Faysal Aziz

Practical Harness Engineering with Python

Building Controlled and Verifiable AI-Agent Workflows - 2 days

FA-0364 · 2 days

Software Development · Data & Analytics

Introduction

Turn harness concepts into a working Python-based agent system.

AI agents become useful engineering systems only when they can perform controlled actions and demonstrate that those actions were completed correctly. This requires more than sending a prompt to a language model. The surrounding software must manage task instructions, repository context, tool execution, permissions, state, validation and failure handling.

This two-day course moves from Harness Engineering concepts into practical Python implementation. Participants will construct a small coding-agent harness capable of accepting a task, assembling relevant context, exposing approved tools, recording activity and running automated verification. The emphasis is not on developing a sophisticated autonomous platform, but on understanding and implementing the essential control mechanisms from which larger harnesses are built.

Python is particularly suitable for this work because its standard library supports file management, structured data, logging, process execution and testing without requiring a large framework. Participants will use these capabilities to build a transparent harness whose behaviour can be inspected and modified. Python's process-management facilities also allow external development commands to be executed while capturing output, return codes and failures.

Current industry work treats the harness and model as a combined system. OpenAI emphasises repository structure, agent legibility, feedback loops and enforceable constraints, while Anthropic highlights persistent progress, verification and effective context transfer for longer-running work.

The course remains intentionally limited to a single-agent Python harness. Multi-agent orchestration, distributed execution, advanced sandboxing and production-scale agent platforms are reserved for the advanced phase. The instructor has more than 30 years of industry experience and will use practical,

industry-demanded approaches rather than treating the subject as a purely academic exercise.

Learning outcomes

By the end of this course, participants should be able to:

- Structure a small Harness Engineering project in Python
- Implement task, context, tool and state components
- Create controlled file and command-execution tools
- Apply tool allowlists, timeouts and workspace boundaries
- Record agent actions using structured logging
- Build a basic agent execution and correction loop
- Connect automated checks to completion criteria
- Test and assess a Python-based harness

Prerequisites

- Completion of Harness Engineering Foundations or equivalent knowledge
- Working knowledge of Python functions, classes and modules
- Basic understanding of JSON and file operations
- Familiarity with Git repositories
- Basic command-line and software-testing experience
- Access to Python 3.11 or later
- Access to an approved language-model API or local model endpoint

Training outline

1. Python Harness Project Setup

1. Project structure
2. Virtual environments
3. Configuration management
4. Environment variables
5. Dependency management
6. Test-project preparation

2. Harness Data Models

1. Task specifications
2. Acceptance criteria
3. Tool requests and results
4. Agent messages
5. Execution status
6. Structured JSON state

3. Context Management

1. Repository instruction loading
2. File discovery
3. Context filtering
4. Context-size controls
5. Task-specific context assembly
6. Sensitive-file exclusions

4. Python Tool Development

1. Tool interfaces
2. File-reading tools
3. Controlled file-writing tools
4. Repository-search tools
5. Command-execution tools
6. Structured tool responses
7. Tool error handling

5. Safety and Execution Controls

1. Tool allowlists
2. Command validation
3. Workspace restrictions
4. Path validation
5. Execution timeouts
6. Output-size limits
7. Approval checkpoints

6. Agent Execution Loop

1. Task intake
2. Prompt and context assembly
3. Model invocation
4. Tool-call processing
5. Result return
6. Iteration controls
7. Completion detection

7. State and Observability

1. Task-state persistence
2. Progress tracking
3. Structured logging
4. Tool-call history
5. Error records
6. Completion summaries

8. Verification Pipeline

1. Formatting checks
2. Linting
3. Unit tests
4. Command return codes
5. Acceptance-criteria validation
6. Verification evidence
7. Failure-driven correction

9. Harness Testing

1. Tool unit tests
2. Mock model responses
3. Invalid command tests
4. Path-escape tests
5. Timeout tests
6. Incomplete-task tests

7. Harness integration testing

10. Practical Harness Assembly

1. Component integration
2. Controlled coding task execution
3. Validation and correction cycle
4. Activity-log inspection
5. Failure diagnosis
6. Final harness assessment

11. Industry Integration Direction

1. Git workflow integration
2. Continuous integration entry points
3. Model Context Protocol overview
4. MCP tools and resources
5. Human review boundaries
6. Preparation for advanced Harness Engineering

Disclaimer

This outline is provided as a professional instructional guideline for a two-day practical programme. The trainer reserves the right to amend, reorder, reduce, expand or substitute topics at their professional discretion to reflect participant ability, available infrastructure, organisational requirements, technical developments or instructional priorities. Such adjustments may be made without prior notice where reasonably necessary to maintain the relevance, safety and practical value of the training.

© 2026 Faysal Aziz · Kuala Lumpur