

**JAVITA TECH**

COURSE CURRICULUM · LIVE ONLINE · 2026 EDITION

Become an AI Engineer

Python + Machine Learning + Generative AI + Agentic AI

An 80-session professional AI engineering journey — one-hour sessions, hands-on from the first line of Python to a deployed, evaluated, multi-component AI system. Concept first, then built live, then you build it yourself in the lab.

80

Live sessions

1 hr

Per live session

Live

on Zoom · recorded

12

phases · 8+ projects

NEXT BATCH**Collecting interest****Date to be confirmed**

Register interest on the course page · no payment

DAYS AND TIMING**To be confirmed**

Tell us the days and times that suit you

MENTOR**To be confirmed**

Confirmed with the batch dates · hands-on lab in every session

01

Python & programming foundation

02

SQL · APIs · FastAPI backend

03

Machine learning

04

LLMs · prompt engineering

05

Embeddings · RAG · LangChain

06

AI agents · LangGraph

07

MCP · evaluation · LLMOps

08

Production-ready AI engineer

Register your interest: training.javitatech.com/become-an-ai-engineer/

WhatsApp **+91 87122 87999** · info@javitatech.com · javitatech.com

The program in one line

From writing Python code to engineering intelligent AI systems. It starts with Python fundamentals and data handling, moves through SQL, APIs and backend engineering with FastAPI, and only then introduces machine learning, large language models, retrieval-augmented generation and AI agents. Each phase supplies a component the next phase depends on.



What is an AI Engineer?

- Builds applications powered by ML models and large language models
- Owns the engineering layer around them: data pipelines, APIs, retrieval systems, agent orchestration, evaluation and production operations
- Sits at the intersection of software engineering, applied machine learning and generative AI systems design
- Writing prompts is a skill inside the role, not the whole role

How every 60-minute session runs

- **10 min** — concept: the “why” behind today’s topic
- **15 min** — live demonstration by the mentor
- **25 min** — hands-on lab: you build it yourself
- **10 min** — troubleshooting, Q&A and a challenge to extend the day’s build
- Every session is recorded · revision recordings for 1 year from full fee payment · module-wise FAQs and soft-copy materials

Who should enroll

Prerequisite: comfort using a computer for development work and basic logical thinking. Prior programming exposure helps, but Days 1–15 build Python from the ground up, so a complete beginner can start from Day 1.

Engineering graduates & freshers

A structured, project-based path into AI engineering roles. Python is taught from Day 1.

Python & software developers

Extend existing coding and backend skills into ML, LLMs, RAG and agents.

Data professionals & ML aspirants

Move from analysis and notebooks to applied ML and AI application building.

Backend & automation engineers

Serve ML/LLM models behind APIs and build agentic, tool-calling systems.

GenAI aspirants

Move beyond prompting into engineering full Generative AI applications.

Professionals transitioning into AI

A sequenced path from fundamentals to production AI engineering.

The AI Engineer skill stack

Eight layers, taught in order, each depending on the one beneath it. Python from Day 2 is still in use on Day 80: SQL feeds ML, APIs feed FastAPI, RAG feeds agents, agents feed MCP.

08	Production	Evaluation · observability · LLMOps · deployment
07	Integration	MCP · APIs · external systems
06	Agents	Tools · memory · LangGraph · multi-agent systems
05	Knowledge	Embeddings · vector DB · RAG · LangChain
04	Generative AI	LLMs · prompt engineering · structured outputs
03	Machine learning	Regression · classification · evaluation
02	Engineering	REST APIs · FastAPI · databases
01	Programming	Python · SQL · Git

Why not learn it from separate tutorials?

- A Python course + an ML course + a GenAI course + a RAG tutorial + an agent tutorial — different instructors, codebases and datasets
- No single project ties the pieces together; gaps appear exactly where real AI applications need integration
- Typical AI learning stops at “build one chatbot”

What you get instead

- One sequenced 80-session journey, one skill stack, one capstone
- Every phase produces a component the next phase reuses
- 8 milestone projects validate each layer before the next is added
- Regular technical mock interviews, resume guidance, interview preparation and certification guidance
- JavitaTech Certificate of Completion (a training certificate, not a vendor certification)

The 80-session roadmap — 12 phases at a glance

Twelve phases, each with a clear engineering purpose, moving from language fundamentals to a production-oriented capstone. The session-by-session plan follows on the next pages.

