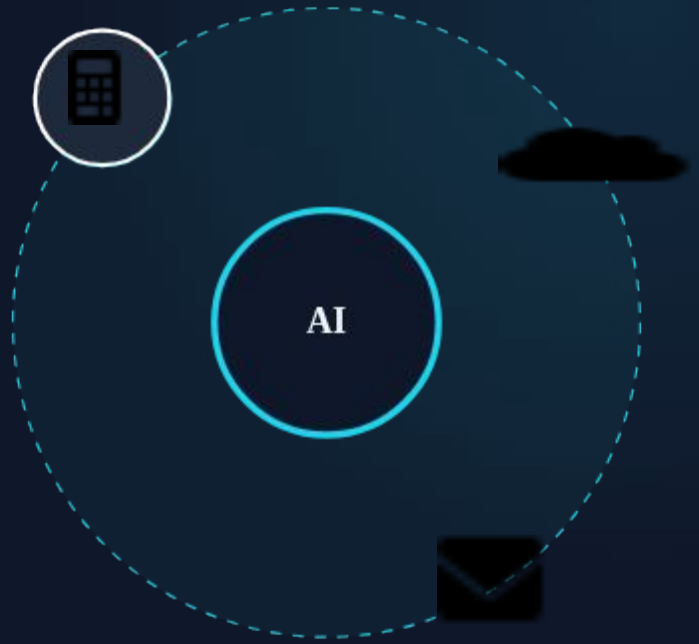
DAY 03 MODULE

The Tools: Function Calling

Giving LLMs the power to act on the real world.



By,

Shikha Tyagi

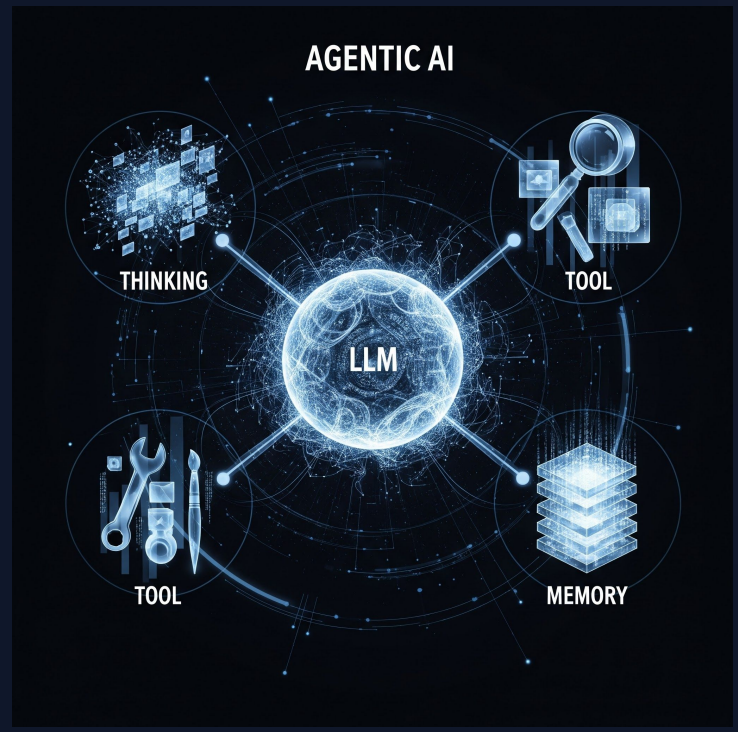
Founder - AI JAMIC (AI Research and Consulting)

Education: IIT Delhi (M.Tech.)

Guidelines

- Attendance is mandatory for all 5 sessions
- Hands on activity is mandatory
- 15 min break at 10:30PM
- QnA session at the end (10-15 min)
- Feel free to drop your questions in chat
- There will be quizzes in-between, drop your answers in chat

