AGENTIC AI

WEEK 1

Week 1: Foundation & Environment Setup

1

VS Code & Virtual Environment

Our setup is powered by VS Code, enabling high productivity. We use a fast and simple virtual environment for project isolation.

2

Git & Cursor Installation

Key steps include installing Git for version control and Cursor for enhanced coding.

3

Dotenv Module for API Keys

Utilize the `dotenv` module and `load_dotenv` function to securely manage API keys from `.env` files.

4

OS Module for Environment Variables

The `os` module's `getenv()` function reads API keys, with `override=True` to update existing values.

Model Integration

We're can also use free open source API keys other than openai api_key that is gemini and groq, which means some functions and classes differ from standard OpenAI API usage.

Message Format

```
messages = [
    {"role": "user", "content": "What is 2+2?"}
]
```

Calling OpenAI

```
client = OpenAI(api_key = "",base_url="")
response = client.chat.completions.create(
 model="...", # model to be used
 messages=messages
)
print(response.choices[0].message.content)
```

Day 2: Agents & Agentic Architecture

An Agent is a system where the output from an LLM can decide what tasks are carried out in sequence.People tend to often use the term Agentic AI ,if any or all of the hallmarks are met:-

Multiple LLM Calls

Agents often involve several interactions with language models.

Tool Usage

LLMs can use external tools to accomplish tasks.

Interactive Environment

LLMs interact with each other in a coordinated environment.

Planner & Autonomy

A planner coordinates activities, and autonomy defines the essence of agentic AI.

Agentic Systems: Workflows vs. Agents

Workflows

Systems where LLMs and tools are orchestrated through predefined code paths.

- Prompt Chaining: Chaining LLM calls for responses.
- Routing: Directs input to specialized sub-tasks.
- Parallelization: Breaks down tasks for concurrent execution.

Agents

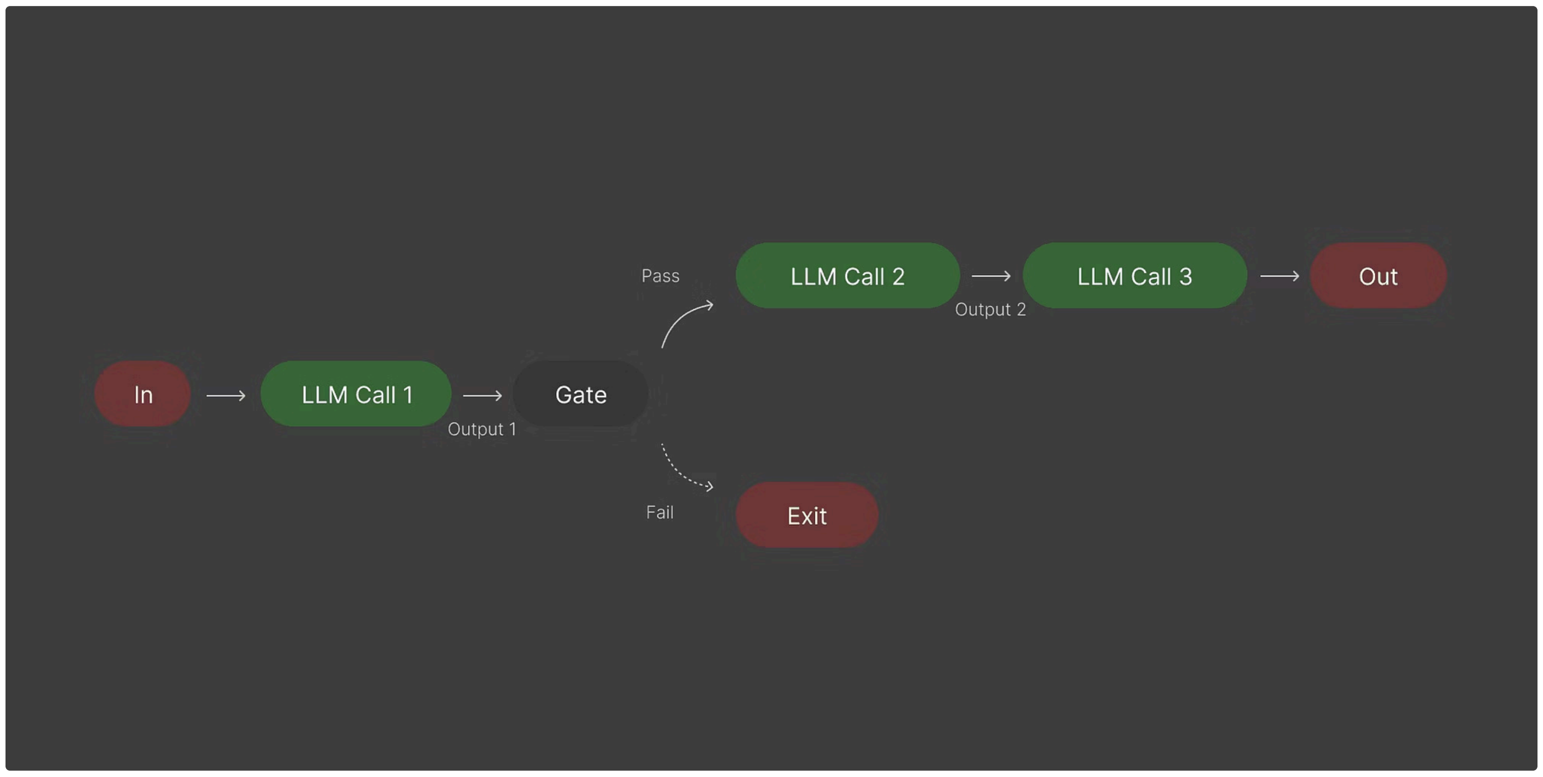
Systems where LLMs dynamically direct their own processes and tool usage, maintaining control over task accomplishment.

