Intelligent AI apps and autonomous agents Development

Building Intelligent, Agentic, and Production-Ready AI Solutions on Azure

Course Duration: 40 Hours (5 Days) | Audience: Python Developers & AI Engineers

Overview

This five-day advanced, hands-on program prepares Python developers and AI engineers to design, build, evaluate, and deploy intelligent AI applications and autonomous agents on Microsoft Azure. The course is structured directly around the official AI-103 exam skill domains: planning and managing Azure AI solutions, implementing generative AI and agentic solutions, computer vision, text analysis, and information extraction. Participants will work hands-on with Microsoft AI Foundry, Azure OpenAI, Semantic Kernel, the Microsoft Agent Framework, Azure AI Search, Azure AI Vision, Azure AI Language, Azure AI Speech, and Azure AI Document Intelligence. Labs are delivered through Microsoft Learn on Demand (LOD) environments using Python throughout.

Audience Profile

This course is designed for Python developers, AI engineers, solution architects, and data scientists who build or want to build AI-infused applications and agents on Microsoft Azure. Candidates should be comfortable with Python, REST APIs, and Azure fundamentals.

Prerequisites

- Proficiency in Python including functions, classes, async/await, and working with JSON
- Familiarity with REST APIs and consuming SDKs in Python
- Basic understanding of Azure services, the Azure portal, and cloud computing concepts
- Awareness of machine learning concepts is beneficial but not required
- Labs are delivered via Microsoft LOD — no personal Azure subscription is required
- Familiarity with VS Code or a similar IDE and working in a terminal is recommended

What You Will Be Able to Do After This Course

- Plan, provision, and secure Azure AI Foundry hubs, projects, and model deployments using managed identity and RBAC
- Select appropriate Azure AI models and services for a given scenario and justify the choice
- Configure private networking, content filters, quota management, and cost controls for Azure AI solutions
- Monitor AI solution health, token usage, latency, and safety events using Azure Monitor and AI Foundry diagnostics
- Build chat completion, function calling, structured output, and streaming applications using Azure OpenAI in Python
- Implement vector embeddings, hybrid search, and full RAG pipelines grounded in enterprise data using Azure AI Search
- Build and evaluate AI agents with tools, memory, and planning using Semantic Kernel in Python

- Orchestrate multi-agent systems with role assignment and task delegation using the Microsoft Agent Framework
- Analyze images, video, and multimodal content using Azure AI Vision and GPT-4o vision capabilities
- Extract entities, summaries, sentiment, and structured data from text using Azure AI Language and Speech
- Extract structured information from documents, forms, and PDFs using Azure AI Document Intelligence
- Evaluate AI outputs using AI Foundry built-in evaluators for groundedness, relevance, safety, and coherence
- Apply responsible AI controls including content safety, prompt injection detection, and groundedness checks
- Deploy AI solutions to production using Container Apps, Key Vault, CI/CD pipelines, and Foundry prompt flows

Course Syllabus

Module 1: Planning an Azure AI Solution with Microsoft AI Foundry

- The Microsoft AI platform in 2026: Azure AI Foundry, Azure OpenAI, Azure AI Services, and the AI ecosystem
- AI Foundry architecture: hubs, projects, connections, deployments, and compute resources
- Model selection criteria: capability, cost, latency, token limits, multimodal support, and compliance requirements
- Comparing model families available on Azure: GPT-4o, o-series reasoning models, Phi-4, and embedding models
- Deployment types: standard pay-as-you-go vs. provisioned throughput — when to choose each
- Lab: Create an AI Foundry hub and project, deploy a GPT-4o model, and connect to it from Python using the Foundry SDK

Module 2: Securing and Governing Azure AI Solutions

- Authentication patterns: API keys vs. Microsoft Entra ID managed identity — why managed identity is production-standard
- Role-Based Access Control (RBAC) for AI Foundry: Cognitive Services OpenAI User, Contributor, and custom roles
- Private networking for Azure AI: virtual networks, private endpoints, and disabling public access
- Content filtering in Azure OpenAI deployments: severity levels, block vs. annotate policies, and custom blocklists
- Responsible AI principles: fairness, reliability, privacy, inclusiveness, transparency, and accountability in AI design
- Quota management: model quota limits, request throttling, and retry-with-exponential-backoff patterns in Python
- Lab: Configure managed identity authentication, assign RBAC roles, and add a content filter policy to an Azure OpenAI deployment

Module 3: Monitoring, Cost Management, and CI/CD for AI Solutions

- Monitoring Azure AI solutions: diagnostic settings, Azure Monitor, Log Analytics, and Application Insights integration