5 day roadmap



1

Shift

Agentic Thinking vs. Chatbots

2

Brain

LLM Types & Prompting

3

Hands

Function Calling & Tools

4

Memory

RAG & Vectors

Build

End to end pipeline & Capstone

Detailed Agenda

Module 1: The Concept of Function Calling

Module 2: Understanding Schemas

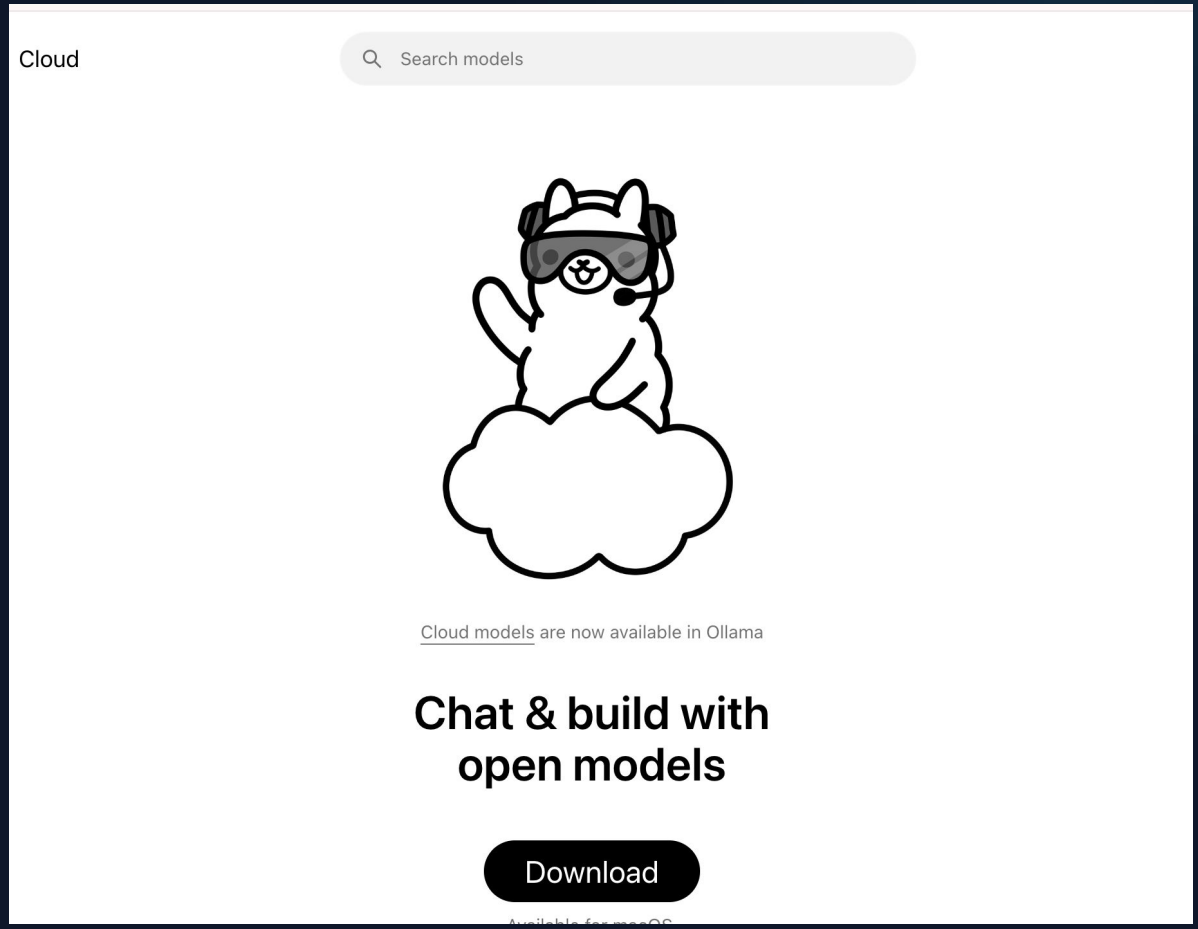
Module 3: The Execution Flow

Module 4: Safety & Error Handling

Module 5: Hands-On Build

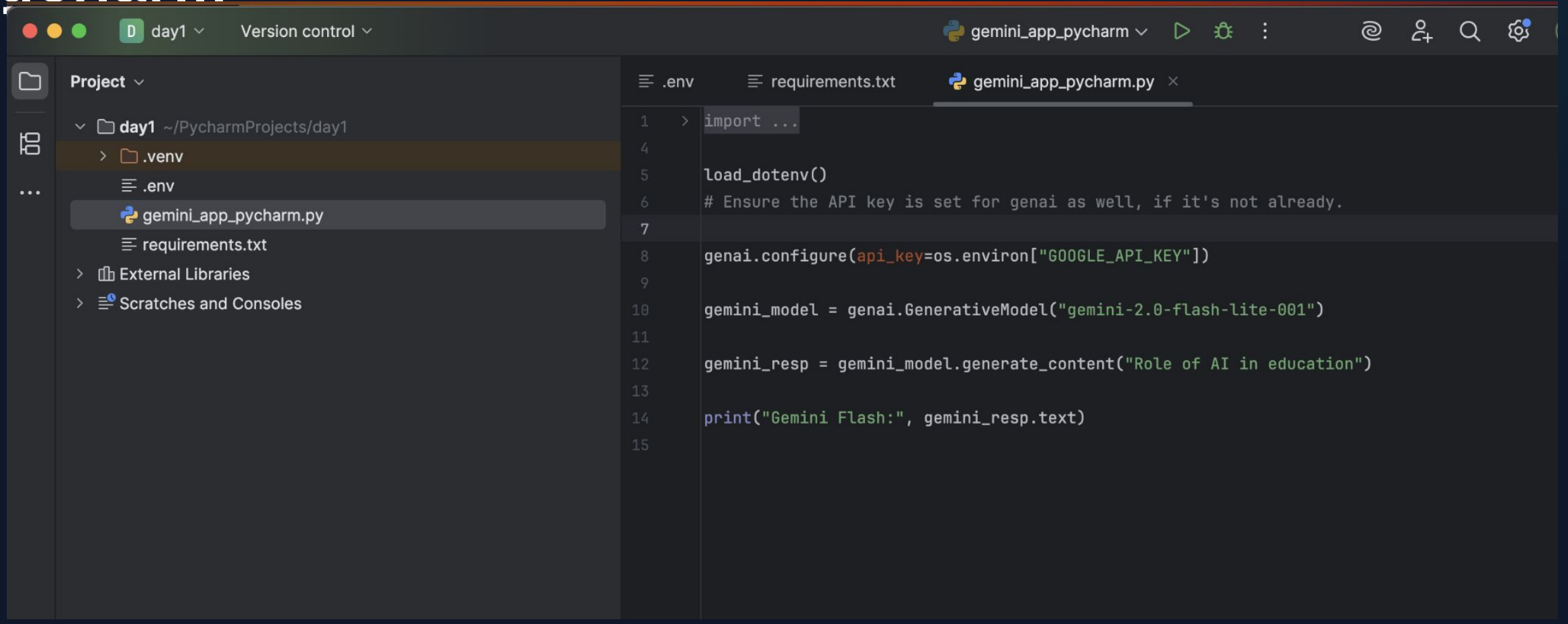
Quick Recap of Downloaded Softwares:

- Ollama



Quick Recap of Downloaded Softwares:

Pucharm



Quick Recap

Quiz-1

Chain of thought prompting is best suited for?

Quick Recap

Quiz-2

Which software you can download to run open source models?

Quick Recap

Quiz-3

Which issues might arise while trying to run open source models?

Quick Activity

Open ChatGPT/Gemini

What is current temperature in delhi?
do not use any tool for external search.

Why do we need Tools?

Math

LLMs are probabilistic, not deterministic. They are bad at math.

Real-Time Data

LLMs are frozen in time (training cut-off). Tools give live stock prices/weather.

Private Data

LLMs don't know your company SQL database.

Module 1
What is Function Calling?

From Text Generators to Action Takers.

The Paradigm Shift

Before (Standard LLM)

Input: "What is current weather in XYZ location"

Process: Use historic data.

Output: (Often hallucinated/wrong).

After (Function Calling)

Input: "What is current weather in XYZ location"

Process: Detect weather intent.

Output: call_weather_api(location)

Quick check

What is NER (named entity recognition)?

- identification of sentiment
- identification of location, name etc
- Identification of article topic

Crucial Concept

LLMs DO NOT EXECUTE CODE

They generate TEXT that LOOKS like code.

They act as a **Translator** between Natural Language (English) and API Language (JSON).

