipelines with sequential and parallel tool invocations.
- Build streaming LLM endpoints in FastAPI using SSE and WebSockets with cancellation handling.
- Manage conversation history, context window budgets, and session persistence.
- Apply retry, timeout, fallback-model, and rate-limit patterns for production reliability.
- Implement RAG essentials: embeddings, chunking, vector search, and grounded prompting.
- Evaluate LLM application quality with golden datasets, regression suites, and LLM-as-judge scoring.
- Track tokens and costs per request and session, and apply caching to control spend.
- Deploy containerized LLM services with secrets management, health checks, and tracing.

## Training Materials

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

## Software Requirements

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

Cloud Contraptions
TRAINING & CONSULTING

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

## Training Topics

### Model API Integration

- Anthropic, OpenAI, and Gemini Python SDK setup
- Messages API request/response structure
- Sync, async, and streaming clients
- Error types, retry strategies, and idempotency

### Prompt Pipeline Design

- Pipeline architecture: input → prompt → model → output
- Separating prompt templates from application logic
- Parameterized prompts and prompt versioning
- Multi-step pipelines and prompt chaining

### Structured Outputs with Pydantic

- JSON mode and native structured output support
- Pydantic models as output contracts
- Error recovery for malformed outputs
- Type-safe response handling patterns

### Tool/Function Calling in Applications

- Tool definition and invocation round-trip
- Sequential and parallel tool call patterns
- Tool result aggregation and re-prompting
- Error handling in tool pipelines

### Streaming Services with FastAPI

- Streaming over Server-Sent Events and WebSockets
- Async generators and backpressure
- Cancellation and client-disconnect handling
- Request validation and dependency injection

### Conversation State Management

- Storing and trimming message history
- Context window budget management
- Summarization for long conversations
- Session persistence with a data store

Cloud Contraptions
TRAINING & CONSULTING

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

## Reliability Patterns

- Retry with exponential backoff
- Timeout and cancellation handling
- Rate limit management
- Primary/fallback model routing

## RAG Essentials

- Embeddings and similarity search
- Chunking strategies and metadata
- Vector stores: pgvector and hosted options
- Grounded prompting and citation patterns

## Evaluation and Quality

- Golden datasets and regression suites
- LLM-as-judge scoring and its pitfalls
- Evals in CI: catching prompt regressions
- Human review workflows

## Cost, Observability, and Deployment

- Token counting and per-session cost attribution
- Prompt and response caching
- Tracing and metrics for LLM calls
- Secrets management and containerized deployment
- Health checks and graceful degradation

## Capstone Workshop

- Build and deploy a streaming, tool-using LLM service with FastAPI
- Structured output validation pipeline lab
- Eval suite lab: golden dataset + CI regression gate
- Q&A session