- Key AI metrics to monitor: token consumption, request latency, error rates, content filter events, and safety alerts
- Cost management: estimating token spend, setting budget alerts, and using deployment quotas to cap costs
- Prompt flow in AI Foundry: building, versioning, and testing LLM pipelines as deployable artifacts
- CI/CD for AI: automating prompt flow deployment using Azure DevOps pipelines or GitHub Actions
- Troubleshooting AI solution failures: reading diagnostic logs, tracing API errors, and identifying throttling patterns
- Lab: Enable diagnostic logging for an AI Foundry project and build a simple prompt flow with automated deployment

Module 4: Azure AI Language — Text Analysis and NLP

- Azure AI Language service overview: prebuilt capabilities vs. custom model training
- Sentiment analysis and opinion mining: document-level and aspect-based sentiment in Python
- Named entity recognition (NER) and PII detection: identifying and redacting sensitive information
- Key phrase extraction and text summarization: abstractive and extractive summaries via the Language API
- Conversational Language Understanding (CLU): intents, entities, utterances, and model training
- Structured output from language analysis: combining Language API results with OpenAI for enriched responses
- Lab: Build a Python pipeline that detects PII, extracts key phrases, and generates an abstractive summary from a document corpus

Module 5: Azure AI Speech and Translation

- Azure AI Speech: real-time speech-to-text and batch transcription using the Speech SDK in Python
- Text-to-speech: neural voices, SSML customization, and streaming audio output
- Speaker recognition and diarization: identifying and separating multiple speakers in a transcript
- Azure AI Translator: real-time text translation, language detection, and transliteration
- Custom speech models: fine-tuning recognition for domain-specific vocabulary and accents
- Combining Speech and Azure OpenAI: building a voice-enabled AI assistant pipeline
- Lab: Build a voice-to-summary Python application that transcribes a recorded meeting, detects speakers, translates output, and generates a structured summary

Module 6: Azure OpenAI — Chat Completions, Prompt Engineering, and Function Calling

- Azure OpenAI chat completions API: system messages, conversation history, and multi-turn dialogue in Python
- Prompt engineering techniques: zero-shot, few-shot, chain-of-thought, and structured system instruction design
- Streaming completions: real-time token-by-token response handling for responsive user experiences
- Function calling: defining tools, handling tool call responses, executing Python functions, and returning results
- Parallel function calling and structured output mode: enforcing JSON schema compliance in model responses
- Token management: counting tokens with tiktoken, managing context window limits, and truncation strategies

- Lab: Build a multi-turn Python chat assistant with streaming, function calling to a live REST API, and structured JSON output

Module 7: Vector Embeddings, Azure AI Search, and RAG Architecture

- What embeddings are: converting text into semantic vector representations using Azure OpenAI embedding models
- Generating embeddings in Python with text-embedding-3-large and text-embedding-3-small
- Azure AI Search: indexes, fields, semantic configuration, vector search, and integrated vectorization setup
- Chunking strategies for RAG: fixed-size, paragraph-based, and semantic chunking — trade-offs for retrieval quality
- Hybrid search with RRF: combining BM25 keyword scoring and vector similarity for precision and recall
- Semantic ranker: applying a cross-encoder re-ranking model to improve hybrid search result quality
- Lab: Chunk and embed a document corpus, index into Azure AI Search with integrated vectorization, and run hybrid semantic queries from Python

Module 8: Building Complete RAG Pipelines

- The full RAG pipeline: chunk, embed, index, retrieve, augment, generate — implemented end-to-end in Python
- Constructing augmented prompts: injecting retrieved context into system messages with source attribution
- Query rewriting: using an LLM to reformulate user queries before retrieval for improved accuracy
- Groundedness and citation: returning document references alongside generated answers for verifiability
- Evaluating RAG quality: groundedness, relevance, answer completeness, and context precision
- Agentic RAG pattern: giving the retrieval decision to an agent rather than using fixed retrieval on every turn
- Lab: Build a grounded RAG application in Python that answers questions over a document library with source citations and query rewriting

Module 9: Computer Vision — Image Analysis, Multimodal AI, and Content Understanding

- Azure AI Vision: image analysis, object detection, captioning, smart crops, and background removal
- GPT-4o multimodal capabilities: sending images and video frames with text in the chat completions API
- Visual question answering: building a Python solution that reasons over images to answer natural language questions
- Image and video generation: Azure OpenAI DALL-E 3 for image generation and Sora-class video models on Foundry
- Azure AI Content Understanding: analyzing complex documents combining text, images, tables, and layout
- Content safety for vision: detecting unsafe image content and configuring visual content filters
- Lab: Build a multimodal Python pipeline that accepts a product image and invoice PDF, describes the image with GPT-4o, and extracts invoice fields using Content Understanding