- Open-minded & Open-ended
- Feedback Loops
- No Fixed Path

Workflow Design Patterns

Prompt Chaining

Creating a sequence of LLM calls to refine responses.

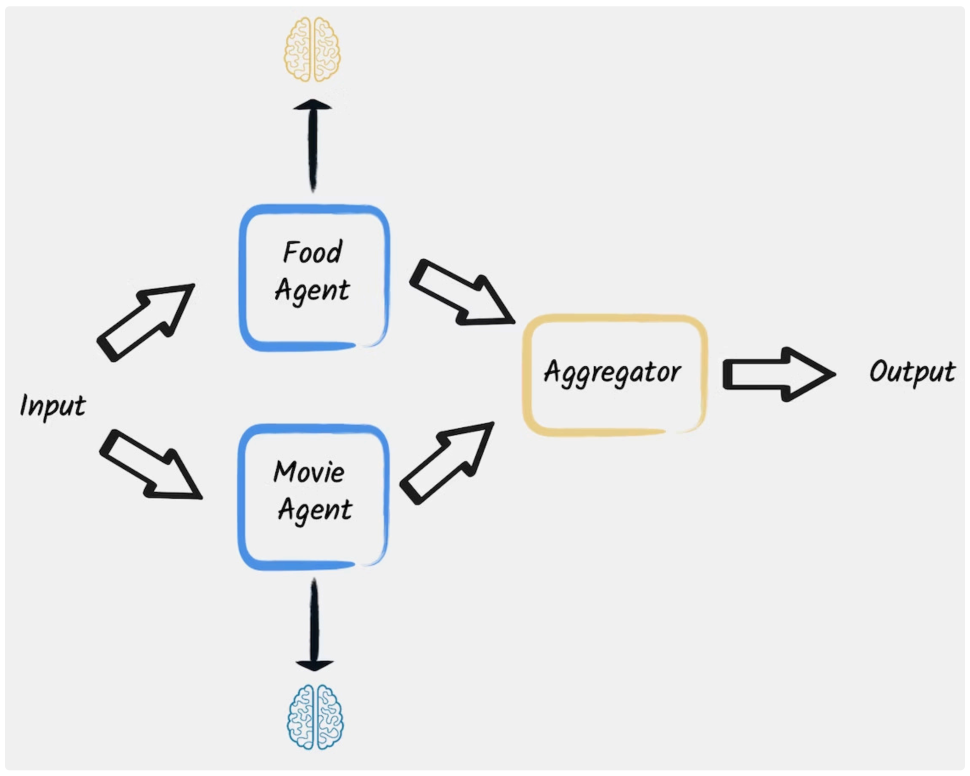


Routing

Directs input to specialized sub-tasks for operational clarity.

