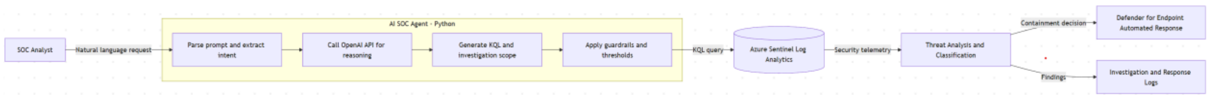
Threat Hunting and Remediation Using Agentic AI with Python and Microsoft Sentinel


AI SOC Agent - Python
SOC Analyst
Natural language request
Parse prompt and extract intent
Call OpenAI API for reasoning
Generate KQL and investigation scope
Apply guardrails and thresholds
KQL query
Azure Sentinel Log Analytics
Security telemetry
Threat Analysis and Classification
Containment decision
Defender for Endpoint Automated Response
Findings
Investigation and Response Logs

This project demonstrates the design and implementation of an AI-assisted SOC workflow that supports threat hunting, investigation, and automated response using Microsoft Sentinel, Defender for Endpoint, and OpenAI. The agent translates natural language security concerns into KQL queries, analyzes enterprise telemetry, and executes controlled response actions with strict cost and safety guardrails. The result is a scalable, low-cost automation approach that enhances analyst efficiency without sacrificing oversight.

Introduction

This document walks through the process of building and testing an AI-powered agent designed to support threat hunting and remediation workflows within a Security Operations Center (SOC).

The goal of this project is to explore how AI agents can be integrated into a real-world SOC analyst workflow—specifically, how they can assist with querying security telemetry, interpreting results, and accelerating investigation efforts.

While technical steps are covered, the emphasis is placed on **how the solution supports real-world security workflows**, rather than on low-level implementation details.

Starting/Preparing the Environment

To begin, a local development environment must be set up to support Python-based development and secure communication with both OpenAI and Microsoft Azure services.

Step 1: Install Python and Set Up the IDE

The first requirement is installing **Python** on a Windows machine. Python serves as the foundation for the AI agent and is used to orchestrate API calls, process security logs, and format outputs.

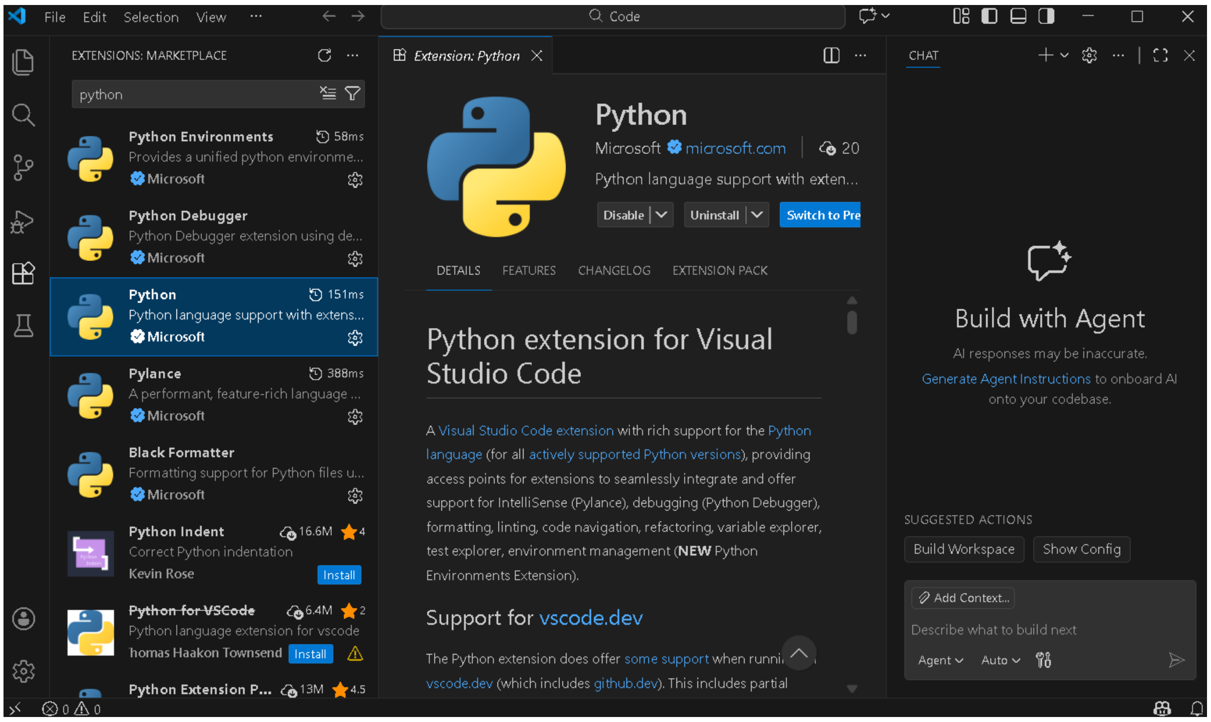
For development, **Visual Studio Code (VS Code)** was selected as the Integrated Development Environment (IDE) due to its lightweight nature and strong Python support.

To optimize VS Code for this project, the following extensions were installed from the VS Code Marketplace:

- **Python** – Enables Python syntax support, debugging, and execution

- **Black Formatter** – Automatically formats Python code to ensure consistency and readability

This setup provides a clean, efficient environment suitable for rapid experimentation and iteration.

