

01

AI Landscape & Python for AI

Week 1 · Sunday 15 Nov 2026 · 10:00–15:00 IST · Phase I · AI Foundations

v0.1

GitHub release

LEARNING OBJECTIVES

- › Explain how AI, ML, deep learning, GenAI and agentic AI relate
- › Load, clean and profile a real dataset with NumPy and Pandas
- › Name the main sources of bias and privacy risk in data

1 · FUNDAMENTAL

10:00–11:30

AI, ML, deep learning, GenAI and agentic AI: how they relate; NumPy, Pandas and the statistics you actually use; Ethics primer: bias, privacy and accountability

2 · APPLICATION

11:45–13:15

Clean and profile a real enterprise ticket and document corpus; find quality issues and bias.

3 · PRODUCTION

13:45–15:00

Scaffold the AP-Nexus repo with a data-ingestion module, tests and a README.

KEY TERMS

Agentic AI

AI that pursues a goal by planning and taking actions with tools

Feature

A measurable input column a model learns from

EDA

Exploratory data analysis: summaries and plots before modelling

Data leakage

Test information sneaking into training

Bias

Systematic error that harms a group or skews results

TOOLS THIS WEEK

Python

NumPy

Pandas

Matplotlib

Git & GitHub

SELF-CHECK QUESTIONS

1. What separates an agent from a chatbot?
2. Which three checks do you run on a new dataset?
3. Give one example of sampling bias in ticket data.

HOMEWORK (BEFORE NEXT SUNDAY)

Finish the EDA report and add two data-quality tests to the repo.

DELIVERABLE · PUSH BY 15:00 SUNDAY

```
git commit -m "feat(data): ingestion pipeline + EDA report"
git tag v0.1 && git push origin main --tags
```

MY NOTES

Week 3 · Sunday 29 Nov 2026 · 10:00–15:00 IST · Phase I · AI Foundations

GitHub release

- › Explain backpropagation and gradient descent in plain words
- › Fine-tune a pretrained encoder with PyTorch
- › Serve text embeddings for later weeks

10:00–11:30

Backpropagation, optimisers, regularisation; CNNs, sequence models, embeddings; Transfer learning in PyTorch

11:45–13:15

Fine-tune a small transformer encoder on tickets and compare it with the Week 2 model.

13:45–15:00

Add an embedding service used by every later layer.

Neuron / layer

Weighted sum + activation, stacked into networks

Backpropagation

Passing error gradients back to update weights

Overfitting

Memorising training data; fixed by regularisation

Transfer learning

Reusing a pretrained model for a new task

Embedding

A vector that captures meaning of text

PyTorch

Hugging Face Transformers

sentence-transformers

SELF-CHECK QUESTIONS

1. What does a learning rate control?
2. Why freeze early layers when fine-tuning?
3. How do you compare two embeddings?

HOMEWORK (BEFORE NEXT SUNDAY)

Compare the neural classifier with Week 2 and keep the better one.

DELIVERABLE • PUSH BY 15:00 SUNDAY

```
git commit -m "feat(dl): fine-tuned encoder + embedding service"
git tag v0.3 && git push origin main --tags
```

MY NOTES

Week 4 · Sunday 6 Dec 2026 · 10:00–15:00 IST · Phase II · Generative AI

GitHub release

- › Describe attention and the transformer block
- › Run and benchmark open and hosted LLMs
- › Build a gateway that routes by cost and quality

10:00–11:30

11:45–13:15

13:45–15:00

Token

Unit of text a model reads and writes

Attention

Each token weighs every other token's relevance

Context window

Max tokens a model sees at once

Temperature

Randomness of sampling

Instruction tuning

Training a model to follow prompts

Ollama

Hugging Face

OpenAI / Claude / Gemini APIs

FastAPI

SELF-CHECK QUESTIONS

1. Why do longer prompts cost more?
2. When would you choose a local model?
3. What should a gateway log for every call?

HOMEWORK (BEFORE NEXT SUNDAY)

Add retries, timeouts and a cost report to the gateway.

DELIVERABLE • PUSH BY 15:00 SUNDAY

```
git commit -m "feat(llm): model gateway with routing + logging"
git tag v0.4 && git push origin main --tags
```

MY NOTES

LEARNING OBJECTIVES

- › Write prompts that return validated JSON
- › Apply chain-of-thought, ReAct and self-reflection
- › Detect hallucination and prompt-injection failures

1 • FUNDAMENTAL

10:00–11:30

Structured outputs and schemas;
Chain-of-thought, ReAct, reflection;
Hallucination and prompt-injection
failure modes

2 • APPLICATION

11:45–13:15

Design and A/B test prompt chains for
summarising and classifying tickets.

3 • PRODUCTION

13:45–15:00

Add a versioned prompt library and
reasoning chains with JSON-schema
validation.

