Lecture 07: Tool Use and External APIs

🎯 Learning Objectives

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

- Understand how and why agents use tools.
- Implement basic tool-calling logic in an LLM-based agent.
- Integrate external APIs (e.g., calculator, web search, weather).
- Build a multi-function agent with reasoning and action steps.

🧩 Key Concepts

Why Tools Matter

- LLMs have limits: they can't browse the web, do math reliably, or access live data.
- Tools extend LLM capabilities to:
  - Perform calculations.
  - Retrieve real-time or factual data.
  - Interact with databases, APIs, or file systems.

Agent + Tool Architecture

- The LLM decides **when** and **which** tool to use.
- A controller or wrapper calls the tool and returns the result.
- The LLM observes the output and continues reasoning.

Common Tool Types

- **Math tools**: Calculators, math engines (SymPy, Wolfram)
- **Search tools**: Google, DuckDuckGo, SerpAPI
- **Knowledge tools**: Wikipedia, APIs
- **Custom tools**: Local files, internal services, business APIs

---

🛠️ Required Tools/Libraries

- OpenAI API (or similar LLM)
- Requests / HTTP client library
- Python
- (Optional) LangChain or custom agent shell

🧪 Hands-on Exercise: Calculator Tool Agent

Goal: Build a simple agent that uses a calculator tool when prompted.

Step 1: Define a tool function

```
def calculator(query):
    try:
        return str(eval(query))
    except:
        return "Error"
```

Step 2: Prompt agent to use tools

```
Prompt = """
You are a smart agent. If a question involves math, use the calculator tool.

Question: What is 12 * 8?
Thought: This is a math question. I will use the calculator.
Action: calculator("12 * 8")
Observation: 96
Final Answer: 96
"""

# Simulate the flow
# (LLM generates Thought and Action; you execute Action and return Observation)
```

Step 3: Integrate with an LLM

```
- Accept a question from the user.
- LLM generates Thought + Action.
- Your agent parses and executes the action.
- Feed the result back and let LLM finalize the answer.
```

Bonus:

- Add a search tool using SerpAPI or DuckDuckGo.
- Let the agent choose between "calculator" and "search" based on question type.
- Log tool calls for traceability and debugging.
