Lecture 11: CrewAI and Role-based Collaboration

🎯 Learning Objectives

By the end of this lecture, you should be able to:

- Understand the purpose of multi-agent collaboration using CrewAI.
- Define roles, tasks, and communication protocols between agents.
- Implement a basic crew of role-based agents that collaborate.
- Identify when and why to use multi-agent systems over single-agent loops.

🧩 Key Concepts

What is CrewAI?

- A Python-based framework that enables **multi-agent collaboration**.
- Agents are assigned **roles**, **tools**, and **goals**.
- Tasks are delegated, coordinated, and executed in a structured order.

Why Use Multi-Agent Systems?

- Specialization: Each agent can focus on a narrow expertise.
- Scalability: Parallelize subtasks across roles.
- Modularity: Easier to update, test, and optimize components individually.

Common Roles in Multi-Agent Systems

- **Researcher**: Gathers data from tools or memory.
- **Planner**: Breaks high-level goals into subtasks.
- **Writer**: Synthesizes outputs into structured text.
- **Validator**: Checks consistency or fact accuracy.

🛠️ Required Tools/Libraries

- Python

- CrewAI

  ```
  pip install crewai openai langchain
  ```

- OpenAI API key or other LLM provider

🔬 Hands-on Exercise: Multi-Agent Blog Writer

Goal: Create a crew of agents that research, draft, and validate a blog article.

Step 1: Define agents with roles

```
from crewai import Agent

researcher = Agent(name="Researcher", role="Collects relevant facts and links.")
writer = Agent(name="Writer", role="Drafts structured blog posts.")
reviewer = Agent(name="Reviewer", role="Ensures factual consistency and clarity.")
```

Step 2: Assign tasks to agents

```
from crewai import Task

research_task = Task(agent=researcher, goal="Gather facts about LangChain and its use cases.")
write_task = Task(agent=writer, goal="Write a blog post based on the research.")
review_task = Task(agent=reviewer, goal="Fact-check and polish the draft.")
```

Step 3: Create and run the crew

```
from crewai import Crew

crew = Crew(tasks=[research_task, write_task, review_task])
crew.run()
```

Step 4: Review output

- Observe how agents pass information.
- Check role effectiveness and task completion quality.

Bonus:

- Add custom tools to specific roles (e.g., web search for researcher).
- Include memory so agents recall prior collaborations.
- Visualize agent flow using a task dependency graph.