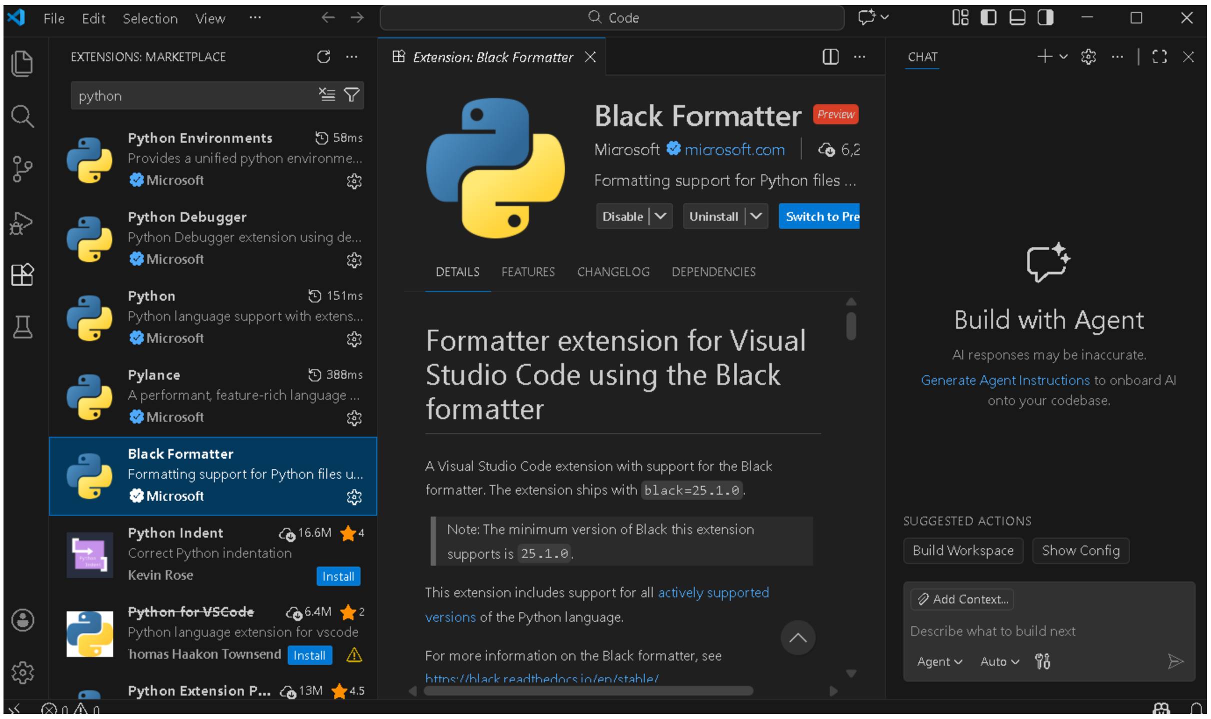

File
Edit
Selection
View
Code
EXTENSIONS: MARKETPLACE
python
Python Environments 58ms
Provides a unified python environme...
Microsoft
Python Debugger
Python Debugger extension using de...
Microsoft
Python 151ms
Python language support with extens...
Microsoft
Pylance 388ms
A performant, feature-rich language ...
Microsoft
Black Formatter
Formatting support for Python files u...
Microsoft
Python Indent 16.6M 4
Correct Python indentation
Kevin Rose
Install
~~Python for VSCode~~ 6.4M 2
Python language extension for vscode
homas Haakon Townsend
Install
Python Extension P... 13M 4.5
Extension: Python
Python
Microsoft microsoft.com 20
Python language support with exten...
Disable
Uninstall
Switch to Pre
DETAILS
FEATURES
CHANGELOG
EXTENSION PACK
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
SUGGESTED ACTIONS
Build Workspace
Show Config
Add Context...
Describe what to build next
Agent
Auto

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported Python versions), providing access points for extensions to seamlessly integrate and offer support for IntelliSense (Pylance), debugging (Python Debugger), formatting, linting, code navigation, refactoring, variable explorer, test explorer, environment management (**NEW** Python Environments Extension).

Support for vscode.dev

The Python extension does offer some support when runni... vscode.dev (which includes github.dev). This includes partial


File
Edit
Selection
View
Code
EXTENSIONS: MARKETPLACE
python
Python Environments 58ms
Provides a unified python environme...
Microsoft
Python Debugger
Python Debugger extension using de...
Microsoft
Python 151ms
Python language support with extens...
Microsoft
Pylance 388ms
A performant, feature-rich language ...
Microsoft
Black Formatter
Formatting support for Python files u...
Microsoft
Python Indent 16.6M 4
Correct Python indentation
Kevin Rose
Install
~~Python for VSCode~~ 6.4M 2
Python language extension for vscode
homas Haakon Townsend
Install
Python Extension P... 13M 4.5
Extension: Black Formatter
Black Formatter
Preview
Microsoft microsoft.com 6,2
Formatting support for Python files ...
Disable
Uninstall
Switch to Pre
DETAILS
FEATURES
CHANGELOG
DEPENDENCIES
CHAT
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
SUGGESTED ACTIONS
Build Workspace
Show Config
Add Context...
Describe what to build next
Agent
Auto

Formatter extension for Visual Studio Code using the Black formatter

A Visual Studio Code extension with support for the Black formatter. The extension ships with `black=25.1.0`.

> Note: The minimum version of Black this extension supports is `25.1.0`.

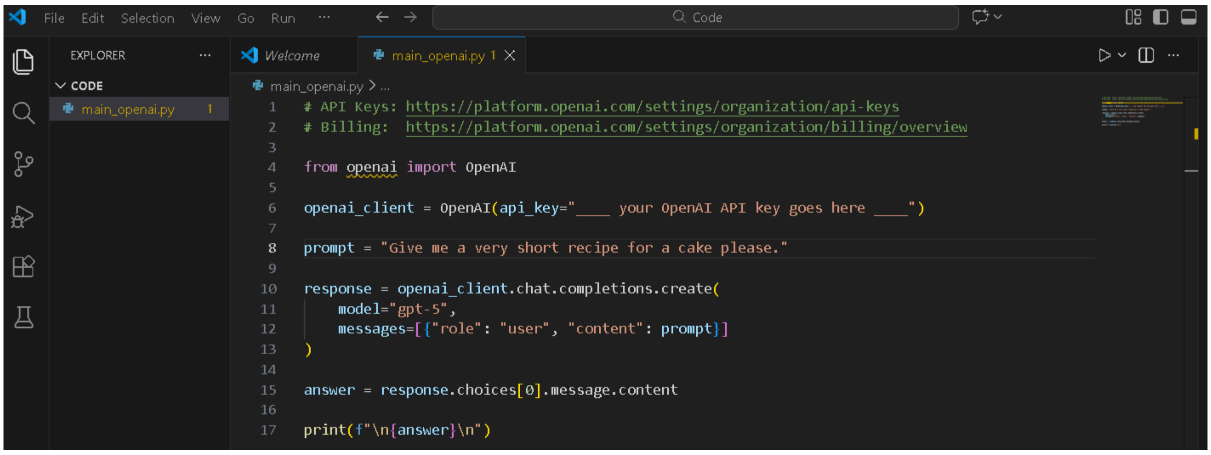
This extension includes support for all actively supported versions of the Python language.

