Design and Implement Generative AI Solution using Python

Duration: 08 days (64 hours)

Course Outcomes:

- Master Python basics and essential libraries for AI development.
- Develop and deploy Python-based AI agents.
- Implement Generative AI models and understand LLMs.
- Enhance AI functionality with advanced prompt engineering.
- Integrate vector databases and build RAG systems.
- Boost coding efficiency with AI-powered tools like Cursor.ai.

Section 1: Python Basics

Module 1: Python Basics for AI Applications

1.1. Core Python Concepts (For Beginners or as a Refresher)

- Variables, data types, and control flow
- Functions and modules
- Object-oriented programming
    - Classes, inheritance
- Working with files and I/O operations
- Building APIs with Flask and Django

1.2. Data Handling in Python

- Lists, dictionaries, sets, and tuples
- List comprehensions and lambda functions
- Error handling and debugging basics

Module 2: Key Python Libraries for AI

2.1. NumPy

- Basics of numerical computations and arrays.
- Operations: array creation, slicing, indexing, and broadcasting.
- Use in data manipulation for AI models.

2.2. Pandas

- Dataframes for structured data.
- Data cleaning, filtering, and manipulation.
- Use cases in AI workflows (e.g., preprocessing training data).

2.3. TensorFlow and PyTorch

- Basics of deep learning libraries.
- Creating simple neural networks.
- Differences and when to use each.

2.4. Other Libraries

- Matplotlib/Seaborn: For data visualization.
- Scikit-learn: For basic machine learning tasks.

Module 3: Using Python with LLM APIs

3.1. Accessing LLM APIs

- Overview of popular APIs (OpenAI, Anthropic, Hugging Face, etc.).
- Setting up API keys and environment.
- Making API calls using Python (basic examples).

3.2. Handling Responses from LLMs

- Parsing responses (e.g., JSON parsing).
- Handling long responses and token limits.

3.3. Writing and Evaluating Prompts

- Writing effective prompts using Python scripts.
- Fine-tuning prompts based on output quality.

Module 4: Building Python-Based AI Agents

4.1. Building a Simple AI Agent in Python

- Define a task (e.g., a chatbot or data-driven decision-maker).
- Integration with LLM APIs for decision-making.
- Simple examples:
    - Customer service chatbot.
    - Text summarization assistant.
    - Data analysis bot (integrated with Pandas and LLMs).

4.2. Advanced AI Agent Concepts

- Adding memory and context to agents (e.g., conversation history).
- Multi-step reasoning and task automation.

4.3. Deploying Python-Based AI Agents

- Running agents locally or on the cloud.

- Deployment considerations: latency, scaling, and API usage limits.

Section 2: Generative AI

Module 5: Generative AI

5.1. What is Generative AI?

- Definition and core concepts.
- Difference between Generative AI and traditional AI.

5.2. History and Evolution

- Milestones in AI development.
- The transition from rule-based systems to deep learning models.
- Evolution of large language models (LLMs).
- Experiment with LLama, Mixtral 7B, GPT models, Hugging Face, and On-premise LLMs.

5.3. Applications

- Text, image, and code generation.
- Business applications (chatbots, virtual assistants, summarization).

5.4. Ethical Considerations

- Bias, toxicity, misinformation, and societal impact.
- Responsible AI development and usage.
- Fairness, transparency, and accountability.

Module 6: Overview and Comparison of LLMs

6.1. Overview of Large Language Models (LLMs)

- Basic principles of LLMs.
- Architecture and functionality.

6.2. Comparison of Popular LLMs

- GPT (OpenAI), PaLM (Google), Claude (Anthropic), LLaMA (Meta), Mixtral 7B.
- Strengths, weaknesses, and specific use cases.

6.3. Training Paradigms

- Pre-training and fine-tuning.
- Reinforcement Learning from Human Feedback (RLHF).

  Instruction-tuning.

Module 7: Sentence Embeddings

7.1. Overview of Sentence Embeddings

- What are embeddings and why are they important?
- Embeddings as the backbone of natural language understanding.

7.2. Techniques

- Word2Vec.
- GloVe.
- BERT and its successors (e.g., RoBERTa, Sentence-BERT).

7.3. Applications

- Semantic search.
- Clustering and classification.
- Recommendation systems.

Module 8: Prompt Engineering

8.1. Basics of Prompt Engineering

- How prompts influence AI behavior.
- Types of prompts (simple, contextual, structured).

8.2. Optimizing Prompts

- Iterative refinement of prompts.
- Using examples and demonstrations in prompts.

8.3. Advanced Techniques

- Instruction-tuning and its significance.
- Chain-of-thought prompting for complex reasoning.

Module 9: Tailoring Prompts for Coding Assistance

9.1. Prompts for Code Generation

- Writing effective prompts for generating functional code.
- Tailoring for specific programming languages.

9.2. Refactoring and Debugging

- Guiding LLMs to refactor existing code.
- Debugging strategies with LLMs.

9.3. Code Documentation

- Generating clear and comprehensive comments.
- Writing README files and technical guides.

Module 10: Introduction to Vector Databases

10.1. What Are Vector Databases?

- Storing and retrieving high-dimensional vectors.
- Why vector databases are critical for LLM-based applications.

10.2. Tools

- Pinecone.
- Weaviate.
- Other alternatives (e.g., Milvus, Chroma).

10.3. Integration

- Connecting LLMs with vector databases for enhanced performance.

Module 11: Retrieval-Augmented Generation (RAG)

11.1. Concept of RAG

- Combining LLMs with external data sources.
- Benefits of retrieval augmentation.

11.2. Building RAG Systems

- Architecture and design principles.
- Integration with vector databases and APIs.

Module 12: Introduction to LLM Agents

12.1. What Are LLM Agents?

- Defining autonomous AI agents.
- Role of LLMs in agent-based systems.

12.2. Frameworks

- LangChain: Modular components for LLM applications.
- LlamaIndex: Enhancing data integration for agents.

12.3. Designing Architectures

- Agent workflows and task management.

- Integrating APIs and tools for advanced functionality.

Module 13: Multi-Agent Systems

13.1. Concept of Multi-Agent Systems

- Overview and key characteristics.
- Benefits of collaboration between agents.
- Agent Frameworks (AutoGen, Crew AI).

13.2. Communication

- Coordination and message-passing techniques.
- Challenges in multi-agent communication.

13.3. Applications

- Distributed problem-solving.
- Use cases in simulations and decision-making.

Module 14: Introduction to Cursor.ai

14.1. What is Cursor.ai?

- Overview of Cursor.ai as an AI-powered coding assistant.
- Designed for developers to enhance productivity in coding, debugging, and refactoring.

14.2. Key Features of Cursor.ai

- Auto-completion: Context-aware code suggestions for faster development.
- Code Debugging: Identify and resolve issues with AI-guided assistance.
- Code Refactoring: Simplify and optimize code structure while maintaining functionality.
- Code Querying: Ask the AI about specific code sections for explanations or improvements.

14.3. How Cursor.ai Works

- Underlying AI capabilities (leveraging LLMs).
- Integration with development environments (e.g., VS Code, JetBrains).
