Corporate responsibility programme · Youth employability

Build the agent. Ship the stack. Prove it works.

A 60-hour hands-on academy, three teaching days a week for four weeks, that takes a mixed-ability cohort from their first model call to a deployed agentic application with retrieval, guardrails, an evaluation harness, and a portfolio a hiring manager can open.

60 contact hours · 11 training days · 4 weeks, on premises · 3 teaching days a week · Hackathon & live demo day

Eleven days, at a glance

60 hrs across 4 weeks

Each tile is a training day, coloured by thematic strand.
Pick one to jump to it.

1 6h	2 5h	3 5h	4 6h	5 5h	6 5h
7 5h	8 6h	9 5h	10 8h	11 4h	

Python & LLM engineering 11h · Data pipelines & RAG 11h · Agentic AI & LangGraph 15h · Evaluation & guardrails 6h · Production & delivery 17h

Curriculum

Eleven days across five thematic strands, three teaching days a week with two days in between for team work. Each day ends with something that runs and gets committed. The arc is deliberate: nothing is deployed before it can be measured, and nothing is measured before it works.

Week 1 Foundations & the model layer

16 hrs · 3 days

Build the engineering bedrock: a well-structured Python environment, disciplined prompt design, and document ingestion that turns a pile of files into something a model can actually retrieve from.

1
DAY

Python engineering & the LLM ecosystem

6 hrs

Python & LLM engineering

Set up a production-quality environment and make the model behave predictably before writing any agent code.

- Project structure with uv; dependencies, configuration, .env and secrets
- Strong typing, dataclasses and Pydantic schemas for structured data
- The framework landscape in one session: LangChain and LangGraph, agent runtimes, hosted agent services
- What each layer actually owns: model client, orchestration, runtime harness, hosting
- Models and clients: OpenAI, Azure OpenAI, local runtimes
- First model call and LCEL; tokens, context windows and rate limits
- Structured output and validation; retry on malformed responses
- Tool calling mechanics: the @tool decorator and bind_tools

Ships: a CLI utility that turns unstructured text into schema-validated JSON, committed and pushed.

2
DAY

Prompt engineering & model control

5 hrs

Python & LLM engineering

The craft layer that separates an experimental notebook from a system somebody else can rely on, plus the discipline for working alongside AI coding agents.

- API parameters measured rather than guessed: temperature, top_p, max_tokens
- System roles, instruction hierarchy and persona control
- Patterns: prompt chaining, few-shot, chain-of-thought, ReAct as a concept
- Prompt templates and versioning; why prompts belong in version control
- Failure modes: hallucination, instruction drift, format collapse
- Working with AI coding agents: spec-first prompting, reading the diff, owning everything you commit
- Lab: A/B two prompts against a fixed 20-example test set and report the winner

Ships: a versioned prompt library and a comparison table with numbers in it. Assessed.

3
DAY

Document processing, embeddings & vector stores

5 hrs

Data pipelines & RAG

Turn raw documents into AI-ready data. Most retrieval systems are won or lost here, several steps before the model is involved.

- Parsing PDF, DOCX and HTML; tables, multi-column layouts, embedded images
- Chunking strategies: fixed-size with overlap, recursive, semantic, and how to choose between them
- Metadata enrichment, deduplication and lineage
- PII detection at ingestion time
- Embeddings: dimensionality, similarity metrics, batching at scale
- Vector stores: ChromaDB and pgvector locally, Azure AI Search hosted, with HNSW and hybrid indexes

Ships: a personal corpus indexed and searchable with working metadata filters.

Two protected days at the end of each week. Self-directed work on the running example, roughly 4 hrs, plus 2 hrs of mentor office hours. Not counted in the 60.

Week 2 Grounded generation & first agents

16 hrs · 3 days

Finish the retrieval application and measure it, then graduate from single prompts to agents that reason, act and remember. The team project brief lands at the start of the week so everything after it has somewhere to go.

4
DAY

RAG pipelines & evaluation

6 hrs

Data pipelines & RAG Evaluation & guardrails

A complete pipeline from ingestion to cited response, and the measurement loop that tells you whether any change made it better. The two are taught together so no one learns to tune retrieval by feel.