For more information on the Black formatter, see https://black.readthedocs.io/en/stable/

Step 2: Preparing the AI Agent Template

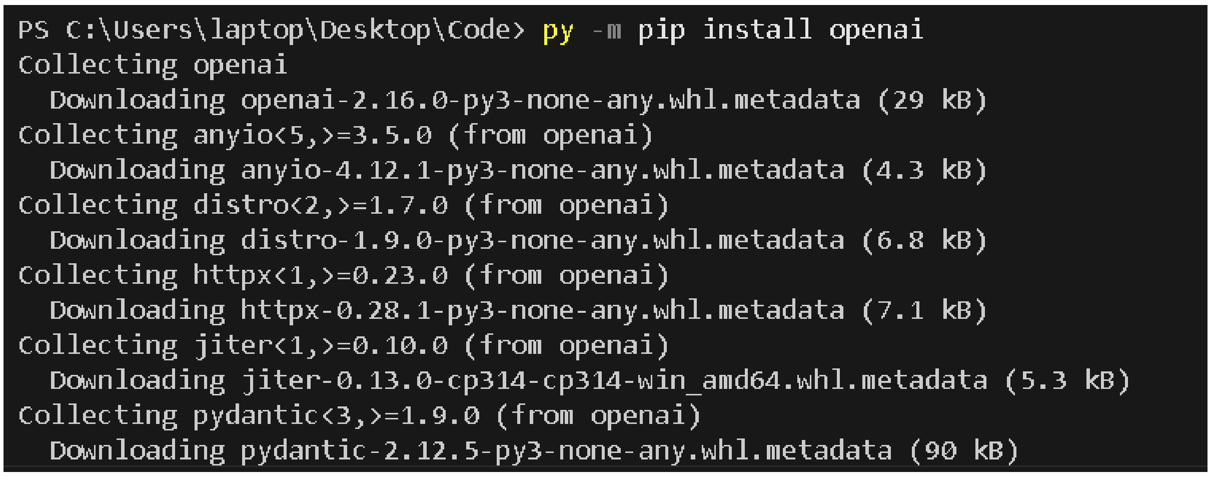
With the environment configured, a basic Python template is created to serve as the starting point for the AI agent. This template establishes the structure needed to send prompts, receive responses, and later integrate external data sources.

At this stage, an **OpenAI API key** is generated, and the account is funded with a small amount of usage credit. This key allows the program to securely communicate with OpenAI's services. Usage costs are minimal and are discussed later in the project.



```python
# API Keys: https://platform.openai.com/settings/organization/api-keys
# Billing:  https://platform.openai.com/settings/organization/billing/overview

from openai import OpenAI

openai_client = OpenAI(api_key="____ your OpenAI API key goes here ____")

prompt = "Give me a very short recipe for a cake please."

response = openai_client.chat.completions.create(
    model="gpt-5",
    messages=[{"role": "user", "content": prompt}]
)

answer = response.choices[0].message.content

print(f"\n{answer}\n")
```

Before inserting the API key into the code, it is necessary to install the OpenAI Python library. This is done using the following command:



```
PS C:\Users\laptop\Desktop\Code> py -m pip install openai
Collecting openai
  Downloading openai-2.16.0-py3-none-any.whl.metadata (29 kB)
Collecting anyio<5,>=3.5.0 (from openai)
  Downloading anyio-4.12.1-py3-none-any.whl.metadata (4.3 kB)
Collecting distro<2,>=1.7.0 (from openai)
  Downloading distro-1.9.0-py3-none-any.whl.metadata (6.8 kB)
Collecting httpx<1,>=0.23.0 (from openai)
  Downloading httpx-0.28.1-py3-none-any.whl.metadata (7.1 kB)
Collecting jiter<1,>=0.10.0 (from openai)
  Downloading jiter-0.13.0-cp314-cp314-win_amd64.whl.metadata (5.3 kB)
Collecting pydantic<3,>=1.9.0 (from openai)
  Downloading pydantic-2.12.5-py3-none-any.whl.metadata (90 kB)
```

Once installed, the API key is added to the appropriate location in the script.

Step 3: Verifying OpenAI API Connectivity

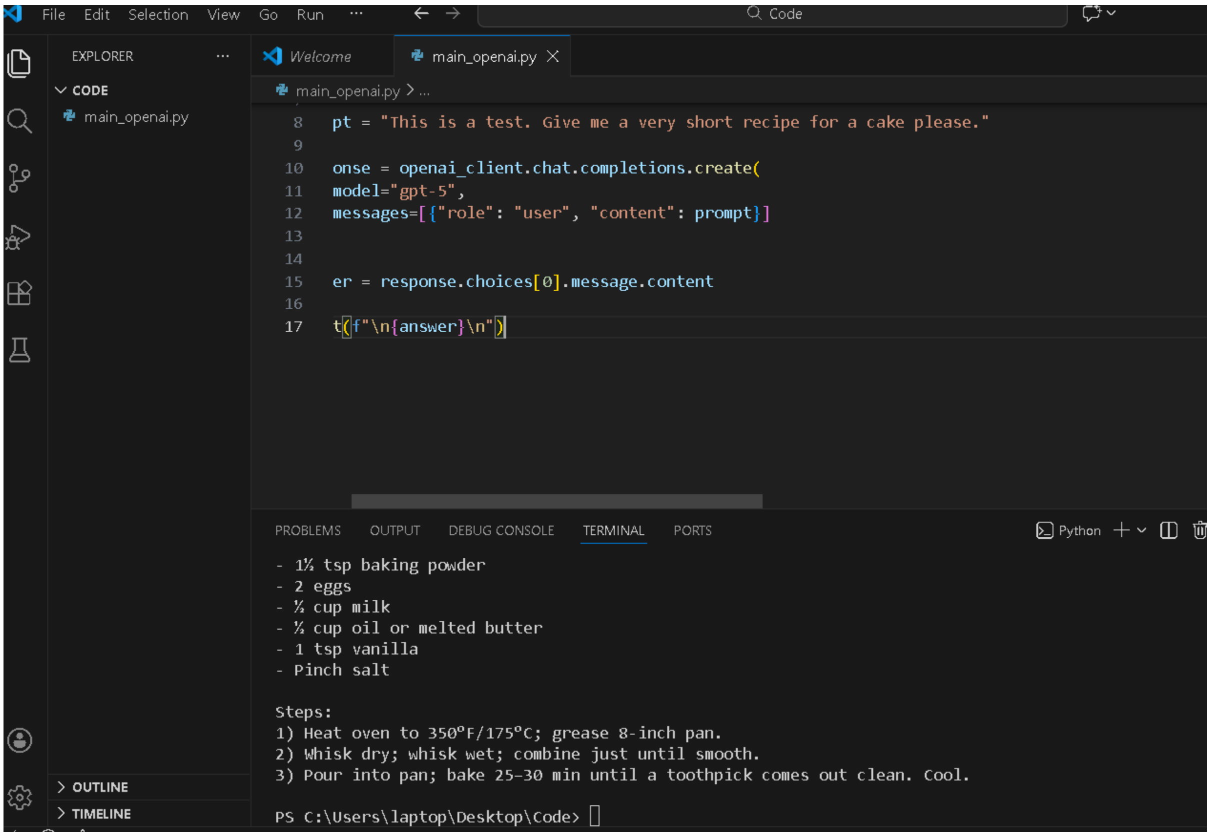
To confirm that the environment is configured correctly, a simple test prompt is sent to the OpenAI API:

“Give me a very short recipe for a cake.”

