M10 — Agents & Workflows Lab

Lab — Generative AI Boot Camp for Developers — Exported 2026-09-02

Time: 45 min | **Day 3 PM**

2026-09-02

Learning Objectives

Build a simple agent workflow — the capstone pattern of the workshop. Understand when agents are the right solution and when they add unnecessary complexity.

2026-09-02

Concept Review: AI Agents

What Makes an Agent?

An agent is an AI system that can **reason, plan, use tools, and take multi-step actions.** The key difference from a chatbot: agents don't just respond — they act.

```
Agent = LLM (brain) + Tools (actions) + Memory (context) + Planning (task decomposition)
```

The Agent Loop

```
Goal → Plan steps → Execute Step 1 → Evaluate result →
Adjust plan → Execute Step 2 → ... → Deliver result
```

When to Use Agents vs. Simple Prompts

Use an Agent When	Use a Simple Prompt When
Multi-step task with branching logic	Single, well-defined task
Task requires external tools (search, APIs, code execution)	Task is pure text generation
The path isn't known in advance	You know exactly what you want
You need the AI to *decide* next steps	A single prompt produces the answer

2026-09-02

Agent Frameworks at a Glance

Framework	Best For	Complexity
Custom Python loop	Learning, maximum control	Low
LangChain/LangGraph	Production workflows, broad ecosystem	Medium-High
CrewAI	Multi-agent collaboration	Medium
AutoGen (Microsoft)	Multi-agent conversations	Medium

> **Today's lab uses a custom Python loop** — understand the fundamentals before reaching for a framework.

Prerequisites

- LLM API access configured
- Python environment ready
- Agent scaffold from `workshop/m10/scaffold.py`

2026-09-02

Exercise: Research Assistant Agent

Step 1 — Understand the Scaffold (5 min)

Copy `workshop/m10/scaffold.py` to your working directory. Read through it. The scaffold provides the agent loop structure — you implement the AI interaction logic.

Step 2 — Implement Core Logic (20 min)

The agent should: take a research question → search/analyze → produce a summary.

Must include at least **2 sequential LLM calls** with processing between them:

```
# Example flow:
# 1. LLM Call: "Break this research question into sub-questions: {question}"
# 2. Process: extract sub-questions
# 3. LLM Call: "Answer these sub-questions: {sub_questions}"
# 4. Process: combine answers into a summary
```

Use **OpenCode** to help implement the logic:

```
opencode run "Read the agent scaffold. Implement the core agent logic so it takes
a research question, uses 2 sequential LLM calls (first to decompose, second to
answer each sub-question), and produces a final summary."
```

2026-09-02

Step 3 — Add a Tool (10 min)

Add one tool to your agent: a simulated web search or file reader. The agent should *decide* when to use the tool.

Step 4 — Test & Reflect (10 min)

Test with different research questions. Reflection: - Where did the agent succeed? Struggle? - Would a single prompt have worked just as well? - What would you need to change for a production agent?

Deliverable

Working agent with 2+ sequential LLM calls + 1 tool integration.

2026-09-02

Troubleshooting

Symptom	Fix
Agent loops forever	Add `max_iterations` limit to the loop
Agent makes bad decisions	Improve the planning prompt — be more specific about the goal
Tool integration fails	Check tool function signature; ensure it returns parseable results
Scaffold not found	Copy from `workshop/m10/scaffold.py`

2026-09-02

Hints

- **Start simple:** 2 LLM calls + 1 tool. More complexity = more things to debug.
- **The agent's planning prompt is the most important part** — it determines every subsequent decision.
- **Ask yourself after the lab:** Would a single well-crafted prompt have solved this problem? If yes, you probably didn't need an agent. That's a valuable lesson too.

2026-09-02