Phase 01	Sessions 1–15	Python for AI Engineering Programming foundation — every later phase depends on this one.
Phase 02	Sessions 16–21	SQL & Data Engineering Fundamentals Structured data access, required before data-heavy ML work.
Phase 03	Sessions 22–27	Git, GitHub & API Engineering Collaboration and integration skills, required before backend and AI system work.
Phase 04	Sessions 28–38	Machine Learning Fundamentals Predictive modeling — the foundation AI engineering builds on before generative AI.
Phase 05	Sessions 39–43	FastAPI & AI Application Backends The serving layer used to expose ML and, later, LLM-based systems.
Phase 06	Sessions 44–52	Generative AI & LLM Engineering Language intelligence — LLM architecture, APIs and prompt engineering.
Phase 07	Sessions 53–62	Embeddings, RAG & LangChain Knowledge retrieval — grounding LLM responses in real data.
Phase 08	Sessions 63–71	Agentic AI & LangGraph Autonomous reasoning — agents that plan, call tools and act.
Phase 09	Sessions 72–74	Model Context Protocol (MCP) Standardized integration between agents and external systems.
Phase 10	Sessions 75–77	AI Evaluation Quality assurance for LLM, RAG and agent outputs.
Phase 11	Sessions 78–79	LLMOps & Production AI Operating AI systems responsibly: logging, monitoring, deployment.
Phase 12	Sessions 80	Final Capstone Integration demonstration across every layer of the skill stack.

Session-by-session plan — all 80 sessions

Every session has a topic, a hands-on lab and a specific outcome; the highlighted sessions close a phase with a milestone project. The order may be fine-tuned by the mentor for the batch, but every topic below is covered.

PHASE 01 Python for AI Engineering

Sessions 1–15

Purpose: Programming foundation — every later phase depends on this one.

DAY
1

AI Engineer Roadmap & Python Environment

Topics: AI engineering landscape, development environment, Python setup, VS Code/Jupyter, virtual environments

Lab: Set up a complete local AI engineering workspace with Python, VS Code and a virtual environment, and run a first script.

Outcome: A working, isolated Python development environment ready for the program.

DAY
2

Python Fundamentals

Topics: Variables, data types, operators, input/output

Lab: Write scripts that take user input, apply operators, and print formatted output.

Outcome: Can write basic Python scripts using core language constructs.

DAY
3

Control Flow

Topics: Conditions, loops, comprehensions

Lab: Build small programs using if/else, for/while loops and list comprehensions.

Outcome: Can control program execution using conditional and iterative logic.

DAY
4

Python Data Structures

Topics: Lists, tuples, sets, dictionaries

Lab: Model real-world data (e.g. a contact book) using each data structure.

Outcome: Can select and use the right data structure for a given problem.

DAY
5

Functions

Topics: Functions, parameters, return values, scope, lambda

Lab: Refactor repetitive code from earlier days into reusable functions, including lambdas.

Outcome: Can write modular, reusable functions with correct scope handling.

DAY
6

Modules & Packages

Topics: Imports, packages, pip, requirements.txt

Lab: Organize code into modules, install third-party packages with pip, generate requirements.txt.

Outcome: Can structure Python code into reusable modules and manage dependencies.

**DAY
7****Exception Handling**

Topics: try/except, custom exceptions, debugging

Lab: Add error handling and custom exceptions to a script, then debug it step by step.

Outcome: Can write robust Python code that handles and reports failures gracefully.

**DAY
8****Object-Oriented Python**

Topics: Classes, objects, inheritance, encapsulation

Lab: Model a small domain (e.g. a task manager) using classes, inheritance and encapsulation.

Outcome: Can design Python programs using object-oriented principles.

**DAY
9****Files & Data Processing**

Topics: CSV, JSON, text files, reading/writing data

Lab: Read, transform and write data across CSV, JSON and plain text formats.

Outcome: Can work with common file-based data formats used in AI applications.

**DAY
10****Python for Data Handling**

Topics: NumPy fundamentals

Lab: Perform array creation, indexing, reshaping and vectorized operations using NumPy.

Outcome: Can use NumPy for efficient numerical computation.

**DAY
11****Pandas**

Topics: Series, DataFrames, filtering, transformation

Lab: Load a real dataset into Pandas and perform filtering, transformation and aggregation.

Outcome: Can manipulate tabular data using Pandas.