Analogy: The Head Chef

- **LLM = Head Chef:** Knows the recipes (plans), but doesn't chop vegetables.
- **Tools = Prep Cooks:** They chop, fry, and fetch ingredients.
- **The Process:** Chef says "Chop onions!" → Prep Cook chops → Chef adds to pot.

LLM as a Router

The model analyzes the user query and decides:

1. Do I know the answer directly? (e.g., "Hi", "Write a poem")
2. OR do I need to use a tool? (e.g., "What is the weather?")

This "Routing" capability is fine-tuned into models like GPT-4o.

Quick check

What possible arguments weather api will take?

Let’s explore popular APIs

https://www.geeksforgeeks.org/blogs/free-apis-list/

Module 2
The Schema

Defining the "API" for the model.

The Tool Definition (Schema)

To use a tool, we must explain it to the LLM in a format it understands (JSON).

This definition is passed to the model **alongside the user prompt**.

Key Components

- **Name:** Unique identifier.
- **Description:** When to use it.
- **Parameters:** What inputs it needs.

The Description is Critical

Bad Description: `"func_1"`

The LLM has no idea when to call this.

Good Description: `"get_current_weather: Call this when the user asks about temperature or rain in a specific city."`

Explicit instructions on usage context.

Parameter Types

We use standard **JSON Schema** to define arguments.

String

City names, email bodies, queries.

Integer/Float

Numbers, quantities, prices.

Enum

Restricted choices: ["Celsius", "Fahrenheit"]

Required Fields

You can tell the LLM which arguments are optional vs. mandatory.

```
"required": ["location"]
```

If the user says "What is the weather?", the LLM knows it is missing location and will ask the user for it instead of hallucinating.

Example Schema

```
{ "name": "get_stock_price", "description": "Get the current stock price for a given ticker symbol.", "parameters": { "type": "object",
"properties": { "ticker": { "type": "string", "description": "The stock symbol, e.g. AAPL" } }, "required": ["ticker"] } }
```

Quiz: Schemas

Question 1:

Which part of the schema tells the LLM **when** to use the tool?

A. The parameter type
B. The description
C. The function name

Module 3
The Execution Flow

The Lifecycle of an Agentic Action.

The 4-Step Loop

1
Prompt

Send User Query + Tool Schemas to LLM.

2
Decide

LLM stops generating text and generates a "Tool Call" object.

3
Execute

Your Python script runs the function.

4
Synthesize

Feed the result back to LLM for final answer.

Step 1: The Setup

```
messages = [ {"role": "user", "content": "What is 50 * 5?"} ] tools = [multiply_schema] response =
client.chat.completions.create( model="gpt-4o", messages=messages, tools=tools )
```

Step 2: The Decision

The LLM response will contain a tool_calls array.

```
# response.choices[0].message { "content": null, "tool_calls": [ { "id": "call_123", "function": { "name": "multiply",
"arguments": "{\"a\": 50, \"b\": 5}" } } ] }
```

Step 3: The Execution

This happens in **Your Python Code**, not OpenAI.