KEY TERMS

System prompt

Standing instructions that frame the model's role

Structured output

Responses forced into a schema

ReAct

Reason → act with a tool → observe → repeat

Self-reflection

Model critiques and revises its own draft

Prompt injection

Untrusted text that tries to override instructions

TOOLS THIS WEEK

LangChain

Pydantic

JSON Schema

promptfoo

SELF-CHECK QUESTIONS

1. Why version prompts like code?
2. How does ReAct differ from plain chain-of-thought?
3. Name two defences against prompt injection.

HOMEWORK (BEFORE NEXT SUNDAY)

A/B test two prompt chains and keep the higher-scoring one.

DELIVERABLE • PUSH BY 15:00 SUNDAY

```
git commit -m "feat(chains): prompt library + validated reasoning chains"
git tag v0.5 && git push origin main --tags
```

MY NOTES

Week 6 · Sunday 20 Dec 2026 · 10:00–15:00 IST · Phase II · Generative AI

GitHub release

- › Build a retrieval pipeline over company documents
- › Use hybrid search and re-ranking
- › Measure faithfulness and relevance with RAGAS

10:00–11:30

11:45–13:15

13:45–15:00

RAG

Retrieve relevant passages, then generate a grounded answer

Chunking

Splitting documents into retrievable pieces

Vector database

Stores embeddings for similarity search

Hybrid search

Keyword (BM25) + vector search together

Faithfulness

Is every claim supported by retrieved text?

ChromaDB

FAISS

LlamaIndex

RAGAS

SELF-CHECK QUESTIONS

1. What happens if chunks are too large?
2. Why add a re-ranker?
3. When is fine-tuning better than RAG?

HOMEWORK (BEFORE NEXT SUNDAY)

Add citations to every answer and run RAGAS in CI.

DELIVERABLE · PUSH BY 15:00 SUNDAY

```
git commit -m "feat(rag): cited knowledge service + RAGAS eval"
git tag v0.6 && git push origin main --tags
```

MY NOTES

LEARNING OBJECTIVES

- › Explain the observe → reason → act → learn loop
- › Design safe, well-described tools for function calling
- › Expose tools through an MCP server

1 • FUNDAMENTAL

10:00–11:30

The observe $\rightarrow$ reason $\rightarrow$ act $\rightarrow$ learn loop; Function calling and tool design; Model Context Protocol: servers, tools, resources

2 • APPLICATION

11:45-13:15

Build a LangGraph agent that searches, reads and calls a ticketing API.

3 · PRODUCTION

13:45–15:00

Expose company tools through an MCP server; the first agent goes live in AP-Nexus.

KEY TERMS

Agent

LLM + tools + loop + state, working toward a goal

Function calling

Model chooses a tool and its arguments

MCP

Model Context Protocol: standard way to serve tools and resources

State graph

LangGraph nodes and edges that control the loop

Stop condition

When the agent decides the goal is met

TOOLS THIS WEEK

LangGraph

MCP SDK

FastAPI

SELF-CHECK QUESTIONS

1. What makes a tool description good?
2. How do you stop an agent looping forever?
3. What belongs in agent state?

HOMEWORK (BEFORE NEXT SUNDAY)

Add two new MCP tools and a max-steps guard.

DELIVERABLE · PUSH BY 15:00 SUNDAY

```
git commit -m "feat(agent): MCP tool server + research agent"
git tag v0.7 && git push origin main --tags
```

MY NOTES

Week 8 · Sunday 3 Jan 2027 · 10:00–15:00 IST · Phase III · Agentic AI

GitHub release

- › Compare single-agent and multi-agent designs
- › Build role-based crews with hand-offs
- › Add a supervisor and a critic

10:00–11:30

11:45-13:15

13:45–15:00

Add a triage, research, writer and critic crew under a supervisor.

Agent with one job, e.g. researcher or writer

Agent that routes work and merges results

Passing task and context to another agent

Agent that reviews and requests fixes

Extra tokens and latency from more agents

CrewAI

AutoGen

LangGraph

1. When is one agent enough?

2. How do agents share context safely?

3. What does the critic check?

Run the crew on 20 tickets and log success and cost.

```
git commit -m "feat(crew): supervisor + 4 role agents"
git tag v0.8 && git push origin main --tags
```

MY NOTES

Week 9 · Sunday 10 Jan 2027 · 10:00–15:00 IST · Phase III · Agentic AI

GitHub release

- › Add short- and long-term memory to agents
- › Plan, replan and recover from tool errors
- › Build an evaluation harness that runs on every PR

10:00–11:30

Short- and long-term agent memory;
Planning, replanning and recovery;
Trajectory, success-rate and cost
metrics

11:45-13:15

Add memory, then measure task success before and after with LangSmith traces.

13:45–15:00

Commit an evaluation harness that runs on every pull request.

Short-term memory

Conversation and scratchpad within a task

Long-term memory

Stored facts reused across tasks

Replanning

Changing the plan after a failed step

Trajectory eval

Scoring the steps, not only the final answer

Tracing

Recording every call for debugging

LangSmith

Redis / SQLite

pytest

GitHub Actions

SELF-CHECK QUESTIONS

1. What should never be stored in memory?
2. Which three metrics define agent success?
3. Why evaluate trajectories?

HOMEWORK (BEFORE NEXT SUNDAY)

Make the eval harness fail the build when success drops.