Receiving a valid response confirms that:

- The API key is working
- The program can successfully send requests
- Responses are being returned and processed correctly

This validation step ensures the AI agent's core communication mechanism is functioning before introducing additional complexity.



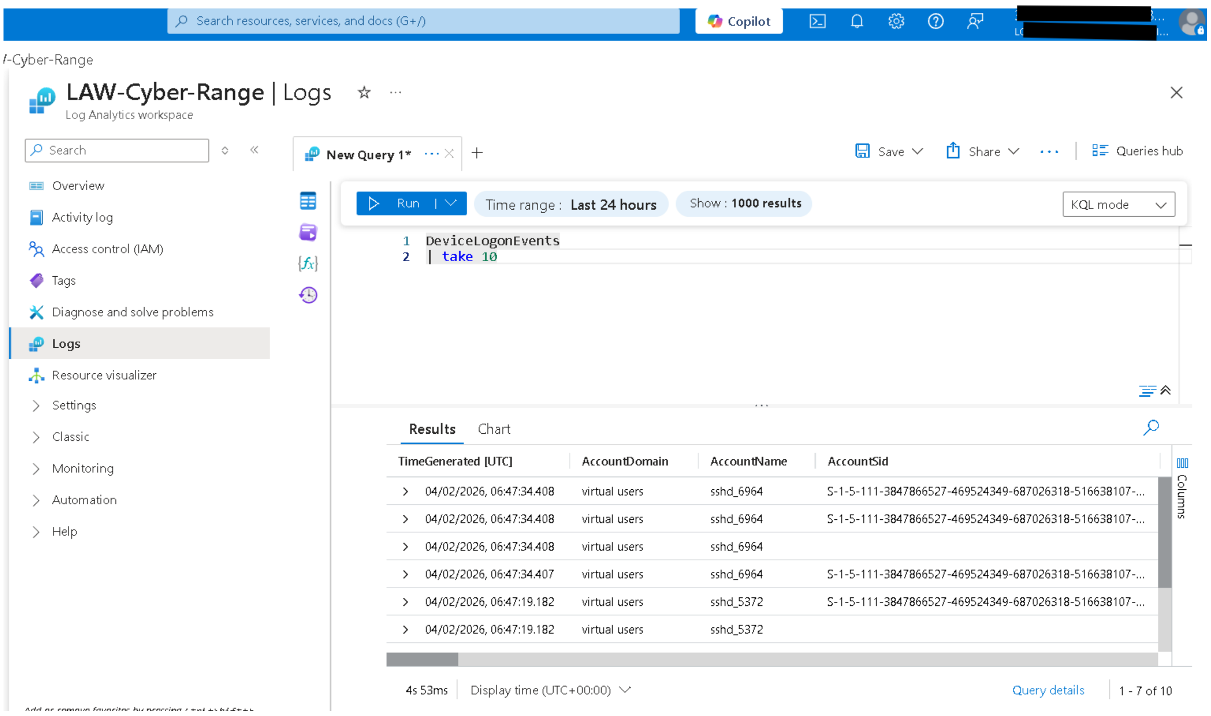
```
pt = "This is a test. Give me a very short recipe for a cake please."

onse = openai_client.chat.completions.create(
model="gpt-5",
messages=[{"role": "user", "content": prompt}]


er = response.choices[0].message.content

t(f"\n{answer}\n")
```

```
- 1½ tsp baking powder
- 2 eggs
- ½ cup milk
- ½ cup oil or melted butter
- 1 tsp vanilla
- Pinch salt

Steps:
1) Heat oven to 350°F/175°C; grease 8-inch pan.
2) Whisk dry; whisk wet; combine just until smooth.
3) Pour into pan; bake 25-30 min until a toothpick comes out clean. Cool.

PS C:\Users\laptop\Desktop\Code>
```

Step 4: Introducing the Security Lab Environment

For realistic testing, this project uses a controlled lab environment known as **Cyber Range**. This environment simulates an enterprise organization that is continuously exposed to security incidents and breaches, providing realistic telemetry for threat hunting scenarios.

Using a lab environment allows the AI agent to operate against real-world-style data without risk to a production system.



```
DeviceLogonEvents
| take 10
```

TimeGenerated [UTC]	AccountDomain	AccountName	AccountSid
04/02/2026, 06:47:34.408	virtual users	sshd_6964	S-1-5-111-3847866527-469524349-687026318-516638107-...
04/02/2026, 06:47:34.408	virtual users	sshd_6964	S-1-5-111-3847866527-469524349-687026318-516638107-...
04/02/2026, 06:47:34.408	virtual users	sshd_6964	
04/02/2026, 06:47:34.407	virtual users	sshd_6964	S-1-5-111-3847866527-469524349-687026318-516638107-...
04/02/2026, 06:47:19.182	virtual users	sshd_5372	S-1-5-111-3847866527-469524349-687026318-516638107-...
04/02/2026, 06:47:19.182	virtual users	sshd_5372	

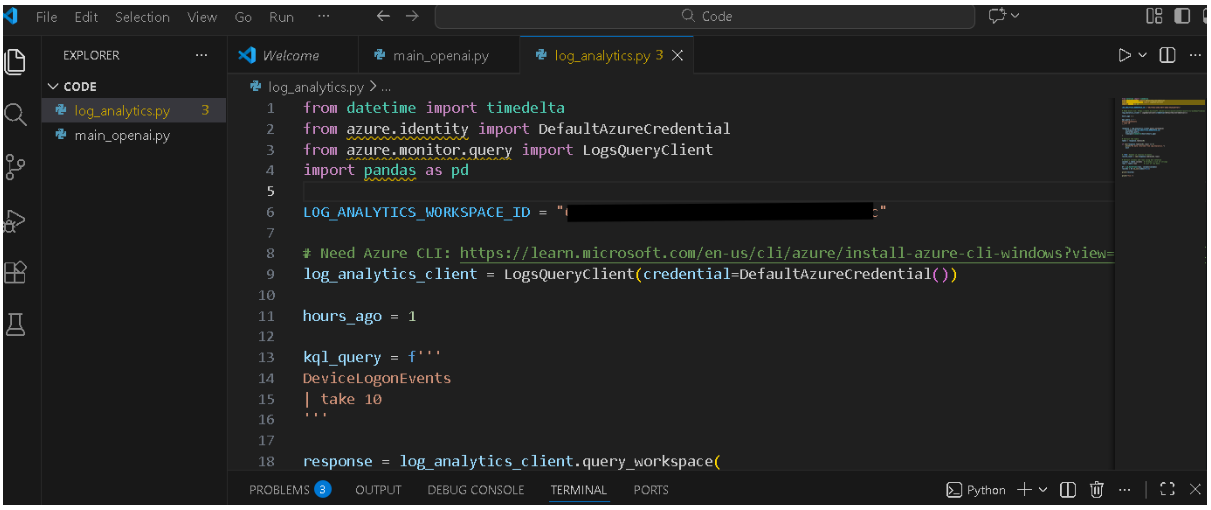
4s 53ms | Display time (UTC+00:00) | Query details | 1 - 7 of 10

Step 5: Connecting to Azure Log Analytics

The next step is integrating the AI agent with Microsoft Azure's **Log Analytics workspace**, which serves as the data backend for Microsoft Sentinel.

