Phase 1 — Foundations of LLM Agents

1. Research Paper Summaries

```
■ Paper 1 — Generative to Agentic AI: Survey, Conceptualization, and Challenges (2025)
Goal: Show the evolution from passive text generation to autonomous, goal-driven AI systems.

Key Points:
- Generative AI: Reactive, produces outputs on demand, no persistent goal.
- Agentic AI: Autonomous, has goals, can perceive, reason, act, and adapt.
- Core Pillars:
  1. Autonomy — ability to initiate actions without direct prompts.
  2. Goal-Directedness — acts toward explicit or implicit objectives.
  3. Tool-Use — interacts with APIs, databases, devices.
  4. Memory — short-term (context window) + long-term (vector DB).
  5. Adaptation — learns/improves from past interactions.
- Challenges: Safety, reliability, interpretability, evaluation.

■ Paper 2 — Large Language Model Agent: A Survey on Methodology (2025)
Goal: Provide a taxonomy of how to design an LLM-powered agent.

Architecture Layers:
1. Planner — Task decomposition (zero-shot, few-shot, or learned). Examples: Chain-of-Thought, Tree-of-T
2. Executor — Executes actions in environment (APIs, simulators, OS commands).
3. Memory — Episodic: raw interaction history. Semantic: distilled facts for retrieval.
4. Tool Interface — Function calls, API bindings, external services.
5. Evaluator — Self-evaluation or human feedback for correction.

Design Variations:
- Single-Agent vs. Multi-Agent collaboration.
- Closed-loop (with feedback) vs. Open-loop (no feedback until end).
- Symbolic Planning (PDDL) vs. Language Planning (natural language reasoning).
```

2. Core LLM Agent Architecture

```
■■■■■■■■■■■■■■■■■■■■■■■■■■
■   User / Environment  ■
■■■■■■■■■■■■■■■■■■■■■■■■■■
          ■
    ■■■■■■■■▼■■■■■■■■■
    ■   Planner    ■
    ■ (LLM-based)  ■
    ■■■■■■■■■■■■■■■■■■
          ■ Plan / Thought
    ■■■■■■■■▼■■■■■■■■■
    ■   Executor   ■
    ■(Tool Calls)  ■
    ■■■■■■■■■■■■■■■■■■
          ■ Observations
    ■■■■■■■■▼■■■■■■■■■
    ■    Memory    ■
    ■(Episodic +   ■
    ■ Semantic DB) ■
    ■■■■■■■■■■■■■■■■■■
          ■
    ■■■■■■■■▼■■■■■■■■■
    ■  Evaluator   ■
    ■■■■■■■■■■■■■■■■■■
```

3. Minimal ReAct-Style Agent in Python

```
python
import openai

openai.api_key = "YOUR_API_KEY"
```

```python
def react_agent(question, tools):
    """
    Minimal ReAct agent loop:
    - Tools: dict with {tool_name: callable}
    - Each loop: Thought → Action → Observation
    """
    prompt = f"You are a helpful AI agent. Answer the question by thinking step-by-step, using tools if
    prompt += f"Available tools: {', '.join(tools.keys())}\n"
    prompt += "Format: Thought: ...\nAction: tool_name[input]\nObservation: ...\nFinal Answer: ...\n"
    prompt += f"Question: {question}\n"

    conversation = prompt
    for step in range(5):  # limit steps
        response = openai.ChatCompletion.create(
            model="gpt-4o",
            messages=[{"role": "user", "content": conversation}],
            temperature=0
        )
        content = response['choices'][0]['message']['content']
        print(content)
        conversation += content + "\n"

        # Tool execution
        if "Action:" in content:
            action_line = [l for l in content.split("\n") if l.startswith("Action:")][0]
            tool_name, tool_input = action_line.replace("Action:", "").strip().split("[", 1)
            tool_input = tool_input.strip("]")
            if tool_name in tools:
                observation = tools[tool_name](tool_input)
                conversation += f"Observation: {observation}\n"
            else:
                conversation += "Observation: Tool not found.\n"

        if "Final Answer:" in content:
            break

# Example tools
def search_tool(query):
    return f"(Simulated search result for '{query}')"

def calculator_tool(expr):
    try:
        return eval(expr)
    except:
        return "Error in calculation"

tools = {
    "search": search_tool,
    "calculator": calculator_tool
}

react_agent("What is the population of France plus 2?", tools)
```

■ Phase 1 Checkpoint:
- [ ] Read and understand the two survey papers.
- [ ] Run the minimal ReAct agent.
- [ ] Add one more tool (e.g., currency converter).