Faysal Aziz

AI-App Craftsman

3-Day Bootcamp from Python to Azure Agent Prototype

FA-0523 · 3-Day

Agentic & Generative AI · Software Development · DevOps, Cloud & Infrastructure

Introduction

This 3-day bootcamp is not a gentle immersion — it's a sprint. You'll arrive with some Python and HTTP knowledge and leave ready to build, wire, and prototype AI apps backed by agents and Azure infrastructure. Gone are lengthy detours through theory; instead, you'll engage in live coding, architectural critique, labs, and real-world considerations gained from decades in enterprise AI and software systems. While we simplify some of the most exotic corners, this remains a **highly intensive** course: you'll be pushed, you'll sweat, and you'll ship working prototypes by day three.

Learning Outcomes (by end of 3 days)

- Rapid fluency (or re-fluency) in modern Python, including asynchronous patterns and concurrency.
- Ability to build async HTTP clients (REST, streaming) with error handling and observability.
- Construct web APIs (using FastAPI) that wrap LLM endpoints with fallback and throttling logic.
- Integrate both local (when possible) and remote LLMs in Python, with prompt engineering, fallback strategies, and awareness of tradeoffs.
- Compose reasoning pipelines / chains and basic agent logic (tool invocation, memory, retries).
- Implement basic Retrieval Augmented Generation (RAG) over domain data.
- Understand agent architectures and patterns, though advanced orchestration is optional.
- Prepare a minimal Azure-targeted AI app prototype (chain + agent) suitable for later migration into Foundry.

- Critically evaluate design tradeoffs in latency, cost, model drift, reliability.
- Leave confident to continue from prototype to production inside Azure ecosystems with minimal friction.

Prerequisites

Participants should come in with:

- Basic Python (functions, classes, data structures)
- Familiarity with HTTP/REST, JSON
- Basic command-line / shell skill

Training outline

1. Python & Async Foundations + HTTP Integration

Core Python refresh

- Data types, control flow, comprehension
- Functions, closures, decorators, context managers
- Modules, packages, imports, virtual environments
- Type hints & basic error handling

Async & concurrency fundamentals

- Blocking vs non-blocking I/O; threads vs processes
- asyncio essentials: event loop, async/await, tasks
- Task coordination: gather, cancellation, timeouts
- Integrating blocking code via executors
- Basic backpressure and throttling strategies

Async HTTP & network I/O

- httpx (async) / aiohttp usage
- Streaming / chunked responses
- Retries, exponential backoff, error policies
- Basic authentication, headers, status handling

Observability in async systems

- Logging, correlation IDs, tracing basics
- Measuring latency, error counts, instrumenting HTTP calls

2. Web APIs & AI Endpoints

Web service design principles

- HTTP verbs, status codes, headers, payloads, error models
- JSON serialization / validation

FastAPI deep dive (we drop explicit Flask coverage to save time)

- Path operations, Pydantic request/response models
- Dependency injection, background tasks
- Async endpoints, streaming responses
- Auto-generated docs, versioning

Wrapping AI endpoints

- Defining endpoints (e.g. /chat, /generate)
- Throttling, batching, fallback logic
- Chaining multiple LLM calls within a request
- Concurrency control, rate limiting

Architecture & layering

- Separation: handlers / services / utils
- Configuration, secrets, environment management
- Error propagation & fallback paths

3. Local + Remote LLMs, Prompt Engineering & Chain Logic

Running models locally (lighter treatment)

- Motivations, limitations
- Basic usage: transformers with small models
- Moving to GPU / quantized inference (overview)
- Handling memory limits, fallback to remote

Prompt engineering essentials

- Templates, few-shot, chain-of-thought
- Guardrails, prompt versioning, prompt testing
- Injection risks, fallback prompts

Remote LLM APIs

- OpenAI / Azure OpenAI integration
- Auth, batching, error handling
- Hybrid strategy: local + remote fallback
- Cost and latency tradeoffs

Chaining & reasoning

- Concept: chains, nodes, simple branching
- Building simple chains (e.g. question → retrieval → summarization)
- Debugging, instrumentation

Retrieval Augmented Generation (RAG)

- Document indexing: embeddings, vector stores (e.g. FAISS)
- Similarity search, context assembly
- Prompt grounding, managing hallucination
- Context window and truncation strategies

Agent Basics, Architecture, and Azure Prototype

Basic agent patterns

- Agent as loop: planning, invocation, replan, abort
- Tool wrappers (e.g. search, calculator, database)
- Memory / state handling, error handling
- Fallback / abort policies

Simplified orchestration

- Single agent with tool invocation
- Optional multi-agent sketch (survey level)

© 2026 Faysal Aziz · Kuala Lumpur