Parallelization

Breaking down tasks to run multiple sub-tasks concurrently.

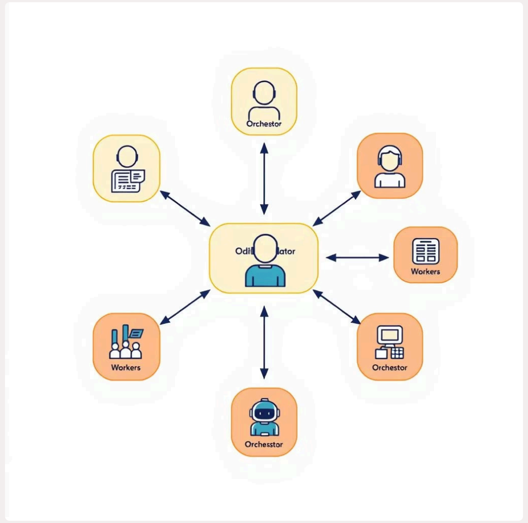


Made with GAMMA

Advanced Workflow Patterns

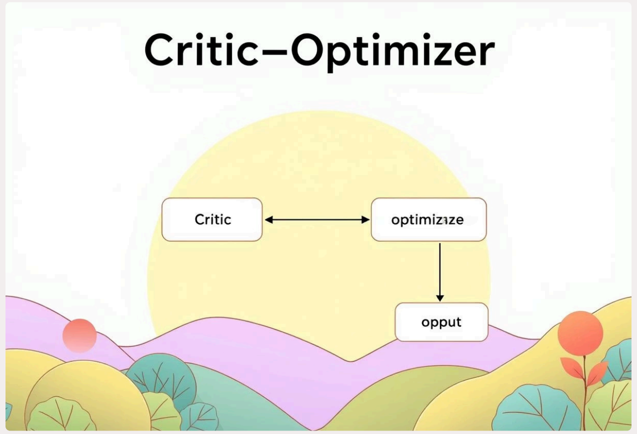
Orchestrator–Worker

Complex tasks are dynamically broken down and combined, with an LLM orchestrating the process.