- Naive RAG end to end: ingest, chunk, embed, index, retrieve, generate with citations
- Retrieval failure analysis: reading what came back and diagnosing why it was wrong
- Advanced retrieval: query rewriting, hybrid search with BM25 and dense fusion, MMR, cross-encoder re-ranking
- RAGAS: faithfulness, answer relevance, context recall and precision
- Building a small golden dataset that reflects real questions
- LLM-as-judge: methodology, bias and limits
- Team project hand-out, brief and mandatory scope review

Ships: a citing RAG application with an evaluation set and a baseline quality report. Assessed.

5 DAY — LangGraph foundations & core agent patterns

5 hrs

Agentic AI & LangGraph

Learn to think in graphs rather than scripts, and instrument the thing from the first commit.

- Why graphs: control flow, inspectability, resumability
- State, nodes, edges; the first linear graph; conditional routing
- Tool nodes and the ReAct loop, implemented rather than described
- Memory: conversational state, short-term versus long-term patterns
- Persistence and checkpointers on Postgres
- Testing nodes and tools in isolation
- Agentic observability: tracing with Langfuse or LangSmith from day one, not after the bug

Ships: a stateful, tool-using agent that survives a multi-turn conversation.

6 DAY — Multi-agent patterns, runtimes & observability

5 hrs

Agentic AI & LangGraph

Scale to a topology with more than one agent in it, and make the whole thing debuggable by somebody who did not write it.

- Parallelisation and orchestrator-workers; planning agents
- Supervisor and multi-agent network patterns; subgraphs and modular design
- Agent runtimes in practice: the harness, the virtual filesystem, sandboxed execution and sub-agent delegation, shown on the Deep Agents SDK with the Claude Agent SDK as a second reference point
- Context management: summarisation and offloading; persistent memory on pgvector
- Streaming: watching graph execution in real time
- Reading a bad trace: attributing latency and finding the step that went wrong

Ships: a supervisor agent with delegation, traced end to end. Assessed.

Two protected days at the end of the week. Team project work, roughly 4 hrs, plus 2 hrs of mentor office hours.

Week 3 Tools, guardrails & shipping

16 hrs · 3 days

Connect the agent to real tools, make it safe enough to put in front of a stranger, then wrap it in something a user can open and an employer can deploy.

7 DAY Tools over MCP & agentic RAG

5 hrs

Agentic AI & LangGraph Data pipelines & RAG

The protocol reshaping how agents reach tools, and retrieval promoted from a fixed pipeline to something the agent decides when to use.

- MCP architecture: hosts, clients and servers, and what the protocol actually standardises
- Consuming external tools through an MCP server
- Building and exposing your own tool as a server
- Agentic RAG: retrieval as a tool the agent calls, retries and reformulates
- Advanced retrieval inside an agent: self-query, MMR, re-ranking
- Agent-to-agent protocols in outline: what A2A adds and when you would reach for it
- Knowledge-graph retrieval with Neo4j, demonstrated rather than built

Ships: an MCP server plus an agent that retrieves through it.

8
DAY

Guardrails, safety & agent evaluation

6 hrs

Evaluation & guardrails

Taught as principles first, because libraries change and principles transfer: defend in depth, grant tools least privilege, validate at every boundary, fail closed, keep a human on irreversible actions, treat observability as a control, and assume every input is hostile. Evaluation sits in the same day because a guardrail you cannot measure is a guess.

- Where guardrails sit: input, output, tool-use and action layers
- Deterministic validation versus model-based classification, and when each fails
- PII detection and redaction in the request and response path
- Scope enforcement and graceful refusal that does not leak the system prompt
- Prompt injection: direct, indirect through retrieved documents, and tool poisoning
- Red-teaming lab: attack a neighbouring team's agent, then harden your own
- Human in the loop: approval nodes, interrupts and audit trails
- Why agent evaluation is harder than RAG evaluation: trajectories, non-determinism, partial credit
- Metric families: task success, step correctness, tool-call accuracy, steps and latency
- Regression testing in CI so a prompt change cannot silently break the system
- Governance context: model cards, risk registers, EU AI Act obligations in outline

Ships: an agent with layered guardrails, an audit trail, and an evaluation harness scoring at least 15 cases as a pipeline gate. Assessed.