Module 10: Information Extraction — Document Intelligence and Knowledge Mining

- Azure AI Document Intelligence: prebuilt models for invoices, receipts, ID documents, and tax forms

- Custom document models: labeling fields, training a custom extraction model, and calling it from Python
- Layout analysis: extracting tables, paragraphs, key-value pairs, and reading order from complex PDFs
- Azure AI Search as a knowledge mining platform: AI enrichment skillsets, cognitive skills, and knowledge stores
- Building an end-to-end document extraction pipeline: ingest PDF, extract with Document Intelligence, index in AI Search, query with RAG
- Structured output from extraction: transforming document intelligence results into typed Python objects for downstream processing
- Lab: Build a document processing pipeline that ingests contract PDFs, extracts key clauses and parties with Document Intelligence, stores results in AI Search, and answers questions via RAG

Module 11: Introduction to Agentic AI and the Microsoft Agentic Stack

- What agentic AI is: agents, tools, memory, planning, and autonomous goal-directed behavior
- The agent loop: perceive user input, reason with the LLM, select and call a tool, observe the result, iterate
- Types of agents: ReAct agents, tool-calling agents, planning agents, and multi-agent systems
- When to use an agent vs. a direct LLM call: complexity, multi-step requirements, and external tool integration
- The Microsoft agentic stack: Semantic Kernel for single-agent orchestration, Microsoft Agent Framework for multi-agent coordination
- Human-in-the-loop and responsible agentic AI: setting autonomy boundaries and approval checkpoints

Module 12: Semantic Kernel — Building Single Agents in Python

- Semantic Kernel architecture: kernel, plugins, native functions, prompt functions, memory, and filters
- Configuring Semantic Kernel in Python with Azure OpenAI as the underlying AI service
- Creating native plugins: wrapping Python functions as kernel tools available to the agent
- Creating semantic (prompt) functions: defining reusable LLM-powered steps within the kernel
- Agent memory strategies: conversation history buffer, Azure AI Search vector memory, and Cosmos DB persistence
- Kernel filters and middleware: adding logging, safety checks, retry logic, and token usage tracking to the agent pipeline
- Lab: Build a single Python agent with Semantic Kernel that uses a RAG knowledge plugin and a Microsoft Graph email notification plugin to answer HR policy questions and notify a manager

Module 13: Advanced Agent Patterns — Planners and Tool Orchestration

- Sequential planner: decomposing a complex goal into an ordered list of kernel function calls
- Stepwise planner (ReAct pattern): iterative reasoning and tool execution until the goal is satisfied
- Handlebars planner: deterministic multi-step workflows defined in a template for repeatable processes
- OpenAPI integration: exposing external REST APIs as Semantic Kernel plugins using OpenAPI specifications
- Streaming agent responses: surfacing intermediate reasoning steps and tool results to users in real time
- Error handling and fallback in agent pipelines: retries, graceful degradation, and escalation to a human
- Lab: Build a procurement research agent that uses a stepwise planner to search suppliers, compare options using a RAG plugin, and draft a recommendation memo in Word via the Graph API

Module 14: Microsoft Agent Framework — Multi-Agent Orchestration

- Microsoft Agent Framework overview: AutoGen-based multi-agent coordination built for Azure
- Core concepts: agents, group chat, message passing, and conversation termination conditions
- Defining agent roles and personas: orchestrator, specialist, critic, and executor agents
- Group chat patterns: round-robin, selector-based routing, and hierarchical team orchestration
- Shared tool access: giving multiple agents in a group access to the same plugin or external service
- Nested agent teams: orchestrating groups of agents that themselves contain sub-agent teams
- Lab: Build a three-agent Python system — an orchestrator delegates a document analysis task to a specialist agent; a critic agent reviews and scores the output before returning it to the user

Module 15: Human-in-the-Loop, Observability, and Enterprise Data Connectivity

- Human-in-the-loop patterns: pausing the agent loop at defined checkpoints to request user approval or additional input
- Agent observability: tracing agent conversations, logging tool calls, and measuring task completion rates
- Connecting agents to Azure SQL and Cosmos DB via Python tool functions for live data access
- Microsoft Graph API as an agent tool: reading emails, calendar events, Teams messages, and SharePoint files
- Power Automate integration: triggering enterprise workflows from within an agent topic as an action
- Security for agent tool calls: managed identity, least-privilege scopes, and secret management with Key Vault
- Lab: Extend the Day 3 multi-agent system with a human approval checkpoint, full conversation trace logging, and a Cosmos DB order lookup tool