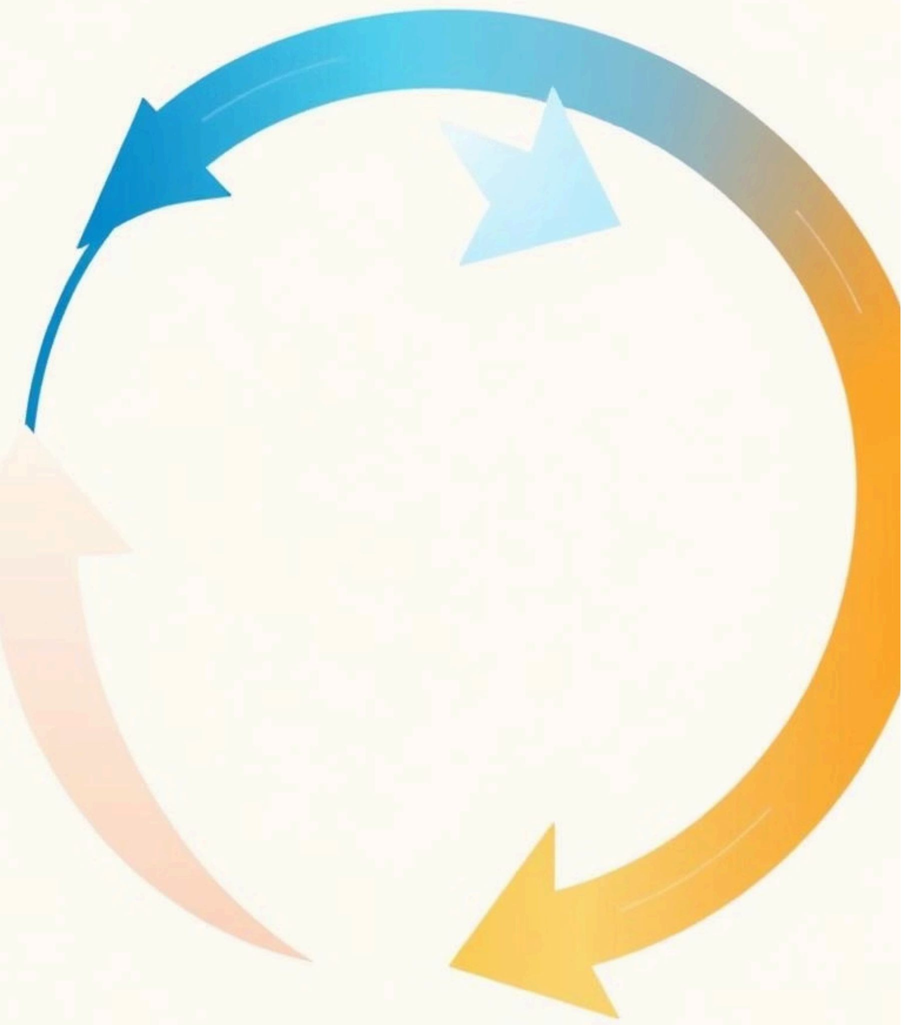


Critic–Optimizer

LLM output is validated by another LLM, creating a powerful feedback loop for refinement.



The Critic-Optimizer pattern is highly effective, as one LLM evaluates the work of another, accepting or rejecting it with reasons for rejection.



Agentic Feedback Loop

In agentic systems, a human initiates a request, which the LLM processes to generate an "Action."

- **Action Execution:** The action is executed, often involving tools like Stable Diffusion or Whisper.
- **Environment Feedback:** The environment provides feedback to the LLM based on the executed action.
- **Learning & Adaptation:** This feedback allows the LLM to learn and improve, adapting its behavior.
- **Goal Achievement:** The LLM continues this loop until the potential goal is achieved.

Drawbacks & Guardrails

Unpredictable Paths

Agents can take unforeseen routes to achieve goals.

Unpredictable Time

The duration of agentic processes can be unknown.

Monitoring & Visibility

Challenges in understanding what the agent is doing.

Guardrails

Ensures agents behave consistently and within intended boundaries, making sure the model performs as expected.

Day 3: Orchestrating LLMs

We utilize both paid APIs and open-source models (cloud and local) throughout this course.

Open-Source Models

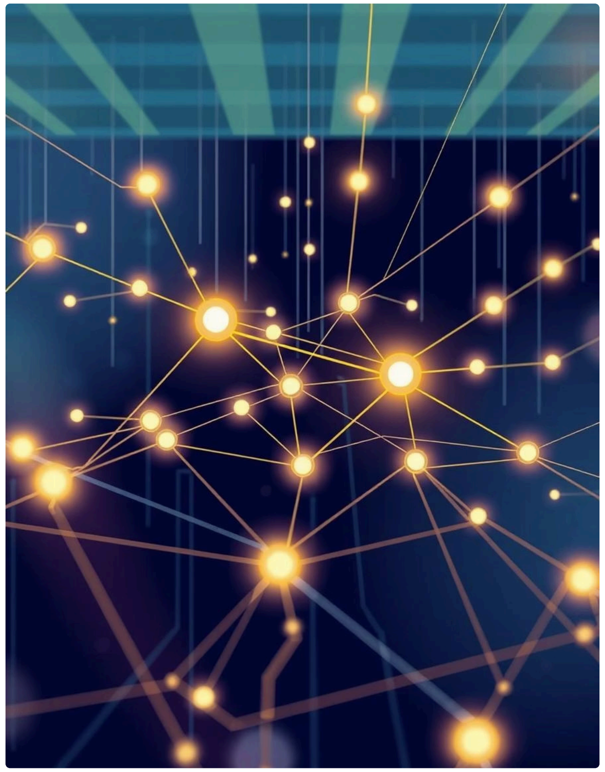
Architectures, code, and weights are publicly available for free use, modification, and redistribution. Examples: LLaMA (LLaMA2, LLaMA3), Mistral, Mixtral, Falcon, Bloom.