**DAY
12****Data Cleaning**

Topics: Missing values, duplicates, type conversion

Lab: Clean a messy dataset by handling missing values, removing duplicates and fixing types.

Outcome: Can prepare raw data for analysis and modeling.

**DAY
13****Python APIs & HTTP**

Topics: HTTP methods, requests, JSON, REST basics

Lab: Consume a public REST API using the requests library and parse the JSON response.

Outcome: Can call external APIs and work with their responses in Python.

**DAY
14****Python Project** **PROJECT 1**

Topics: Build a small data/API-based Python application

Lab: Build an end-to-end Python application that pulls data from an API, processes and outputs results.

Outcome: Can independently assemble Python fundamentals into a working application. (Project 1)

**DAY
15****Python Assessment & Review**

Topics: Coding exercises, debugging, mini-project review

Lab: Complete structured coding exercises and present the Day 14 project for review.

Outcome: Python foundation validated before progressing to data and backend topics.

PHASE 02 SQL & Data Engineering Fundamentals

Sessions 16–21

Purpose: Structured data access, required before data-heavy ML work.

DAY
16

SQL Fundamentals

Topics: Databases, tables, schema, SELECT, WHERE, ORDER BY

Lab: Query a sample relational database using SELECT statements with filtering and sorting.

Outcome: Can read and query data from a relational database.

DAY
17

SQL CRUD Operations

Topics: INSERT, UPDATE, DELETE, constraints

Lab: Perform create, read, update and delete operations on a working database.

Outcome: Can manage the data lifecycle within a relational database.

DAY
18

Aggregation and Data Analysis

Topics: GROUP BY, HAVING, aggregate functions

Lab: Write aggregation queries to summarize and analyze a business dataset.

Outcome: Can produce analytical summaries directly from SQL.

DAY
19

SQL Joins

Topics: INNER, LEFT, RIGHT, self joins

Lab: Combine data across multiple related tables using different join types.

Outcome: Can query data spread across normalized relational tables.

DAY
20

Advanced SQL

Topics: Subqueries, CTEs, CASE, window functions

Lab: Rewrite complex queries using CTEs and window functions for ranking and running totals.

Outcome: Can write advanced SQL for real analytical and engineering scenarios.

DAY
21

Python + SQL **PROJECT 2**

Topics: Database connectivity, parameterized queries

Lab: Connect to a database from Python, run parameterized queries, load results into Pandas.

Outcome: Can integrate SQL data access directly into Python applications. (Project 2)

PHASE 03 Git, GitHub & API Engineering

Sessions 22–27

Purpose: Collaboration and integration skills, required before backend and AI system work.

DAY
22

Git Fundamentals

Topics: init, add, commit, status, log, .gitignore

Lab: Initialize a repository, track changes, build a meaningful commit history.

Outcome: Can version-control code using Git locally.

DAY
23

Git Workflow

Topics: Branching, merging, pull requests, conflict resolution

Lab: Create feature branches, merge changes, and resolve a deliberate merge conflict.

Outcome: Can manage parallel development using Git branching workflows.

DAY
24

GitHub and Collaboration

Topics: Remote repos, push/pull, issues, collaboration workflow

Lab: Push a project to GitHub, open a pull request, and review a collaborator's changes.

Outcome: Can collaborate on shared codebases using GitHub.

DAY
25

REST API Fundamentals

Topics: Resources, methods, status codes, endpoints, headers

Lab: Design the endpoint structure and status-code behavior for a simple REST API.

Outcome: Understands REST API design principles.

DAY
26

API Consumption with Python

Topics: Authentication, query params, pagination, error handling

Lab: Consume an authenticated, paginated API and handle rate-limit and error responses.

Outcome: Can reliably integrate external APIs into Python applications.

DAY
27

API Engineering Mini Project

Topics: Combining API + Python + data processing

Lab: Build a small application that consumes one or more APIs, processes and stores results.

Outcome: Can engineer a working API-driven Python application end-to-end.

PHASE 04

Machine Learning Fundamentals

Sessions 28–38

Purpose: Predictive modeling — the foundation AI engineering builds on before generative AI.

DAY
28

AI vs ML vs Deep Learning vs Generative AI

Topics: Definitions, relationships, use cases

Lab: Map real business scenarios to the correct category of AI, ML, DL or GenAI.

Outcome: Can correctly position ML within the broader AI landscape.

DAY
29

Machine Learning Workflow

Topics: Problem framing, data, training, evaluation, deployment

Lab: Walk through the ML lifecycle using a sample business problem end-to-end.

