SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 03 · Agentic AI Engineering

11

Agentic AI Fundamentals

ReAct loop, tool calling, memory types & cognitive architectures

Module 11 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> ReAct loop, tool calling, memory types & cognitive architectures. This guide gives you the core theory, the tools that matter, a hands-on build, working code, and the pitfalls to avoid — everything you need to learn the topic to a professional standard and apply it in real projects.

1 hour	4 + 4	3	11/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 03

02 Learning Objectives

- Explain what makes a system 'agentic' vs a single LLM call.
- Implement the ReAct (reason + act) loop from scratch.
- Wire tool calling so a model can take real actions.
- Distinguish working, episodic, and long-term memory.

03 Core Concepts

What is an agent

An agent uses an LLM as a reasoning engine in a loop: it observes, decides on an action (often calling a tool), executes, observes the result, and repeats until the goal is met. The defining shift is autonomy over multiple steps rather than a single response.

The ReAct loop

ReAct interleaves Reasoning ('what should I do next?') and Acting (tool calls). Each cycle: Thought -> Action -> Observation. Explicit reasoning traces make the agent more reliable and far easier to debug.

Tool calling

Tools are typed functions the model can invoke (search, calculator, database, API). You expose a schema; the model returns a structured call; your code executes it and feeds the result back. Tools are how agents affect the real world.

Memory types

Working memory = the current context window. Episodic memory = a log of past interactions. Long-term/semantic memory = durable knowledge in a store (often vector-backed). Cognitive architectures combine these so agents remember across sessions.

04 Visual Diagram

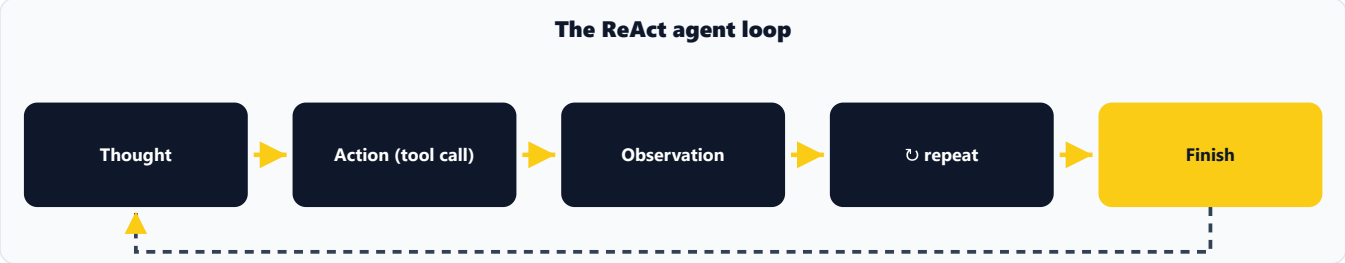

Figure 11 — ReAct interleaves reasoning and acting. The loop repeats until the goal is met or a step limit is hit.

05 Key Frameworks & Tools

Tool-calling APIs (Claude/OpenAI) | LangGraph | Plain Python (from scratch)

06 Hands-On Build

🛠 ReAct Reason-Loop from scratch in Python

1. Define 2-3 tools with JSON schemas (e.g. search, calculator).
2. Implement the Thought -> Action -> Observation loop manually.
3. Add a stopping condition and a max-iterations guard.
4. Log the full reasoning trace for debugging.

07 Code Walkthrough

Core of a hand-rolled ReAct loop — PYTHON

```
def react(goal, tools, llm, max_steps=6):
    history = [f"Goal: {goal}"]
    for _ in range(max_steps):
        thought = llm("\n".join(history) + "\nThink, then choose an Action(tool, args) or FINISH.")
        history.append(thought)
        if "FINISH" in thought:
            return thought
        tool, args = parse_action(thought)
        observation = tools[tool](**args)
        history.append(f"Observation: {observation}")
    return "Stopped: step limit reached"
```

08 🧪 Lab Assistant

Lab 11 · Build a ReAct loop from scratch

Prerequisites: An LLM API key · 2-3 Python tool functions

1. **Define tools with clear JSON schemas**
   > Expected: The model can name a tool and its args
2. **Run the Thought→Action→Observation loop**
   > Expected: The agent calls tools and reaches an answer
3. **Add a max-steps guard and log the trace**
   > Expected: No infinite loops; full trace printed

Troubleshooting

- Loops forever → enforce max_steps and a FINISH token.
- Wrong tool → sharpen tool descriptions.

09 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- No max-iteration guard — agents can loop forever and burn tokens.
- Vague tool descriptions, so the model misuses them.
- Discarding reasoning traces, making failures impossible to debug.

✓ KEY TAKEAWAYS

- Agentic = LLM reasoning in a multi-step observe-act loop.
- ReAct's explicit Thought/Action/Observation aids reliability and debugging.
- Tools give agents real-world effect; memory gives them continuity.

📺 **Companion video:** Watch the module 11 lesson in your Student Dashboard, then use this guide to revise, copy the code, and complete the build. Download both for offline study.

10 Further Resources

'ReAct' paper (Yao et al.) • Anthropic tool-use docs • LangGraph quickstart