9
DAY

Ship it: service layer, interface & deployment

5 hrs

Production & delivery

An agent in a notebook is a demo. An agent behind an API with auth and streaming, in a container, on a URL, is a product. The day runs on a provided scaffold: participants wire their own agent into it rather than building the plumbing from scratch, which is what makes one day enough.

- FastAPI: routing, request and response schemas, validation, error handling
- Exposing an agent: synchronous calls and token streaming over SSE
- Conversation persistence: threads, messages and tool calls
- Authentication in outline: API keys, OAuth2 and JWT; secrets in Azure Key Vault
- Chat interface with Chainlit: message history, tool-call display, citations, approval steps
- Containerising the stack: Dockerfile and docker-compose with database and vector store
- CI/CD with GitHub Actions: lint, test, build and deploy stages
- Deploying to Azure Container Apps, with hosted agent services as the alternative and what each gives up
- Monitoring and reading production logs after the first real user

Ships: the team agent deployed to a public URL, reproducible from a clean clone. Assessed.

Two protected days at the end of the week, and the final scope freeze. Teams confirm what they are not building.

Week 4 Build & present

12 hrs · 2 days

Everything converges into one system, built under mentorship and demonstrated live to evaluators. The lighter week is deliberate: teams need room to finish rather than more material to absorb.

10
DAY

Hackathon: full-stack agentic system

8 hrs

Mentored build sprint

Teams assemble the whole thing: an agent with retrieval, layered guardrails, an evaluation gate, an MCP tool, an API and a working interface.

- Mentored build sprint with a mentor per two teams
- Integration checkpoint at the halfway mark
- Architecture diagram and README written for a reader who was not there
- Presentation rehearsal and demo failure planning

Ships: a deployed system, a repo, an evaluation report and a rehearsed demo.

11
DAY

Demo day

4 hrs

Assessment

Each team demonstrates a live application and defends its design decisions under questioning.

- Live demos to evaluators from the partner organisations, Accenture and Code.Hub
- Technical question and answer on architecture and trade-offs
- Individual portfolio review and hiring-partner introductions
- Closing remarks and certification

Ships: the demo recording and the finished portfolio bundle.

Why four weeks at 60 hours

The same 60 contact hours delivered over four weeks rather than three, at three teaching days a week. The extra calendar week buys practice time, not more material.

Shorter teaching days

Most days now run 5 hours instead of 6. Retention on hands-on technical content falls off sharply in the last hour of a long day, and a shorter day leaves participants able to keep working

afterwards.

Two protected days every week

Each week has two days for self-directed and team work with mentor office hours. That is where the project actually gets built, and it is where a mixed-ability cohort catches up without contact hours being spent on levelling.

MCP gets its own day again

Spreading to eleven days rather than ten released one teaching day. It went to Day 7, so tool interoperability and agentic RAG are built rather than compressed into the multi-agent day.

Week 4 is deliberately light

Two days: the hackathon and demo day. Teams arrive with a deployed system from Day 9 and spend the week finishing rather than starting.

Attendance and access

Three days a week is easier to sustain for participants juggling study or part-time work, which matters for the employability mandate. The trade-off is a longer commitment window and a higher risk of drift between sessions, which the office hours are there to absorb.

Scoped to 60 hours

Forty hours came out of the original design. Cutting evenly would have produced a tour of everything and command of nothing, so the reductions are concentrated where a junior candidate can recover on their own, and the days that generate portfolio evidence are protected.

Protected in full

The retrieval application, the agent, MCP tools, guardrails, evaluation, deployment, the hackathon and demo day. Each of these produces something a hiring manager can open, which is the point of the programme.

Merged rather than dropped

RAG patterns and RAG evaluation share Day 4. Multi-agent patterns and agent runtimes share Day 6. Guardrails and agent evaluation share Day 8. The service layer, the interface and deployment share Day 9, run from a provided scaffold.

Reduced to a survey

The framework comparison is a single session on Day 1 instead of a runtime lab: participants build on one stack and can describe the others. Agent-to-agent protocols are explained rather than implemented. Knowledge-graph retrieval and Databricks ingestion are demonstrated, not built.

Moved outside contact hours

The dedicated day on AI-assisted coding is gone as a standalone; the practice is now a strand inside Day 2 and used throughout the hackathon. Docker and cloud basics move into the pre-programme bridge, which grows to roughly 10 hours.