Outcome: Understands the full lifecycle a production ML model goes through.

DAY
30

Data Preparation and Preprocessing

Topics: Scaling, encoding, train/test split

Lab: Prepare a raw dataset for modeling using scaling, encoding and train/test splitting.

Outcome: Can prepare structured data for machine learning algorithms.

**DAY
31****Regression****Topics:** Linear regression, MAE, RMSE, R^2 **Lab:** Train and evaluate a linear regression model on a real dataset.**Outcome:** Can build and evaluate regression models for continuous prediction.**DAY
32****Classification****Topics:** Logistic regression, precision/recall/F1, confusion matrix**Lab:** Train a classification model and evaluate it using a confusion matrix.**Outcome:** Can build and evaluate models for categorical prediction problems.**DAY
33****Decision Trees****Topics:** Tree-based models, splits, overfitting**Lab:** Train a decision tree model and visualize its splits and decision boundaries.**Outcome:** Understands tree-based models and their overfitting behavior.**DAY
34****Ensemble Learning****Topics:** Random forests, boosting, bagging**Lab:** Train ensemble models and compare their performance against a single decision tree.**Outcome:** Can apply ensemble methods to improve model performance.**DAY
35****Unsupervised Learning****Topics:** Clustering (k-means), dimensionality reduction**Lab:** Apply clustering to segment a dataset and visualize the resulting groups.**Outcome:** Can apply unsupervised learning when labeled data isn't available.**DAY
36****Model Evaluation****Topics:** Cross-validation, metric selection, bias/variance**Lab:** Apply cross-validation to compare multiple models and select the most reliable one.**Outcome:** Can rigorously evaluate and compare machine learning models.**DAY
37****Feature Engineering****Topics:** Feature creation, selection, transformation**Lab:** Engineer new features from raw data and measure their impact on model performance.**Outcome:** Can improve model performance through better feature design.**DAY
38****End-to-End Machine Learning Project****PROJECT 3****Topics:** Full pipeline, model persistence**Lab:** Build a complete ML pipeline from raw data to a saved, reusable trained model.**Outcome:** Can independently build a working, evaluated ML pipeline. (Project 3)

PHASE 05 FastAPI & AI Application Backends

Sessions 39–43

Purpose: The serving layer used to expose ML and, later, LLM-based systems.

**DAY
39****FastAPI Fundamentals**

Topics: Setup, first endpoint, path/query parameters

Lab: Build and run a first FastAPI service with multiple endpoints.

Outcome: Can stand up a working Python web API service.

**DAY
40****Routes and Request Handling**

Topics: Path operations, request bodies, status codes

Lab: Build multiple routes handling different HTTP methods and request payloads.

Outcome: Can design and implement RESTful routes in FastAPI.

**DAY
41****Pydantic and Validation**

Topics: Schemas, data validation, response models

Lab: Define Pydantic models to validate incoming requests and shape API responses.

Outcome: Can enforce data integrity in an API using schema validation.

**DAY
42****Database Integration and Authentication Basics**

Topics: Connecting FastAPI to a database, basic auth/API keys

Lab: Connect a FastAPI service to a database and secure an endpoint with basic authentication.

Outcome: Can build a FastAPI service backed by persistent, access-controlled data.

**DAY
43****Expose an ML/AI Model through FastAPI****PROJECT 4**

Topics: Model loading, prediction endpoint

Lab: Wrap the Day 38 trained model inside a FastAPI prediction endpoint.

Outcome: Can serve a machine learning model as a callable API service. (Project 4)

PHASE 06 Generative AI & LLM Engineering

Sessions 44–52

Purpose: Language intelligence — LLM architecture, APIs and prompt engineering.

**DAY
44****Generative AI Fundamentals**

Topics: What GenAI is, model types, use cases

Lab: Compare outputs of different generative tasks and map them to real use cases.

Outcome: Understands what generative AI is and where it applies.

**DAY
45****LLM Architecture**

Topics: Tokens, embeddings, context windows, transformers (conceptual)

Lab: Inspect tokenization and context window behavior for sample prompts.

Outcome: Understands the core architecture concepts behind large language models.

DAY
46

Working with LLM APIs

Topics: API setup, requests, responses, streaming

Lab: Call an LLM API from Python and handle both standard and streaming responses.

Outcome: Can integrate an LLM API into a Python application.