DELIVERABLE • PUSH BY 15:00 SUNDAY

```
git commit -m "feat(eval): memory store + agent eval harness in CI"
git tag v0.9 && git push origin main --tags
```

MY NOTES

10 Deployment & AgentOps

Week 10 · Sunday 17 Jan 2027 · 10:00–15:00 IST · Phase III · Agentic AI

v0.95

GitHub release

LEARNING OBJECTIVES

- › Serve agents behind an API and dashboard
- › Containerise and deploy with CI/CD
- › Monitor latency, cost and approvals

1 · FUNDAMENTAL

10:00–11:30

Serving with FastAPI and Streamlit;
Docker, CI/CD with GitHub Actions;
Cloud, edge, hybrid; monitoring and
cost control

2 · APPLICATION

11:45–13:15

Containerise AP-Nexus and load-test it.

3 · PRODUCTION

13:45–15:00

Deploy to the cloud with a CI/CD
pipeline, dashboard and
human-approval queue.

KEY TERMS

Container

Packaged app + dependencies (Docker)

CI/CD

Automatic test, build and deploy on every change

Observability

Logs, metrics and traces in production

Human-in-the-loop

A person approves risky actions

Rate limit

Cap on requests to control cost and abuse

TOOLS THIS WEEK

Docker

FastAPI

Streamlit

GitHub Actions

Cloud VM / Render / Railway

SELF-CHECK QUESTIONS

1. What must be in the Dockerfile?
2. Which actions need human approval?
3. How do you roll back a bad deploy?

HOMEWORK (BEFORE NEXT SUNDAY)

Load-test the deployed agent and add a cost alert.

DELIVERABLE · PUSH BY 15:00 SUNDAY

```
git commit -m "chore(deploy): docker + CI/CD + monitoring dashboard"
git tag v0.95 && git push origin main --tags
```

MY NOTES

11

Safety, Governance & AI Ethics

Week 11 · Sunday 24 Jan 2027 · 10:00–15:00 IST · Phase III · Agentic AI

v0.99

GitHub release

LEARNING OBJECTIVES

- › Add guardrails, permissions and sandboxing
- › Red-team your own agents
- › Apply EU AI Act, India DPDP Act and NIST AI RMF to your system

1 • FUNDAMENTAL

10:00–11:30

Guardrails, permissions, sandboxing; Red-teaming and prompt-injection defence; EU AI Act, India DPDP Act, NIST AI RMF; fairness and accountability

2 • APPLICATION

11:45–13:15

Red-team your own agents and log every successful attack.

3 • PRODUCTION

13:45–15:00

Add guardrails and permission policies; publish a safety and ethics audit.

KEY TERMS

Guardrail

Check that blocks unsafe input or output

Least privilege

Give each tool only the access it needs

Red-teaming

Attacking your own system to find failures

DPDP Act

India's Digital Personal Data Protection Act, 2023

Accountability

Clear ownership of AI decisions and errors

TOOLS THIS WEEK

Guardrails AI / NeMo Guardrails

promptfoo

Model cards

SELF-CHECK QUESTIONS

1. Which of your tools is most dangerous and why?
2. What personal data does your agent touch?
3. Who is accountable when the agent errs?

HOMEWORK (BEFORE NEXT SUNDAY)

Publish docs/ethics_audit.md with risks, controls and fairness checks.

DELIVERABLE • PUSH BY 15:00 SUNDAY

```
git commit -m "feat(safety): guardrails + red-team + ethics audit"
git tag v0.99 && git push origin main --tags
```

MY NOTES

Week 12 · Sunday 31 Jan 2027 · 10:00–15:00 IST · Phase III · Agentic AI

GitHub release

- › Integrate, document and release AP-Nexus v1.0
- › Present a clear live demo to a panel
- › Turn the project into a portfolio piece or paper

1 • FUNDAMENTAL

10:00–11:30

System design review; Technical writing for portfolios and papers; Research track: paper or patent pathway

2 • APPLICATION

11:45–13:15

Rehearse the demo with mentors and fix the last issues.

3 · PRODUCTION

13:45–15:00

Tag v1.0, publish docs and present live to the faculty and industry panel.

KEY TERMS

Release tag

Named version in Git, e.g. v1.0

README

How to run, use and evaluate the project

System design

Diagram + choices + trade-offs

Demo script

Planned 7-minute walkthrough with a fallback

Portfolio write-up

Problem, approach, results, what you'd do next

TOOLS THIS WEEK

GitHub Releases

Mermaid diagrams

Slides

SELF-CHECK QUESTIONS

1. Can a recruiter run your repo in 5 minutes?
2. What one metric proves your system works?
3. What would you build next?

HOMEWORK (BEFORE NEXT SUNDAY)

Publish the portfolio write-up and share the release link.

DELIVERABLE • PUSH BY 15:00 SUNDAY

```
git commit -m "release: AP-Nexus v1.0 – Demo Day"
git tag v1.0 && git push origin main --tags
```

MY NOTES