Proprietary APIs

Integration with services like OpenAI (GPT-4o-mini), Anthropic (Claude-3-sonnet), and Google (Gemini). We can switch between models by adjusting the base URL.

Navigating AI Agent Frameworks

This presentation explores various AI agent frameworks, from foundational protocols to advanced orchestration tools. We'll delve into how these frameworks enable the development of sophisticated, autonomous AI solutions.



Agenda

What We'll Cover

01

Agentic AI Frameworks

Understanding the core concepts and their role.

02

Model Context Protocol (MCP)

Anthropic's open-source standard for LLM interaction.

03

CrewAI

A Python-based framework for multi-agent orchestration.

04

LangGraph & AutoGen

Advanced frameworks for complex agent workflows.

05

Resources & Tools

Enhancing LLM autonomy and practice.

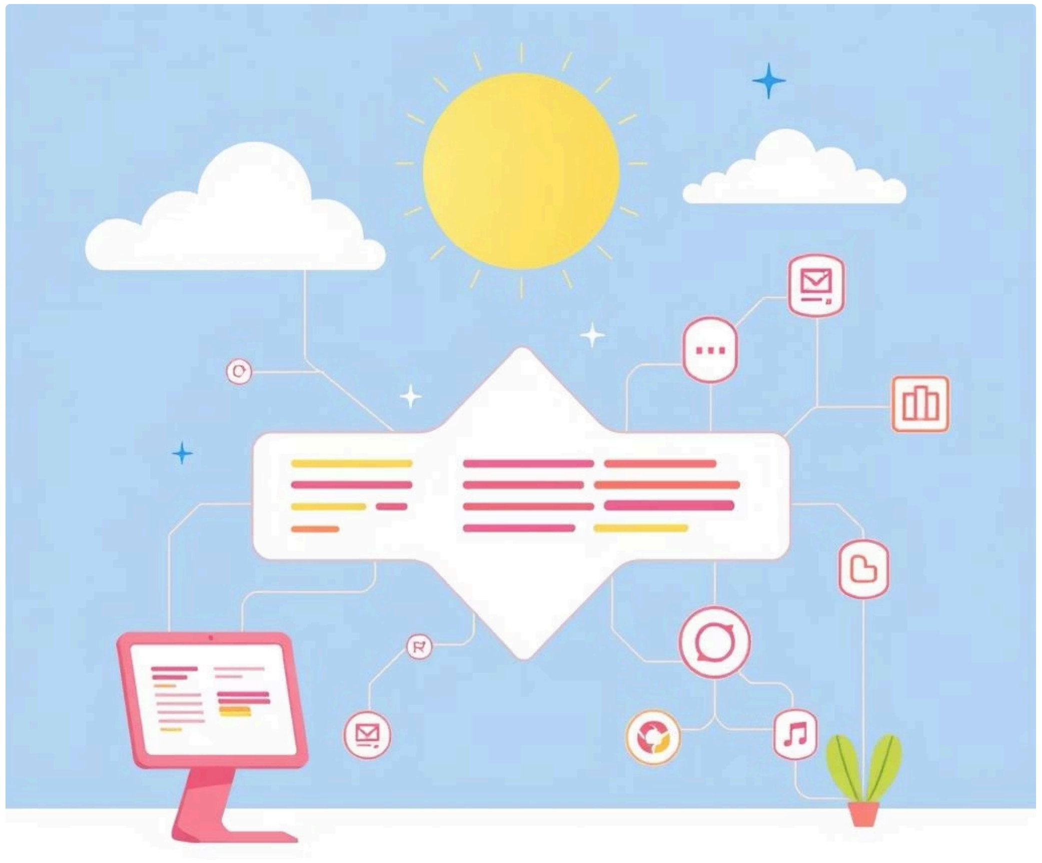
Agentic AI Frameworks

These frameworks provide an abstraction layer, simplifying the complexities of AI implementation. They offer an elegant structure for building agentic solutions, acting as the "glue" that connects various AI components.