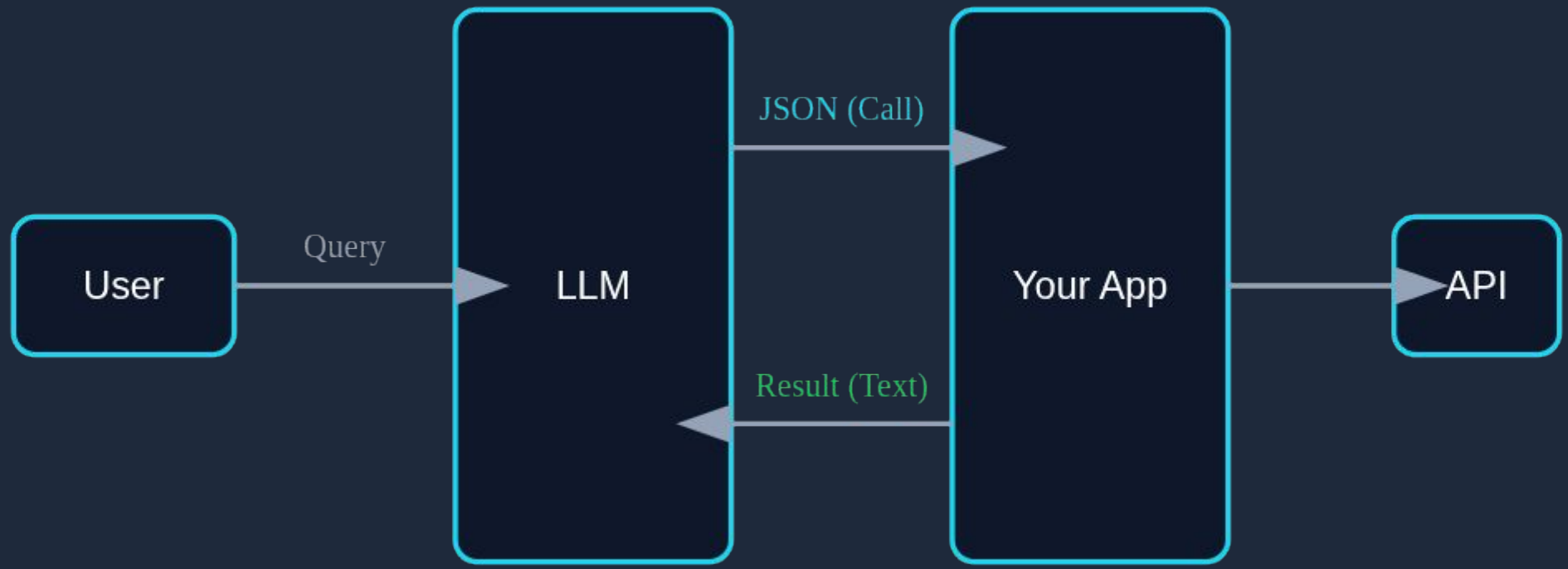
```
import json tool_call = response.choices[0].message.tool_calls[0] args = json.loads(tool_call.function.arguments) #
ACTUALLY RUNNING THE MATH result = args["a"] * args["b"] # result is 250
```

Step 4: The Loop Back

We must tell the LLM what happened.

```
messages.append({ "role": "tool", "tool_call_id": "call_123", "content": "250" }) # Call LLM again to get the final
sentence final_response = client.chat.completions.create(...) # "The answer is 250."
```

Visualizing the Loop



Module 4
Advanced Concepts

Parallel calls, Safety, and Errors.

Providing Multiple Tools

You can give the LLM a swiss-army knife.

```
tools = [ get_weather_schema, get_stock_price_schema, send_email_schema ]
```

The LLM will automatically choose the right one, or none at all.

Parallel Function Calling

The Scenario

"What is the weather in Tokyo, Paris, and London?"

The Result

GPT-4o will output **3 tool calls** in a single response.

```
tool_calls: [ {name: "weather", args: "Tokyo"}, {name:
"weather", args: "Paris"}, {name: "weather", args:
"London"} ]
```

What is Human-in-the-Loop?

https://cloud.google.com/discover/human-in-the-loop?hl=en

Human-in-the-Loop

DANGER ZONE

If you give an LLM a tool to delete_database() or send_email(), it might use it unexpectedly.

Solution: Always pause execution and ask the user for confirmation before running sensitive tools.

What is Prompt injection?

https://gandalf.lakera.ai/intro

Handling Errors

What if the API fails? Or the LLM hallucinates an invalid city name?

Strategy: Feed the error message back to the LLM!

```
{ "role": "tool", "content": "Error: 'Atlantis' is not a valid city." }
```

The LLM will read this and say: *"I'm sorry, I couldn't find weather data for Atlantis."*

Quiz: Safety

Question 3:

Why should you validate arguments before execution?

A. To save tokens.

B. Because LLMs can hallucinate invalid or dangerous inputs.

C. To make the code faster.

Module 5: Hands-On Building the Agent

Let's write code.

The Goal: Weather Agent

”What is the weather in Tokyo ”

The Goal: Currency conversion Agent

”Get real time currency conversion”

Quick check:
What input will currency conversion API will take?

Day 3 Complete

Tomorrow: The Memory (RAG & Vector DBs).

Q & A