DAY
47

Prompt Engineering Fundamentals

Topics: Instructions, roles, context, examples

Lab: Write and iterate on prompts to improve output quality on a defined task.

Outcome: Can design effective prompts for common LLM tasks.

DAY
48

Advanced Prompt Engineering

Topics: Few-shot prompting, chain-of-thought, prompt templates

Lab: Build reusable, parameterized prompt templates for a multi-step task.

Outcome: Can apply advanced prompting techniques to complex tasks.

DAY
49

Structured Outputs and JSON

Topics: Schema-constrained generation, parsing model output

Lab: Constrain an LLM to return structured JSON and parse it reliably in Python.

Outcome: Can get and consume reliable structured output from an LLM.

DAY
50

LLM Parameters and Model Selection

Topics: Temperature, top-p, max tokens, model trade-offs

Lab: Compare model outputs across different parameter settings for the same prompt.

Outcome: Can tune LLM parameters and select models appropriate to a task.

DAY
51

LLM Application Patterns

Topics: Summarization, extraction, classification, generation patterns

Lab: Implement three common LLM application patterns using a shared codebase.

Outcome: Can recognize and implement standard LLM application patterns.

DAY
52

GenAI Mini Project **PROJECT 5**

Topics: Applying patterns end-to-end

Lab: Build a small end-to-end LLM-powered application combining prompting and structured output.

Outcome: Can independently build a working GenAI application. (Project 5)

PHASE 07

Embeddings, RAG & LangChain

Sessions 53–62

Purpose: Knowledge retrieval — grounding LLM responses in real data.

DAY
53

Embeddings

Topics: What embeddings are, similarity, use cases

Lab: Generate embeddings for a set of documents and compute similarity scores between them.

Outcome: Understands how embeddings represent meaning numerically.

**DAY
54****Vector Databases**

Topics: Storage, indexing, similarity search

Lab: Store embeddings in a vector database and run similarity search queries.

Outcome: Can store and retrieve embeddings using a vector database.

**DAY
55****Document Processing and Ingestion**

Topics: Loaders, parsing, metadata

Lab: Build an ingestion pipeline that loads and parses documents from multiple formats.

Outcome: Can build a document ingestion pipeline for a knowledge base.

**DAY
56****Chunking Strategies**

Topics: Fixed-size, semantic, overlap trade-offs

Lab: Apply different chunking strategies to the same document and compare retrieval quality.

Outcome: Can choose and implement appropriate chunking strategies.

**DAY
57****RAG Architecture**

Topics: Retrieval + generation pipeline design

Lab: Design a RAG pipeline architecture diagram and map it to concrete components.

Outcome: Understands how retrieval and generation combine in a RAG system.

**DAY
58****Build a Basic RAG Application**

Topics: End-to-end retrieval-augmented generation

Lab: Build a working RAG application that answers questions from a custom document set.

Outcome: Can build a functioning RAG application from scratch.

**DAY
59****RAG with LangChain**

Topics: Chains, retrievers, prompt integration

Lab: Rebuild the RAG pipeline using LangChain's chains and retriever abstractions.

Outcome: Can use LangChain to structure RAG applications.

**DAY
60****Advanced Retrieval**

Topics: Hybrid search, re-ranking, metadata filtering

Lab: Improve retrieval quality using hybrid search and metadata filtering.

Outcome: Can apply advanced retrieval techniques to improve RAG accuracy.

**DAY
61****RAG Quality and Troubleshooting**

Topics: Common failure modes, debugging retrieval and generation

Lab: Diagnose and fix retrieval and generation failures in a deliberately broken RAG pipeline.

Outcome: Can identify and resolve common RAG quality issues.

**DAY
62****Enterprise Knowledge Assistant Project****PROJECT 6**

Topics: Full RAG system on a real document set

Lab: Build a complete knowledge assistant over a multi-document enterprise-style dataset.

Outcome: Can independently deliver a production-style RAG application. (Project 6)

PHASE 08 Agentic AI & LangGraph

Sessions 63–71

Purpose: Autonomous reasoning — agents that plan, call tools and act.

DAY 63

Introduction to Agentic AI

Topics: What agents are, agent vs chatbot vs workflow

Lab: Compare a scripted chatbot, a workflow and an agent on the same task.

Outcome: Understands what distinguishes an agent from other AI application types.