What this changes about the outcome

Graduates leave with the same artefacts but less depth in multi-agent architecture. The honest framing for hiring partners is a capable junior who has shipped one full system, rather than one who has built several.

Where the hours go

Five thematic strands run through the eleven days. Production and delivery carries the most time because a deployed system is the deliverable, not a slide about one. Pick a strand to see only its days.

Production & delivery — 17 hrs

The service layer, the interface, containers, deployment, the hackathon and demo day.

Agentic AI & LangGraph — 15 hrs

Graphs and state, multi-agent patterns, agent runtimes, MCP tools and agentic RAG.

Python & LLM engineering — 11 hrs

Environment, structured output, tool calling, prompt design and model control.

Data pipelines & RAG — 11 hrs

Parsing, chunking, embeddings, vector stores, the full retrieval pipeline and its measurement.

Evaluation & guardrails — 6 hrs

Guardrail principles, red-teaming, trajectory evaluation and regression testing in CI.

The framework landscape

At 60 hours participants learn one framework properly and the rest well enough to argue about. Employers use different stacks, so the transferable skill is recognising the same four jobs wherever they surface: model client, orchestration, runtime harness and hosting. Pick one to see where it sits.

LangChain · The teaching spine

Graph-based orchestration with explicit state, nodes and edges. The most mature option for production agents today, and the one every other framework gets compared against. This is the stack participants actually build on.

Reach for it when	State machines you can draw, inspect and resume. Checkpointers give you durable conversations without extra work.
The trade-off	Verbose. The abstraction layers take a week to stop fighting you.
Taught on	Day 1, Day 5, Day 6, Day 7

LangChain · Runtime harness

A harness that sits above LangGraph: built-in tools, a virtual filesystem, context offloading and sub-agent delegation without wiring each one yourself. Used to demonstrate what a runtime gives you for free.

Reach for it when	Long-running research and coding agents where context management is the hard part.
The trade-off	Opinionated. You inherit its choices about how an agent should be shaped.
Taught on	Day 6

Anthropic · Runtime harness

The runtime behind Claude Code, exposed as a library. Sessions, permission modes, an allowed-tools boundary, lifecycle hooks and sub-agents. Shown as a

second reference point so the concepts stay portable, and used as a coding assistant on Day 2.

Reach for it when	Agents that touch a filesystem or shell, and anywhere the tool boundary is a security control rather than a convenience.
The trade-off	Single-vendor by design. Portability is a deliberate trade.
Taught on	Day 2, Day 6

Microsoft | Enterprise runtime

The successor to Semantic Kernel and AutoGen from the same teams, merging AutoGen's multi-agent abstractions with Semantic Kernel's enterprise features and adding graph workflows for explicit orchestration. MCP and A2A are native rather than bolt-on. Surveyed rather than built on at this length.

Reach for it when	Enterprise work: .NET alongside Python, session state, middleware, telemetry, long-term support commitments.
The trade-off	Newer surface than LangGraph, and the ecosystem of examples is still thinner.
Taught on	Day 1 as part of the landscape session

Open protocols | Interoperability

MCP standardises how an agent reaches tools and data; A2A standardises how agents reach each other. MCP is built hands-on across a full day. A2A is explained and demonstrated, not implemented, which is the clearest casualty of the shorter format.

Reach for it when	Writing a tool once and using it from every framework in this list.
The trade-off	A protocol is not a permission model. Everything MCP exposes still needs guardrails.
Taught on	Day 7, Day 8

Microsoft | Hosting

Managed hosting for agents on Azure: deployment, identity, content filters and monitoring handled by the platform rather than your container.

Reach for it when	Getting an agent in front of users without owning the runtime, and inheriting Azure's governance posture.
The trade-off	Less control over the loop. Portability out is real work.
Taught on	Day 9

Reference architecture

Every team builds against the same shape. Naming it on Day 1 means Day 9 is an integration exercise rather than an architecture argument. Pick a layer to see which days teach it.

Solid lines carry requests. Dashed lines carry evidence: traces and scores, which is what makes the system reviewable by someone who did not build it.

Two outcome tracks

Both tracks attend all eleven days. They diverge in project brief, stretch labs and how the final work is weighted, because splitting delivery would halve the instructor ratio for no teaching benefit. Participants choose after Day 4.