At this point, the project has:

- Verified connectivity to the OpenAI API
- Begun validating connectivity to Azure security data



```
from datetime import timedelta
from azure.identity import DefaultAzureCredential
from azure.monitor.query import LogsQueryClient
import pandas as pd

LOG_ANALYTICS_WORKSPACE_ID = "[redacted]"

# Need Azure CLI: https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=
log_analytics_client = LogsQueryClient(credential=DefaultAzureCredential())

hours_ago = 1

kql_query = f'''
DeviceLogonEvents
| take 10
'''

response = log_analytics_client.query_workspace(
```

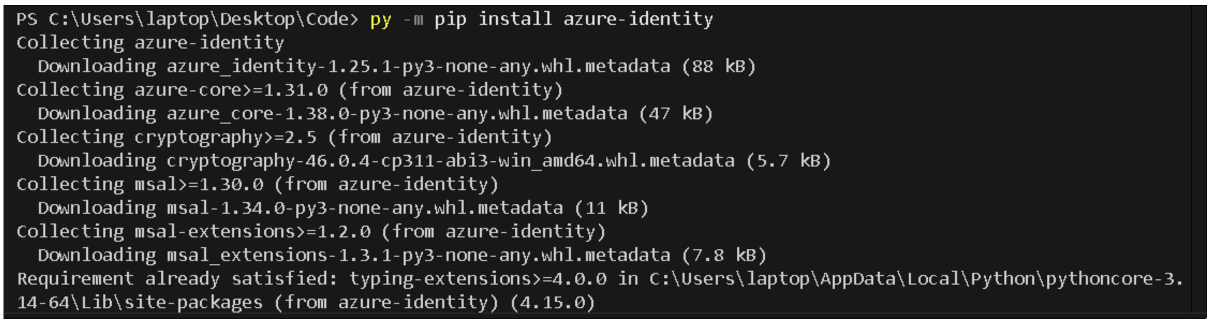
To enable communication with Azure services, several Python libraries are required:

py -m pip install azure-identity

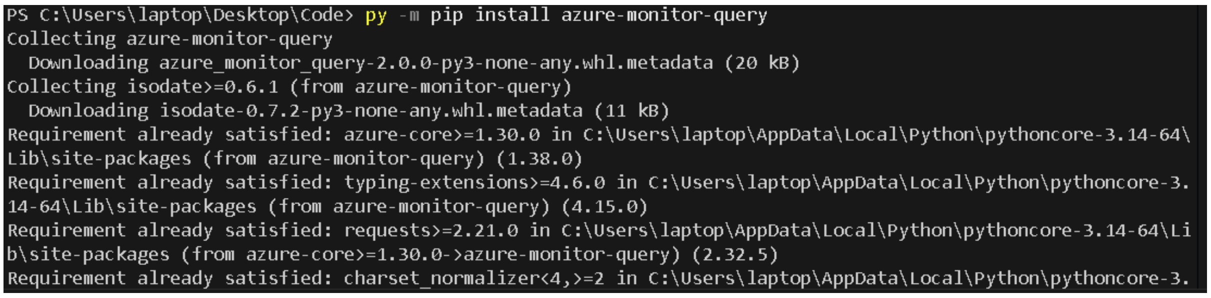
py -m pip install azure-monitor-query

py -m pip install pandas

- **azure-identity** handles authentication
- **azure-monitor-query** enables Log Analytics queries
- **pandas** is used to structure and format data in a human-readable way



```
PS C:\Users\laptop\Desktop\Code> py -m pip install azure-identity
Collecting azure-identity
  Downloading azure_identity-1.25.1-py3-none-any.whl.metadata (88 kB)
Collecting azure-core>=1.31.0 (from azure-identity)
  Downloading azure_core-1.38.0-py3-none-any.whl.metadata (47 kB)
Collecting cryptography>=2.5 (from azure-identity)
  Downloading cryptography-46.0.4-cp311-abi3-win_amd64.whl.metadata (5.7 kB)
Collecting msal>=1.30.0 (from azure-identity)
  Downloading msal-1.34.0-py3-none-any.whl.metadata (11 kB)
Collecting msal-extensions>=1.2.0 (from azure-identity)
  Downloading msal_extensions-1.3.1-py3-none-any.whl.metadata (7.8 kB)
Requirement already satisfied: typing-extensions>=4.0.0 in c:\Users\laptop\AppData\Local\Python\pythoncore-3.
14-64\Lib\site-packages (from azure-identity) (4.15.0)
```

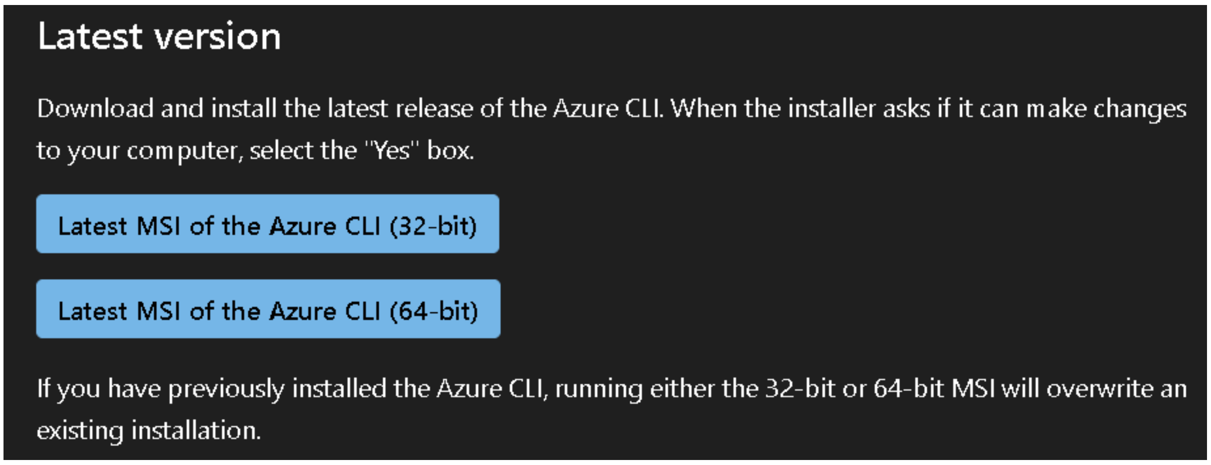


```
PS C:\Users\laptop\Desktop\Code> py -m pip install azure-monitor-query
Collecting azure-monitor-query
  Downloading azure_monitor_query-2.0.0-py3-none-any.whl.metadata (20 kB)
Collecting isodate>=0.6.1 (from azure-monitor-query)
  Downloading isodate-0.7.2-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: azure-core>=1.30.0 in c:\Users\laptop\AppData\Local\Python\pythoncore-3.14-64\
Lib\site-packages (from azure-monitor-query) (1.38.0)
Requirement already satisfied: typing-extensions>=4.6.0 in c:\Users\laptop\AppData\Local\Python\pythoncore-3.
14-64\Lib\site-packages (from azure-monitor-query) (4.15.0)
Requirement already satisfied: requests>=2.21.0 in c:\Users\laptop\AppData\Local\Python\pythoncore-3.14-64\Li
b\site-packages (from azure-core>=1.30.0->azure-monitor-query) (2.32.5)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\Users\laptop\AppData\Local\Python\pythoncore-3.
```