Model Context Protocol (MCP)

Created by Anthropic, MCP is an open-source framework that standardizes how Large Language Models (LLMs) interact with external tools, systems, and data sources.

It's a non-framework, open-source standard designed to connect models with tools and data sources in a flexible, voice-free manner.



MCP is foundational, sitting at the bottom of the hierarchy as it doesn't carry any agentic AI framework itself, but rather connects using APIs.

CrewAI: Orchestrating Multi-Agent Systems

Python-Based

An open-source, Python-based framework.

Team Collaboration

Helps developers organize agents and orchestrate multi-agent teams.

Defined Roles

Agents have defined roles, goals, and workflows with guardrails.

CrewAI is built from scratch, providing a robust environment for developing and managing collaborative AI agents.

Advanced Orchestration: LangGraph & AutoGen

LangGraph

An open-source framework designed to construct, manage, and scale complex, stateful agent workflows using graph-based patterns. It operates as a low-level orchestration layer.

LangGraph models agents as nodes and edges connected by a shared state that routes the flow of execution, making it very powerful.

AutoGen

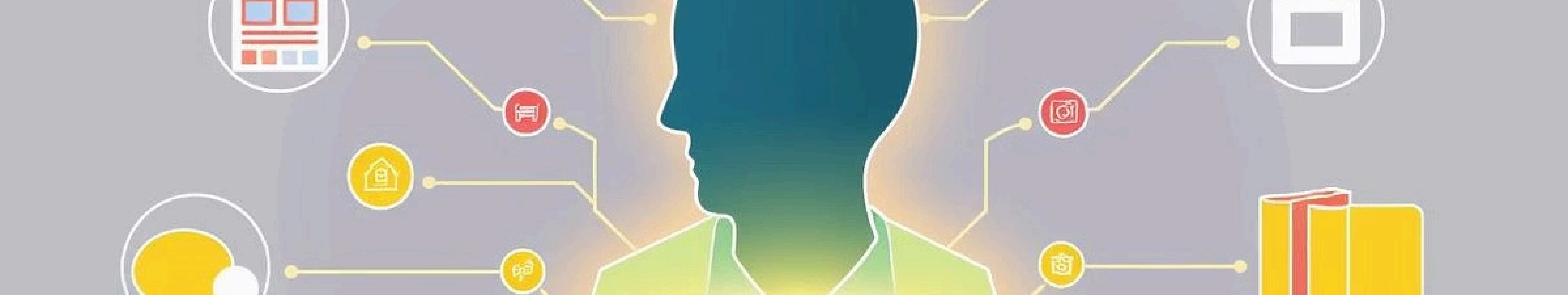
An open-source framework designed by Microsoft Research in collaboration with academic institutions like Penn State University and the University of Washington.

AutoGen makes modular, multi-agent systems and is designed for scenarios involving LLMs and human-in-the-loop oversight, with tool execution and root usage.

LangGraph vs. AutoGen

Complexity	Complex AI orchestration	Modular multi-agent systems
Pattern	Graph-based patterns	Human-in-the-loop oversight
Origin	MIT-developed	Microsoft Research collaboration

While both are powerful, LangGraph is heavier than AutoGen, focusing on low-level orchestration.



Resources for LLM Autonomy

Data Sharing

Provide LLMs with resources to improve expertise by sharing data and prompting with relevant information.

Relevant Content

Utilize techniques to get relevant content from resources, enabling LLMs to reject or answer questions effectively.

Tools in LLM Autonomy Practice

Prompt (LLM)

The initial input or query given to the LLM.

Response Generation

The LLM's process of creating an output based on the prompt.

Tools in Practice

External tools used by the LLM for specific tasks.

Feedback Loop

Continuous improvement based on responses and tool usage.

These elements form a cycle of autonomy, allowing LLMs to interact, learn, and refine their capabilities.

Key Takeaways & Next Steps

- AI agent frameworks provide essential structure for building intelligent systems.
- MCP offers a standardized way for LLMs to interact with external data and tools.
- CrewAI, LangGraph, and AutoGen enable complex multi-agent orchestration.
- Leverage resources and tools to enhance LLM autonomy and performance.