DAY 64

Agent Components

Topics: Goals, reasoning, tools, memory, actions

Lab: Map the components of an agent architecture onto a real business task.

Outcome: Understands the core building blocks of an AI agent.

DAY 65

Tool Calling

Topics: Function/tool schemas, invocation, results handling

Lab: Give an LLM access to custom Python functions as callable tools and handle outputs.

Outcome: Can implement tool calling for LLM-driven applications.

DAY 66

Build a Basic AI Agent

Topics: Single-agent loop with tools

Lab: Build a working single agent that reasons, calls tools and returns a final answer.

Outcome: Can build a functioning AI agent from scratch.

DAY 67

Agent Memory and Context

Topics: Short-term memory, conversation state, context management

Lab: Add conversational memory to the Day 66 agent and test multi-turn interactions.

Outcome: Can manage context and memory across agent interactions.

DAY 68

LangGraph Fundamentals

Topics: Graph-based orchestration, nodes, edges, state

Lab: Build a simple stateful graph in LangGraph with multiple nodes and transitions.

Outcome: Understands graph-based orchestration for AI workflows.

DAY 69

Multi-Step Agents

Topics: Conditional routing, branching logic, loops

Lab: Build a multi-step LangGraph agent with conditional routing based on intermediate results.

Outcome: Can design agents that reason across multiple steps.

DAY 70

Multi-Agent Systems

Topics: Agent collaboration, role specialization, coordination

Lab: Build a small multi-agent system where specialized agents collaborate on a shared task.

Outcome: Understands how to coordinate multiple specialized agents.

**DAY
71**
Agentic AI Project **PROJECT 7**
Topics: Complete agent workflow — tools + memory + routing

Lab: Build a complete agentic workflow combining tools, memory and conditional routing.

Outcome: Can independently design and build a multi-step agentic application. (Project 7)

PHASE 09
Model Context Protocol (MCP)
Sessions 72–74
Purpose: Standardized integration between agents and external systems.

**DAY
72**
MCP Fundamentals
Topics: What MCP is, why standardized context matters

Lab: Explore how MCP standardizes the connection between AI applications and external tools/data.

Outcome: Understands the purpose and structure of the Model Context Protocol.

**DAY
73**
MCP Servers, Tools, Resources and Prompts
Topics: MCP building blocks

Lab: Build a simple MCP server exposing custom tools and resources.

Outcome: Can build an MCP server that exposes tools and data to an AI client.

**DAY
74**
MCP Integration with AI Agents **PROJECT 8**
Topics: Connecting agents to MCP servers

Lab: Connect the Day 71 agent to an MCP server and use its tools within the agent workflow.

Outcome: Can integrate MCP-based tools into an agentic application. (Project 8)

PHASE 10
AI Evaluation
Sessions 75–77
Purpose: Quality assurance for LLM, RAG and agent outputs.

**DAY
75**
LLM Evaluation Fundamentals
Topics: Quality dimensions, evaluation approaches, metrics

Lab: Define evaluation criteria and score a set of LLM outputs against them.

Outcome: Understands how to define and measure LLM output quality.

**DAY
76**
RAG and Agent Evaluation
Topics: Retrieval quality, groundedness, task success

Lab: Evaluate a RAG pipeline and an agent workflow using targeted quality checks.

Outcome: Can evaluate RAG and agent systems beyond simple LLM output quality.

**DAY
77**
Evaluation Frameworks and Test Datasets
Topics: Building test sets, automated evaluation runs

Lab: Build a small evaluation test dataset and run it against the capstone components.

Outcome: Can build repeatable evaluation processes for AI applications.

PHASE 11 LLMOps & Production AI

Sessions 78–79

Purpose: Operating AI systems responsibly: logging, monitoring, deployment.

DAY
78

LLMOps Fundamentals

Topics: Lifecycle, logging, monitoring, tracing

Lab: Add logging and tracing to an existing LLM/agent application to capture its behavior.

Outcome: Understands the operational lifecycle of LLM-based applications.

DAY
79

Production AI Systems

Topics: Deployment, versioning, cost monitoring, security, observability

Lab: Review deployment, versioning and cost/security considerations for a production AI system.

Outcome: Understands what is required to operate an AI system responsibly in production.

PHASE 12 Final Capstone

Sessions 80

Purpose: Integration demonstration across every layer of the skill stack.

