SAMIR GENAI ACADEMY

PROFESSIONAL STUDY GUIDE SERIES

Phase 04 · Advanced Agentic Patterns

18

Code & Browser Agents

Build agents that write, run code & control browsers

Module 18 of 25 · **1 hour**

Instructor: **Samir Kumar** · Edition 2026

STUDY GUIDE

01 Overview

> Build agents that write, run code & control browsers. 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	4	18/25
LESSON LENGTH	CONCEPTS + BUILD STEPS	FRAMEWORKS & TOOLS	PHASE 04

02 Learning Objectives

- Run model-generated code safely in a sandbox.
- Build a code-interpreter agent loop.
- Drive a browser to complete web tasks.
- Handle errors and self-correction.

03 Core Concepts

Why sandboxing

Executing model-written code unlocks data analysis, automation, and tool building — but running it on your machine is dangerous. Sandboxes (E2B, Docker, microVMs) isolate execution so a bad command can't touch your system.

Code-interpreter loop

The agent writes code, runs it in the sandbox, reads stdout/stderr, and iterates until the task succeeds. This write-run-observe-fix loop lets agents solve problems no single completion could, and it self-corrects from real error messages.

Browser automation

With Playwright/Selenium (or vision-based control), agents navigate sites, fill forms, and scrape data. Combine a planner with DOM/visual observation so the agent reacts to the actual page, not an assumed one.

Robustness

Add retries, timeouts, and explicit success checks. Capture errors back into the loop so the agent fixes its own mistakes rather than failing silently.

04 Visual Diagram

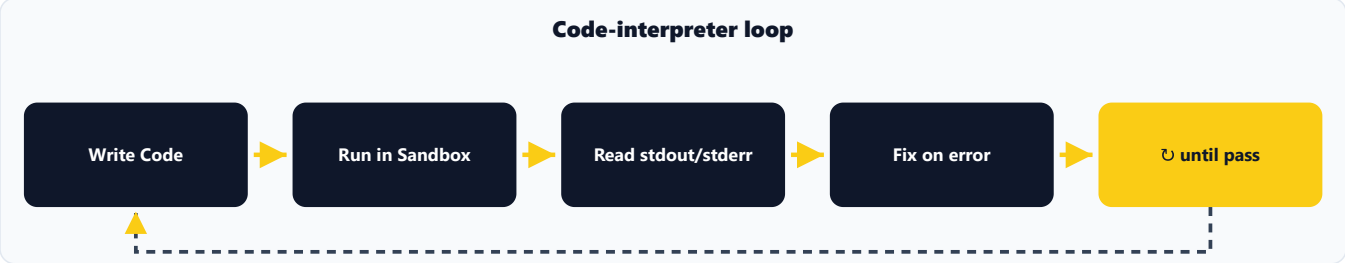

Figure 18 — The agent writes code, runs it safely in a sandbox, reads the result, and self-corrects from real errors.

05 Key Frameworks & Tools

E2B | Docker | Playwright | Code-interpreter tooling

06 Hands-On Build

🛠 E2B Sandboxed Code Interpreter Agent

1. Provision an E2B (or Docker) sandbox.
2. Let the agent write code, execute it, and read results.
3. Feed errors back so it self-corrects until success.
4. Add a browser tool for a web-data subtask.

07 Code Walkthrough

Running agent code in an E2B sandbox — PYTHON

```python
from e2b_code_interpreter import Sandbox

with Sandbox() as sb:
    code = "import numpy as np; print(np.mean([2, 4, 6]))"
    exec_result = sb.run_code(code)
    print(exec_result.logs.stdout)  # ['4.0']
    # feed stderr back to the agent if execution failed
```

08 🧪 Lab Assistant

Lab 18 · Sandboxed code agent

Prerequisites:pip install e2b-code-interpreter · An E2B API key

1. **Open a sandbox and run model-written code**
   > Expected: stdout captured, host untouched
2. **Feed stderr back to the agent on failure**
   > Expected: The agent fixes and retries
3. **Add a max-iteration cap**
   > Expected: No runaway loops

Troubleshooting

- NEVER run model code unsandboxed on your machine.
- Loops forever → cap iterations and require a success check.

09 Pitfalls & Key Takeaways

⚠ COMMON PITFALLS

- Running model-generated code outside a sandbox.
- No timeout/iteration cap on the self-correction loop.
- Brittle selectors that break on minor page changes.

✓ KEY TAKEAWAYS

- Always sandbox model-generated code execution.
- The write-run-observe-fix loop enables real self-correction.
- Ground browser agents in the actual page state, with retries.

📺 **Companion video:** Watch the module 18 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

E2B docs • Playwright docs • Open-interpreter project