Module 16: Responsible AI and Content Safety

- Microsoft responsible AI principles applied to production systems: fairness, reliability, privacy, and accountability
- Azure AI Content Safety API: detecting harmful categories (violence, hate, sexual, self-harm) in text and images
- Configuring content filters on Azure OpenAI deployments: severity thresholds, annotation mode, and custom blocklists
- Prompt injection and jailbreak detection: identifying adversarial inputs designed to override system instructions
- Groundedness detection: identifying AI responses that are not supported by the retrieved source context
- Protected material detection: identifying responses that reproduce copyrighted content or PII from the model
- Lab: Integrate Content Safety API, groundedness detection, and prompt injection filtering into a Python RAG pipeline

Module 17: Evaluating AI Solutions in Microsoft AI Foundry

- Why evaluation is a first-class concern in AI-103: quality, safety, and continuous improvement
- Built-in AI Foundry evaluators: groundedness, relevance, coherence, fluency, similarity, and safety
- Running batch evaluations in AI Foundry: submitting a labeled dataset and analyzing aggregated results

- Custom evaluators: writing Python-based evaluation functions for domain-specific quality criteria
- A/B testing deployments: comparing two model or prompt versions using evaluation datasets in Foundry
- Trace-based debugging: using AI Foundry traces to identify which step in the pipeline caused a quality failure
- Building a continuous evaluation loop: automating quality gates as part of a CI/CD pipeline for AI deployments
- Lab: Run a full evaluation of a RAG agent using AI Foundry built-in evaluators and a custom groundedness evaluator written in Python

Module 18: Production Deployment — Security, Scalability, and MLOps

- Deploying AI solutions as Azure Container Apps with managed identity and private networking
- Environment and secret management: using Azure Key Vault to store and retrieve API keys and connection strings securely
- Scalability patterns: handling concurrent AI workloads with provisioned throughput and load balancing across deployments
- AI Foundry prompt flows as deployable endpoints: versioning, staging, and blue-green deployment strategies
- MLOps for AI: automating prompt flow deployment with GitHub Actions and Azure DevOps pipelines
- Cost optimization in production: semantic caching, batching API calls, and right-sizing model deployments
- Lab: Package a Python AI agent as a Docker container, deploy to Azure Container Apps using managed identity, and set up a GitHub Actions CI/CD pipeline for the prompt flow

Module 19: AI-103 Exam Preparation and Skill Domain Review

- Domain review — Plan and Manage (25-30%): Foundry, model selection, security, networking, monitoring, cost, responsible AI
- Domain review — Generative AI and Agentic Solutions (30-35%): RAG, function calling, agents, multi-agent, evaluation
- Domain review — Computer Vision (10-15%): image analysis, multimodal, video, Content Understanding, safety
- Domain review — Text Analysis (10-15%): language service, sentiment, NER, PII, speech, translation, CLU
- Domain review — Information Extraction (10-15%): Document Intelligence, layout analysis, knowledge mining, AI Search enrichment
- Exam technique: reading scenario questions — identifying the desired state, the constraint, and the service boundary
- Common AI-103 mistakes to avoid: confusing AI-102 service silos with Foundry-first thinking, skipping evaluation, underweighting the planning domain

Module 20: Capstone Project — End-to-End AI App and Agent on Azure

- Capstone brief: build a complete, production-ready AI solution combining at minimum three AI-103 skill domains
- Required components: AI Foundry deployment, Azure AI Search RAG pipeline, a Semantic Kernel or Agent Framework agent, and at least one non-OpenAI Azure AI service

- Security requirement: managed identity authentication, Key Vault secret management, and content safety integration
- Evaluation requirement: run at least one AI Foundry evaluation and present the results alongside the solution
- Each participant or team presents: architecture diagram, Python code walkthrough, live agent demonstration, and responsible AI controls
- Trainer and peer feedback, AI-103 exam registration guidance, and recommended Microsoft Learn paths for continued preparation

Conclusion and Certification

- Participants who complete this program will receive a Certificate of Completion from Koenig Solutions
- The course is fully aligned with Microsoft AI-103: Developing AI Apps and Agents on Azure (April 2026 objectives)
- Participants are strongly encouraged to register for the AI-103 beta or GA exam via Pearson VUE following the training
- Recommended Microsoft Learn paths: Develop AI agents on Azure, Build generative AI solutions with Azure OpenAI, and Implement knowledge mining with Azure AI Search