	Track A · AI & Agent Engineer	Track B · GenAI practitioner
Project shape	Multi-agent system with tool integration, delegation and a deployed API	One narrow domain problem solved properly, end to end
Stretch labs	Supervisor topologies, MCP server authoring, latency reduction	Retrieval tuning, evaluation set design, interface iteration
Judged mainly on	System design, agent architecture, observability depth	Applied outcome, measurement rigour, usability
Typical first roles	AI Engineer, Agent Engineer, junior AI platform engineer	AI solutions associate, GenAI analyst, AI product associate

Before Day 1

A self-paced bridge released three weeks ahead, roughly 10 hours, outside the 60. Completion is a condition of attendance. With fewer contact hours, this is what keeps a mixed-ability cohort workable without spending teaching time levelling.

Python refresher

Functions, classes, type hints, comprehensions, error handling.

Tooling

uv project setup, virtual environments, editor configuration, .env files.

Git essentials

Clone, branch, commit, push, open a pull request.

Docker and cloud basics

Images, containers and one deployed hello-world service. Moved here from contact hours so Day 9 can stay on the agent rather than the plumbing.

Setup gate

Run the provided template repository, make one authenticated model call, commit the output. Cleared by a green check on the participant's own repo plus a short diagnostic. Three online mentor sessions support it.

Assessment

Weighted towards the artefact rather than the exam, because the artefact is what gets someone hired.

Component	Weight	Form
Pre-programme gate	Pass / fail	Setup verification and diagnostic
Assessed deliverables	15%	Days 2, 4, 6, 8 and 9, reviewed against a checklist
Quizzes	10%	End of Weeks 1, 2 and 3
Final project and demo	55%	Rubric: it runs, retrieval quality, agent design, guardrails, measurement, craft, communication
Collaboration	20%	Instructor and peer input across the four weeks

Certification requires the gate cleared, 80% attendance, all assessed deliverables at pass or above, and a project presented on demo day.

What each graduate leaves with

The certificate is not the outcome. This bundle is.

A public repository

Prompt library, RAG application, agent, MCP tool, API and interface, with a README that lets a stranger run it.

An evaluation report

Real measurements on their own system. The rarest thing in a junior AI portfolio, and the fastest way through a screening call.

A deployed application

A live URL and a recorded demo, not a screenshot.

An architecture diagram

One page, drawn by them, defended out loud on demo day.

A skills statement

What they built, mapped to the vocabulary used in AI engineering job adverts.

Hiring-partner exposure

Direct introductions on demo day, with portfolio links in the hands of evaluators.

Plan the cohort

Move the sliders to size the academy. Delivery weeks assume three teaching days a week, staffing follows a one-to-twelve assistant ratio with a floating mentor on the heaviest days, and teams of four are what the hackathon and demo day are built around.

Participants	20
Contact hours	60
Hours per teaching day	5

Below 12 participants the demo-day panel loses its comparative value; above 30 the hackathon needs a second room.

Delivery days	**12**
Delivery weeks	**4**
Lead instructors	**2**

Teaching assistants	**2**
Project teams of 4	**5**
Demo slots at 12 min	**60 min**
Rooms needed	**1**

Delivery requirements

Staffing

One lead instructor plus one teaching assistant per twelve participants. A second floating mentor on Days 4, 7, 8, 9 and 10. Two mentors for the protected days each week.

Cohort

18 to 24 participants. Selection by a 60 to 90 minute screening task rather than CV review, because it predicts completion better and is fairer to non-traditional candidates, which is the point of the mandate.

Rhythm

Three consecutive teaching days a week, same days each week, with the remaining two days protected for team work. Fixed days matter for participants arranging study or shift work around the programme.

Per participant

Laptop with 16 GB RAM, Azure OpenAI or OpenAI access on a shared key, GitHub account, tracing tool on a free tier, an Azure subscription from Week 3.

Provided in advance

Template repository with a devcontainer, pinned dependencies, the setup gate check, the Day 9 service and interface scaffold, and a Codespaces fallback for laptops that fight back.

Model access

Small models for all labs, with a stronger model unlocked for the final project. Usage is capped per participant so one runaway loop cannot exhaust the cohort's access.