Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: +1 434-509-6890 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. Every module is grounded in the frontier model APIs as they stand in mid-2026 (the Anthropic, OpenAI, and Gemini Python SDKs against the Claude 5/4.x, GPT-5.x, and Gemini 3.x families) with patterns that transfer across providers. The core development loop comes first: sync, async, and streaming clients, versioned prompt pipelines, structured outputs validated with Pydantic, and tool calling with explicit error contracts. The course then turns components into services with FastAPI streaming over SSE and WebSockets, conversation and context window management, reliability patterns, and caching strategies, before covering RAG essentials and the evaluation discipline production apps need. The final day completes the production picture with security, observability and cost management,

containerized deployment, latency engineering, and the team workflows that keep prompt changes safe at scale. Developers who want a broad survey of the field first can start with *Generative AI and LLMs for Python Programmers*; natural follow-ons are *Production RAG Systems* and *Building AI Agents with Python and MCP*.

Learning Outcomes

- Integrate frontier model APIs (Anthropic, OpenAI, Gemini) in Python applications using sync, async, and streaming clients.
- Design prompt pipelines with parameterized templates managed as versioned, reviewable code.
- Generate structured outputs validated with Pydantic, and build tool calling pipelines with sequential and parallel invocations and explicit error contracts.
- Build streaming LLM endpoints in FastAPI over SSE and WebSockets, managing conversation history, context window budgets, and session persistence.
- Apply production reliability and cost patterns: retries, timeouts, fallback models, rate-limit handling, provider prompt caching, and application-level response caching.
- Implement RAG essentials (embeddings, chunking, pgvector, hybrid search) and evaluate quality with golden datasets, regression suites, and LLM-as-judge scoring wired into CI.
- Secure LLM applications against prompt injection, unsafe output handling, and PII leakage, with per-request token and cost tracking.
- Deploy containerized LLM services with secrets management, health checks, and graceful degradation, engineer for latency, and establish team workflows for safe prompt changes.

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.

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

Training Topics

Project Setup and Model API Integration

- Anthropic, OpenAI, and Gemini Python SDK setup
- Messages API request/response structure
- Sync, async, and streaming clients
- API keys, environments, and configuration management
- Error types, retry strategies, and idempotency
- Choosing models across the Claude 5/4.x, GPT-5.x, and Gemini 3.x families

Prompt Pipeline Architecture

- Pipeline architecture: input → prompt → model → output
- Separating prompt templates from application logic
- Multi-step pipelines and prompt chaining
- Routing: classification steps that pick the next prompt
- Pipeline boundaries: where validation and logging live
- Testing pipeline stages in isolation

Template Management and Versioning

- Parameterized prompts and template rendering
- Prompt versioning: templates as reviewed, deployable artifacts
- System prompts vs. per-request content
- Migrating prompts across model versions
- Rollbacks and comparing prompt versions in production

Structured Outputs with Pydantic

- JSON mode and native structured output support
- Pydantic models as output contracts
- Field descriptions and constraints as model guidance
- Nested models, enums, and optional fields
- Error recovery for malformed outputs
- Type-safe response handling patterns

Tool/Function Calling in Applications

- Tool definition and invocation round-trip
- Schema design: names, descriptions, and parameters models use reliably

- Sequential and parallel tool call patterns
- Tool result aggregation and re-prompting
- Error contracts: what the model should see when tools fail
- Limiting loops: budgets and termination conditions

Conversation State Management

- Storing and trimming message history
- Context window budget management
- Summarization for long conversations
- Preserving tool results and key facts across turns
- Session persistence with a data store
- Multi-user isolation and session lifecycle

Streaming with Server-Sent Events

- Streaming over Server-Sent Events from FastAPI
- Async generators and backpressure
- Event formats: tokens, deltas, and final messages
- Cancellation and client-disconnect handling
- Buffering, flushing, and proxy considerations

WebSockets and Interactive Sessions

- When WebSockets earn their complexity over SSE
- Bidirectional flows: interrupts and mid-stream input
- Connection lifecycle, heartbeats, and reconnection
- Request validation and dependency injection in FastAPI
- Scaling stateful connections

Resilience Patterns

- Retry with exponential backoff and jitter
- Timeout and cancellation handling
- Rate limit management and request queuing
- Primary/fallback model routing
- Circuit breakers and load shedding
- Degrading gracefully when providers fail

Caching Strategies

- Provider prompt caching: structuring prompts for cache hits
- What prompt caching saves - and what breaks it

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Application-level response caching
- Semantic caching: possibilities and pitfalls
- Cache invalidation when prompts and models change

RAG Essentials

- Embeddings and similarity search
- Chunking strategies and metadata
- Vector stores: pgvector and hosted options
- Hybrid search: combining keyword and vector retrieval
- Grounded prompting and citation patterns
- When RAG is the wrong tool

Evaluation and Quality

- Golden datasets and regression suites
- Building eval sets from real traffic and failures
- LLM-as-judge scoring and its pitfalls
- Pairwise comparison and rubric-based grading
- Evals in CI: catching prompt regressions
- Human review workflows

Security for LLM Applications

- Prompt injection: direct and via retrieved content
- Treating model output as untrusted input
- Safe rendering and output sanitization
- PII handling: redaction and data minimization
- Secrets and credentials: keeping them out of prompts
- Abuse prevention: quotas and usage policies

Observability and Cost Management

- Token counting and per-session cost attribution
- Tracing and metrics for LLM calls
- Structured logging of prompts, responses, and tool calls
- Dashboards and alerting on cost, latency, and error rates
- Capturing production traffic for future evals

Deployment

- Containerized deployment with Docker or Podman
- Secrets management

To discuss this course and customizations:
Call: +1 434-509-6890 or Email: sales@cloudcontraptions.com

- Health checks and graceful degradation
- Rolling out prompt and model changes safely
- Scaling: concurrency, worker models, and provider rate limits

Performance and Latency Engineering

- Latency anatomy: time-to-first-token vs. total completion time
- Streaming-first UX as a latency strategy
- Choosing smaller, faster models where they suffice
- Parallelizing independent LLM calls
- Measuring and budgeting latency per pipeline stage

Team Workflows

- Prompt changes as code review: diffs, owners, and approvals
- Eval gates before merging prompt and model changes
- Shared tooling: clients, templates, and eval harnesses across teams
- Documenting model and prompt behavior for teammates
- Staying current as models and SDKs evolve

