Faysal Aziz

Generative AI with Python

LLM APIs, LangChain and retrieval-based application prototypes

FA-0620 · 4 days

Agentic & Generative AI · Software Development

Build and evaluate a small Python LLM application using provider APIs, LangChain and retrieval-augmented generation.

Introduction

This four-day course is for experienced Python developers with machine-learning foundations. It connects LLM APIs, prompting, LangChain, retrieval and selected Python techniques in a small application-development workflow.

Participants extend one main prototype, with shorter demonstrations of summarisation, code assistance and model-customisation options. The final project emphasises documented behaviour, evaluation and limitations; it is not a promise of production readiness, scalability or reliable autonomous operation.

Learning outcomes

The course teaches participants to:

- Integrate a selected LLM API with appropriate authentication, error handling and usage controls.
- Compare OpenAI, Hugging Face and DeepSeek integration patterns without assuming identical capabilities.
- Use current LangChain components for a small controlled workflow or agent.
- Implement document retrieval and assess grounding, relevance and answer quality.
- Refine prompts and evaluate outputs against a small set of task-specific examples.
- Apply decorators, bounded concurrency and suitable data-processing patterns to the prototype.
- Package a small application and explain further deployment, monitoring and security work needed.

Prerequisites

- Professional Python programming experience.
- Proficiency in object-oriented programming and asynchronous programming.
- A solid understanding of machine learning and deep learning fundamentals.
- Experience with RESTful APIs and JSON data structures.
- Familiarity with Python libraries such as NumPy, pandas, and scikit-learn.

A prepared Python environment with compatible package versions and approved model/API access. Provider availability, model licences, quotas and costs must be checked for the training setup; a chat-product subscription is not assumed to include API usage.

Use public, synthetic or suitably anonymised documents, keep API keys outside source code, and use a bounded training budget.

Training outline

1. Day 1 — LLM Foundations, Customisation and APIs

1. Introduction to Generative AI and LLMs

- Overview of Generative AI and its applications.
- Understanding Large Language Models (LLMs): Architecture and capabilities.
- Ethical considerations and challenges in deploying LLMs.

2. Advanced Machine Learning and Deep Learning Concepts

- Review of neural networks and deep learning architectures.
- Compare transfer learning, fine-tuning and retrieval approaches, including model/data/resource requirements.
- Review a small prepared customisation example or demonstration; do not train a large foundation model from scratch.

3. API Integration with Leading LLM Providers

- Authenticate selected OpenAI, Hugging Face Inference Provider/Endpoint and DeepSeek integrations using current provider documentation; endpoint compatibility does not imply identical model or tool features.
- Handling API responses and error management.

- Secrets, timeouts, bounded retries, rate limits, request validation, data handling and usage/cost controls.

2. Day 2 — LangChain and Prompt Evaluation

4. LangChain Framework for LLM Applications

- Introduction to LangChain and its components.
- Build a small controlled workflow or agent with current LangChain components; use limited tools and clear stopping/approval boundaries.
- State, conversation context and persistence in the selected framework version; distinguish application memory from model weights.

6. Prompt Engineering for Efficient Programming

- Designing effective prompts for various tasks.
- Techniques for refining and iterating prompts.
- Use a small versioned prompt/evaluation set with manual review; generated prompts do not remove evaluation responsibilities.

3. Day 3 — Retrieval and Python Application Patterns

5. Retrieval-Augmented Generation (RAG) Techniques

- Understanding the RAG architecture and its benefits.
- Implementing RAG for document retrieval and question answering.
- Compare retrieval relevance and response quality on a small test set; grounding does not guarantee correctness.

7. Advanced Python Techniques

Decorators:

- Creating and applying decorators for code modularity.
- Use cases in logging, authentication, and performance monitoring.

Recursion:

- Implementing recursive solutions for complex data structures.

- Assess recursion depth and alternatives rather than assuming recursion improves performance.

Parallel Computing:

- Compare threads, multiprocessing and asynchronous I/O for suitable tasks; behaviour depends on Python build and workload.
- Bound concurrent requests, handle cancellation/errors and respect API/resource limits.

4. Day 4 — Application Integration, Packaging and Project Review

Application Patterns and One Integrated Prototype

- Build a small chatbot prototype using one selected LLM integration and LangChain.
- Demonstrate retrieval-assisted document summarisation and compare the summary to the source.
- Review a code-assistance example with tests and manual checks; generated code is not trusted without review.

Packaging and Deployment Considerations

- Containerizing applications with Docker.
- Review a prepared deployment example and cloud requirements; no production cloud rollout is promised.
- Observe sample logs, latency, failures and usage; discuss monitoring and maintenance needs.

10. Capstone Project

- Extend the main prototype to combine an LLM integration, LangChain, retrieval and selected Python patterns within a defined, manageable task.
- Present the prototype, evaluation evidence, known failures, privacy boundaries and further work needed.

© 2026 Faysal Aziz · Kuala Lumpur