DAY
80

End-to-End AI Engineer Capstone Demonstration **CAPSTONE**

Topics: Full system walkthrough and review

Lab: Present the complete Enterprise AI Knowledge & Task Automation Agent, demonstrating architecture, functionality and evaluation.

Outcome: Demonstrates the ability to design, build and explain an end-to-end AI engineering system. (Final Capstone)

Projects you will build yourself

Every major phase closes with a working project you build, break, debug and improve — ending with an enterprise AI agent that integrates every layer of the stack.

- **Project 1 · Day 14** — Build an end-to-end Python application that pulls data from an API, processes and outputs results.
- **Project 2 · Day 21** — Connect to a database from Python, run parameterized queries, load results into Pandas.
- **Project 3 · Day 38** — Build a complete ML pipeline from raw data to a saved, reusable trained model.
- **Project 4 · Day 43** — Wrap the Day 38 trained model inside a FastAPI prediction endpoint.
- **Project 5 · Day 52** — Build a small end-to-end LLM-powered application combining prompting and structured output.
- **Project 6 · Day 62** — Build a complete knowledge assistant over a multi-document enterprise-style dataset.
- **Project 7 · Day 71** — Build a complete agentic workflow combining tools, memory and conditional routing.
- **Project 8 · Day 74** — Connect the Day 71 agent to an MCP server and use its tools within the agent workflow.
- **Capstone · Day 80** — Present the complete Enterprise AI Knowledge & Task Automation Agent, demonstrating architecture, functionality and evaluation.

Tools and frameworks

Widely adopted, general-purpose tools rather than one proprietary provider — the same ones used in production AI teams.

Python

Pandas & NumPy

SQL

Git & GitHub

REST APIs

Scikit-learn

FastAPI

LLM APIs

Prompt engineering

Embeddings & vector databases

LangChain

LangGraph

Model Context Protocol (MCP)

Evaluation & test sets

Logging & monitoring

Deployment

Starting point

- Basic programming exposure or none at all
- Limited AI understanding
- Knowledge from disconnected tutorials
- Limited backend experience
- No agent architecture experience

Capabilities developed by Day 80

- Python engineering and machine learning foundations
- LLM application development and RAG engineering
- Agentic workflow design and MCP integration
- AI evaluation practice and LLMOps awareness
- An end-to-end AI project, built and explained

Frequently asked questions

Is this suitable for beginners?

Yes. The program begins with Python fundamentals on Day 1 and builds up progressively. A learner comfortable with logical thinking can start with no prior AI experience.

How much Python is required beforehand?

None. Days 1–15 build Python from the ground up — data structures, functions, OOP and data handling with Pandas and NumPy — before any AI topic is introduced.

Is machine learning included?

Yes. Phase 4 (Sessions 28–38) covers the ML workflow, regression, classification, decision trees, ensemble learning, unsupervised learning, evaluation and feature engineering, ending in an end-to-end ML project.

Will we build RAG applications and AI agents?

Yes. Phase 7 (Sessions 53–62) covers embeddings, vector databases, chunking, RAG architecture, LangChain and advanced retrieval; Phase 8 (Sessions 63–71) covers agent components, tool calling, memory, LangGraph and multi-agent systems.

What is MCP?

The Model Context Protocol is a standardized way for AI applications to connect to external tools, data and resources. It is introduced in Phase 9 (Sessions 72–74), after agents and tool calling are established.

What are the timings?

80 live sessions of one hour each, live on Zoom. The days, timing and start date of the next batch are to be confirmed. Every session is recorded; paid students get the recordings for 1 year from the date their fee is paid in full.

What happens after the 80 sessions?

You finish with a complete, explainable end-to-end AI engineering project and a documented skill progression across the full stack, plus mock interviews, resume guidance, interview preparation and certification guidance.

Your next step: register your interest

Collecting interest — next batch date to be confirmed.

Register your interest on the course page and tell us the days and times that suit you. We'll contact you about suitable batch options. Registering interest does not reserve a seat or confirm a batch, and a demo registration is confirmed only when you receive a separate confirmation for that session.

Register your interest → training.javitatech.com/become-an-ai-engineer/

WHATSAPP

+91 87122 87999

EMAIL

info@javitatech.com

WEBSITE

javitatech.com

Live online course (Zoom), sessions in IST · revision recordings for paid students, 1 year from full fee payment · JavitaTech Certificate of Completion · refund policy at javitatech.com/cancellation-and-refund · JavitaTech is a brand of Balakishore Enterprises, Hyderabad.