Faysal Aziz

# AI-Powered SOC Engineering

## Python, VS Code and controlled agent workflows

FA-0577 · 2 days

Cybersecurity · Agentic & Generative AI · Software Development

Prototype AI-assisted SOC workflows for log triage, enrichment and documentation using Python, with validation, traceability and human approval.

---

## Introduction

This two-day workshop is designed for SOC engineers and analysts who use scripts to support investigations and operational workflows. It combines Python and VS Code with API-based language models to explore log parsing, enrichment, classification, investigation summaries and runbook preparation.

Participants build selected, bounded workflows using synthetic or approved de-identified security data. They compare hosted OpenAI API usage with self-managed Qwen deployment options and examine reusable skills, lifecycle hooks and tool orchestration. Exercises are prototypes for review, not production response automation; live deployment requires additional integration, security assessment and operational approval.

## Learning outcomes

The course teaches participants to:

- Organise a Python SOC automation project in VS Code and process log, JSON and CSV inputs.
- Make controlled model API calls and validate structured outputs against the original evidence.
- Prototype a narrow log-triage or investigation-summary workflow with reusable functions and explicit tool boundaries.
- Describe agent skills, lifecycle hooks and human approval points for repeatable operations.
- Compare hosted and self-managed model options, including hardware, data handling and compatibility constraints.
- Review generated scripts, summaries and operational recommendations before their use.

# Prerequisites

- Familiarity with SOC operations, alert handling and log investigation.
- Basic Python or scripting experience and comfort with a command line.
- Basic knowledge of JSON, APIs and data structures.
- Experience with SIEM or security automation workflows is helpful; prior agent-development experience is not required.

# Training outline

## 1. Day 1 — Python and AI-Assisted SOC Tasks

## Module 1 — AI in Security Operations

- Identify suitable triage, enrichment, summarisation and documentation tasks.
- Distinguish deterministic scripts, model-assisted workflows and tool-using agents.
- Review hallucination, grounding, privacy and prompt-injection risks; define human responsibility.
- Compare hosted and on-premises model approaches.

## Module 2 — VS Code and Python Foundations

- Workspaces, terminals, extensions, virtual environments and debugging.
- Parse text, regular expressions, JSON, CSV and event records.
- Create reusable functions for transformation, enrichment and external API requests.
- Use exceptions, logging and command-line utilities; manage secrets separately from code.
- Organise and package small automation utilities for team review.

## Module 3 — Model API Calls and Prompt Design

- Build a Python wrapper for model requests and responses.
- Specify task instructions, context and expected output; keep untrusted logs separate from instructions.
- Use supported structured-output schemas and handle refusals, incomplete responses and application-level validation.
- Create prompts for alert explanation, evidence extraction, parser support and investigation summaries.
- Evaluate grounding, ambiguity and cost/latency trade-offs; refine reusable prompt templates.

## Module 4 — SOC Workflow Exercise

- Build a selected log-triage or investigation-summary prototype using prepared data.
- Explore indicator enrichment, timeline consolidation, query/filter assistance and structured analyst notes.
- Review generated code, remediation suggestions, runbook drafts and handover documentation.
- Check factual references against original records; treat model classifications as analyst support, not final incident determinations.

# 2. Day 2 — Agents, Operational Controls and Local Models

## Module 5 — Bounded Agent Workflows

- Compare single-purpose assistants with multi-step agents and state-aware pipelines.
- Decompose a narrow task and define roles, tool inputs, outputs and context transfer.
- Use retries, fallbacks, uncertainty handling and escalation paths.
- Separate observation from action; require approval for sensitive tool operations.
- Keep orchestration inspectable and avoid unnecessary dependencies or open-ended loops.

## Module 6 — Reusable Skills and Lifecycle Hooks

- Define task-specific skill purpose, boundaries, inputs and expected outputs.
- Organise reusable parsing, enrichment, documentation or detection-review instructions.
- Separate reusable skill logic from standalone scripts and orchestration.
- Use supported hooks or explicit wrapper functions to validate inputs, observe execution and review outputs.
- Capture logs and telemetry without leaking secrets or sensitive security evidence.

## Module 7 — Qwen and Self-Managed Inference

- Compare data control, latency, cost, hardware, licensing and operational-support needs.
- Demonstrate a prepared Qwen endpoint or discuss a deployment pattern appropriate to available hardware.
- Check the selected model, serving backend and tool-calling support before reusing hosted-model code; API compatibility is not complete interchangeability.
- Discuss controlled inference, hybrid workflows and limitations of local deployment; local hosting alone does not establish compliance.

# Module 8 — Validation, Governance and Handover

- Validate generated scripts, extracted evidence and summaries; examine edge cases and incomplete information.
- Define data classification, prompt/output retention, credential handling and acceptable hosted-model use.
- Version prompts, code and skills; document workflow ownership and change-review practices.
- Review prototype traceability, cost, usefulness and maintainability.
- Identify tasks that should remain manual, production-readiness gaps and a measured improvement plan.

---

© 2026 Faysal Aziz · Kuala Lumpur