Step 6: Authenticating with Azure

In addition to Python libraries, the **Azure CLI** must be installed on the local machine. The CLI is used to authenticate against an Azure tenant that has:

- Microsoft Sentinel enabled
- A user account with appropriate permissions to query logs

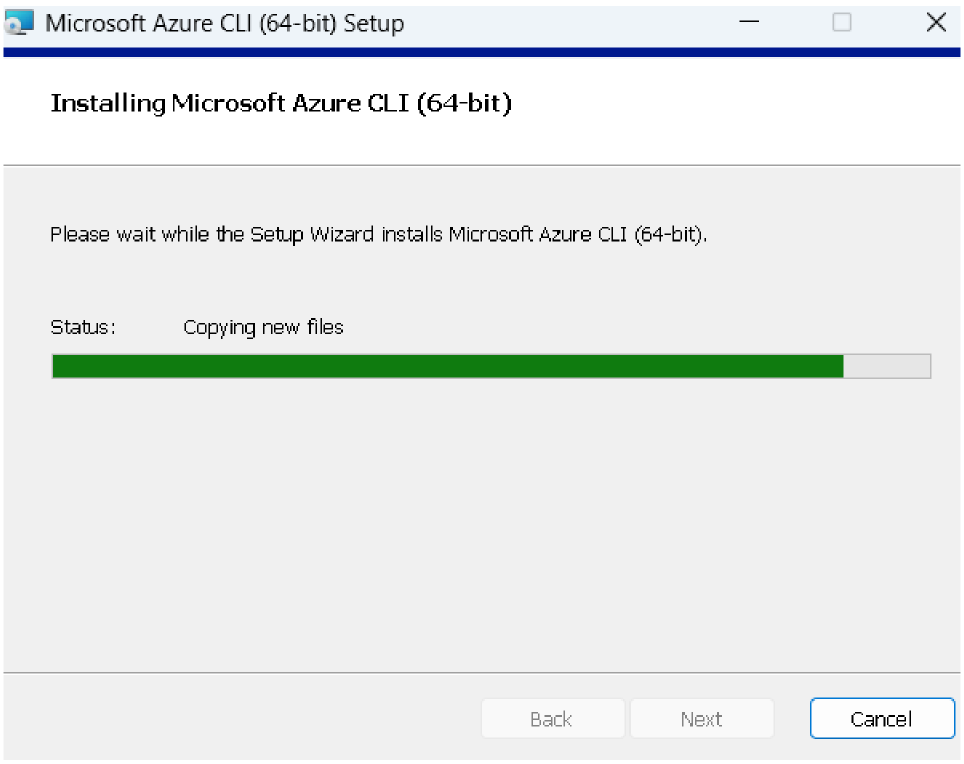


Latest version

Download and install the latest release of the Azure CLI. When the installer asks if it can make changes to your computer, select the "Yes" box.

Latest MSI of the Azure CLI (32-bit)
Latest MSI of the Azure CLI (64-bit)

If you have previously installed the Azure CLI, running either the 32-bit or 64-bit MSI will overwrite an existing installation.

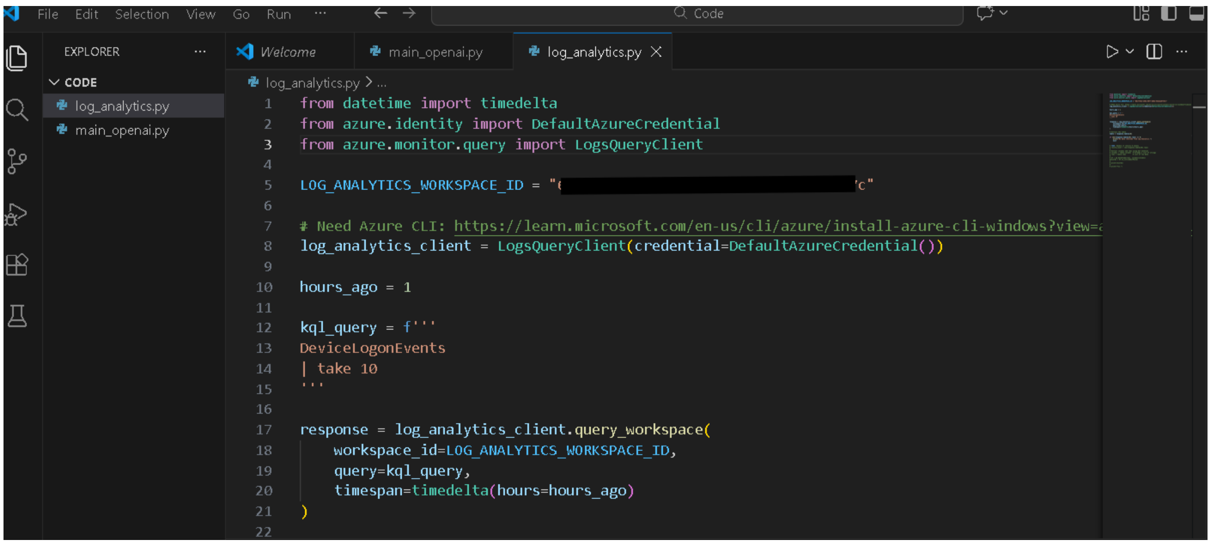

Microsoft Azure CLI (64-bit) Setup

Installing Microsoft Azure CLI (64-bit)

Please wait while the Setup Wizard installs Microsoft Azure CLI (64-bit).

