Lecture 08: Multi-step Autonomy and Looping

🎯 Learning Objectives

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

- Understand how agents maintain autonomy across multiple steps.
- Implement the action-observation loop used in agent execution.
- Handle intermediate results and dynamic decision-making.
- Build a simple looped agent to solve compound tasks.

🧩 Key Concepts

Agent Loop Architecture

- The agent performs a **thought–action–observation** cycle:

    1. **Thought**: Decide what needs to be done.
    2. **Action**: Call a tool or perform a step.
    3. **Observation**: Interpret the result of the action.
    4. **Repeat** until the goal is achieved or a stopping condition is met.

- This creates an iterative loop that allows agents to adapt as new data arrives.

Benefits of Looped Agents

- Can handle tasks with:
    - Uncertainty
    - Changing goals
    - External dependencies
- Capable of retrying or correcting failures

Example Use Cases

- Research assistant searching for multiple facts
- Code generation agent refining its output
- Task automation agent solving a multi-part goal

🛠️ Required Tools/Libraries

- Python
- OpenAI API (or similar LLM)
- Tool functions (calculator, search, etc.)
- LangChain (optional for structured agent loops)

🔬 Hands-on Exercise: Multi-step Research Agent

Goal: Build an agent that answers a research-style question by looping through multiple steps.

Step 1: Define tools

```
def search(query):
    # Simulate a search tool (replace with real API later)
    database = {
        "Who is Elon Musk?": "Entrepreneur and CEO of Tesla and SpaceX.",
        "What is SpaceX?": "A private aerospace company founded by Elon Musk.",
    }
    return database.get(query, "No results found.")
```

Step 2: Simulate a multi-step loop

```
question = "What company was founded by the person who leads Tesla?"

# Step 1 - Thought: Need to find out who leads Tesla
result_1 = search("Who is Elon Musk?")

# Step 2 - Thought: Now find out what company he founded
result_2 = search("What is SpaceX?")

# Final Answer
print("SpaceX is the company founded by Elon Musk.")
```

Step 3: Add LLM to generate each step

```
- Let the LLM generate intermediate questions.
- Use results to guide the next step.
- Track all steps in a log or memory buffer.
```

Bonus:

- Add stopping criteria (e.g., "Final Answer" in the response).
- Limit the number of steps to avoid infinite loops.
- Build a reusable agent loop function to handle any input goal.