Status: Copying new files

Once installed, the Azure CLI is connected to the tenant, allowing the Python script to inherit authenticated access.



```python
from datetime import timedelta
from azure.identity import DefaultAzureCredential
from azure.monitor.query import LogsQueryClient

LOG_ANALYTICS_WORKSPACE_ID = "[redacted]c"

# Need Azure CLI: https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=a
log_analytics_client = LogsQueryClient(credential=DefaultAzureCredential())

hours_ago = 1

kql_query = f'''
DeviceLogonEvents
| take 10
'''

response = log_analytics_client.query_workspace(
    workspace_id=LOG_ANALYTICS_WORKSPACE_ID,
    query=kql_query,
    timespan=timedelta(hours=hours_ago)
)

```

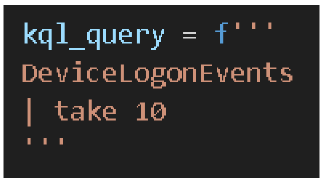
Step 7: Testing Log Analytics Queries

With authentication complete, a test query is executed against the Log Analytics workspace. The script sends a **KQL (Kusto Query Language)** query to Azure, retrieves the results, and outputs them within the program.

At this stage, the raw output confirms that:

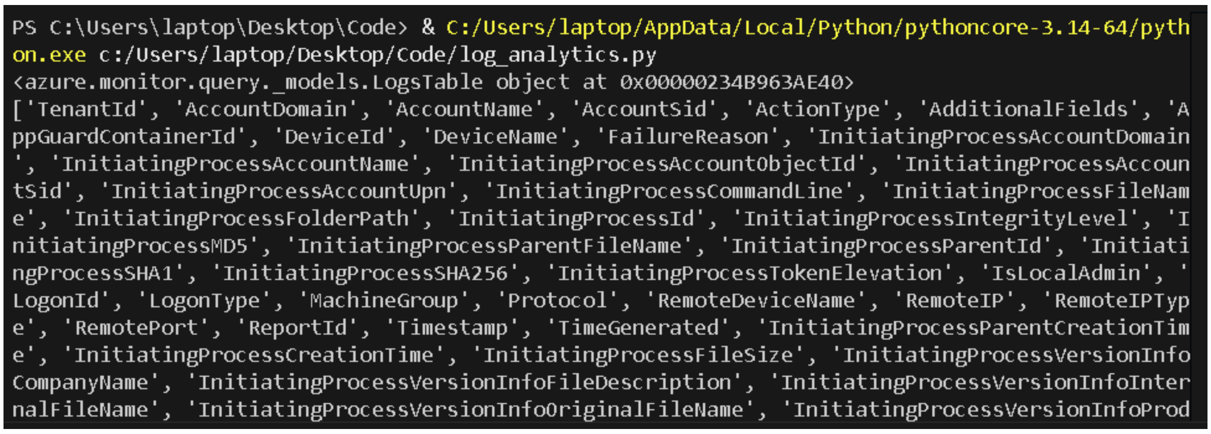
- The agent can successfully query Sentinel data
- Log data is being retrieved as expected

We can see the KQL query that the program will pass to the Azure Log analytics workspace instance.



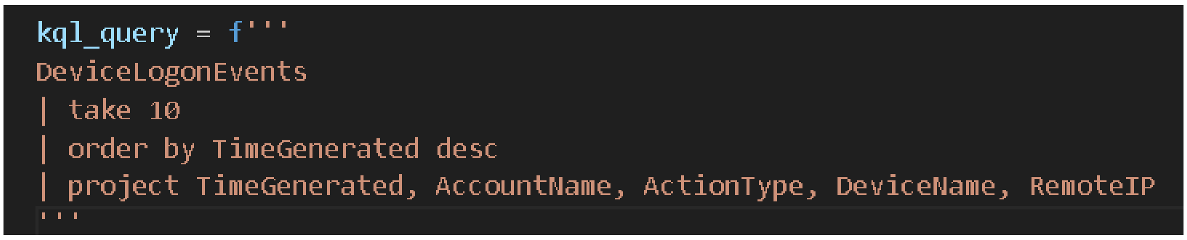
```
kql_query = f'''
DeviceLogonEvents
| take 10
'''
```

With the output being:

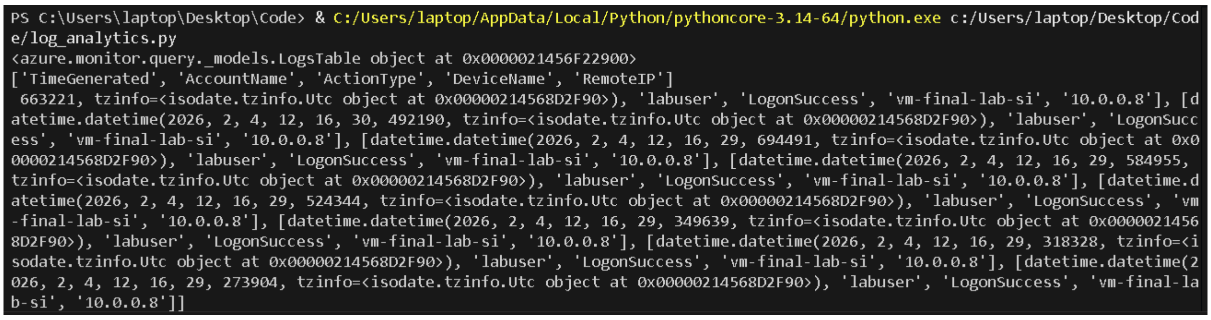


```
PS C:\Users\laptop\Desktop\Code> & C:/Users/laptop/AppData/Local/Python/pythoncore-3.14-64/pyth
on.exe c:/Users/laptop/Desktop/Code/log_analytics.py
<azure.monitor.query._models.LogsTable object at 0x00000234B963AE40>
['TenantId', 'AccountDomain', 'AccountName', 'AccountSid', 'ActionType', 'AdditionalFields', 'A
ppGuardContainerId', 'DeviceId', 'DeviceName', 'FailureReason', 'InitiatingProcessAccountDomain
', 'InitiatingProcessAccountName', 'InitiatingProcessAccountObjectId', 'InitiatingProcessAccoun
tSid', 'InitiatingProcessAccountUpn', 'InitiatingProcessCommandLine', 'InitiatingProcessFileNam
e', 'InitiatingProcessFolderPath', 'InitiatingProcessId', 'InitiatingProcessIntegrityLevel', 'I
nitiatingProcessMD5', 'InitiatingProcessParentFileName', 'InitiatingProcessParentId', 'Initiati
ngProcessSHA1', 'InitiatingProcessSHA256', 'InitiatingProcessTokenElevation', 'IsLocalAdmin', '
LogonId', 'LogonType', 'MachineGroup', 'Protocol', 'RemoteDeviceName', 'RemoteIP', 'RemoteIPTyp
e', 'RemotePort', 'ReportId', 'Timestamp', 'TimeGenerated', 'InitiatingProcessParentCreationTim
e', 'InitiatingProcessCreationTime', 'InitiatingProcessFileSize', 'InitiatingProcessVersionInfo
CompanyName', 'InitiatingProcessVersionInfoFileDescription', 'InitiatingProcessVersionInfoInter
nalFileName', 'InitiatingProcessVersionInfoOriginalFileName', 'InitiatingProcessVersionInfoProd
```

More complex queries are then tested to validate consistency and flexibility. These tests help confirm that the agent can handle increasingly realistic investigation scenarios.



```
kql_query = f'''
DeviceLogonEvents
| take 10
| order by TimeGenerated desc
| project TimeGenerated, AccountName, ActionType, DeviceName, RemoteIP
'''
```

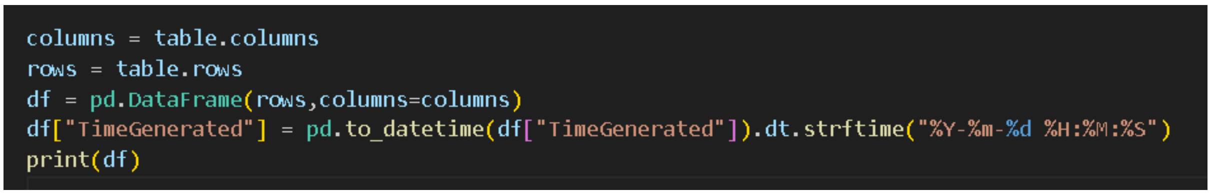


```
PS C:\Users\laptop\Desktop\Code> & C:/Users/laptop/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/laptop/Desktop/Cod
e/log_analytics.py
<azure.monitor.query._models.LogsTable object at 0x0000021456F22900>
['TimeGenerated', 'AccountName', 'ActionType', 'DeviceName', 'RemoteIP']
 663221, tzinfo=<isodate.tzinfo.Utc object at 0x00000214568D2F90>), 'labuser', 'LogonSuccess', 'vm-final-lab-si', '10.0.0.8'], [d
atetime.datetime(2026, 2, 4, 12, 16, 30, 492190, tzinfo=<isodate.tzinfo.Utc object at 0x00000214568D2F90>), 'labuser', 'LogonSucc
ess', 'vm-final-lab-si', '10.0.0.8'], [datetime.datetime(2026, 2, 4, 12, 16, 29, 694491, tzinfo=<isodate.tzinfo.Utc object at 0x0
000214568D2F90>), 'labuser', 'LogonSuccess', 'vm-final-lab-si', '10.0.0.8'], [datetime.datetime(2026, 2, 4, 12, 16, 29, 584955,
tzinfo=<isodate.tzinfo.Utc object at 0x00000214568D2F90>), 'labuser', 'LogonSuccess', 'vm-final-lab-si', '10.0.0.8'], [datetime.d
atetime(2026, 2, 4, 12, 16, 29, 524344, tzinfo=<isodate.tzinfo.Utc object at 0x00000214568D2F90>), 'labuser', 'LogonSuccess', 'vm
-final-lab-si', '10.0.0.8'], [datetime.datetime(2026, 2, 4, 12, 16, 29, 349639, tzinfo=<isodate.tzinfo.Utc object at 0x000002145
68D2F90>), 'labuser', 'LogonSuccess', 'vm-final-lab-si', '10.0.0.8'], [datetime.datetime(2026, 2, 4, 12, 16, 29, 318328, tzinfo=<i
sodate.tzinfo.Utc object at 0x00000214568D2F90>), 'labuser', 'LogonSuccess', 'vm-final-lab-si', '10.0.0.8'], [datetime.datetime(2
026, 2, 4, 12, 16, 29, 273904, tzinfo=<isodate.tzinfo.Utc object at 0x00000214568D2F90>), 'labuser', 'LogonSuccess', 'vm-final-la
b-si', '10.0.0.8']]
```

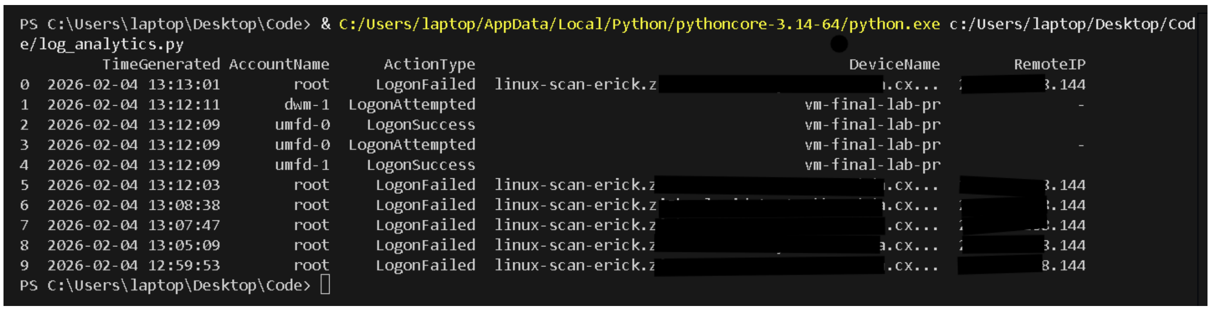
Step 8: Improving Data Readability

Finally, additional code is introduced using **pandas** to convert the raw query results into structured tables. This step transforms machine-oriented output into a format that is easier for humans to interpret and analyze.

Improving readability is a critical step, as it mirrors how SOC analysts and hiring managers would expect insights to be presented—clear, structured, and actionable.

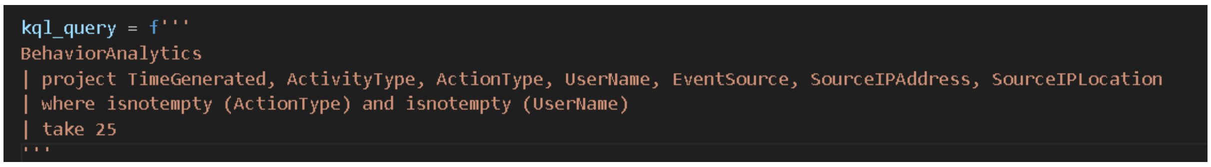


```python
columns = table.columns
rows = table.rows
df = pd.DataFrame(rows,columns=columns)
df["TimeGenerated"] = pd.to_datetime(df["TimeGenerated"]).dt.strftime("%Y-%m-%d %H:%M:%S")
print(df)
```

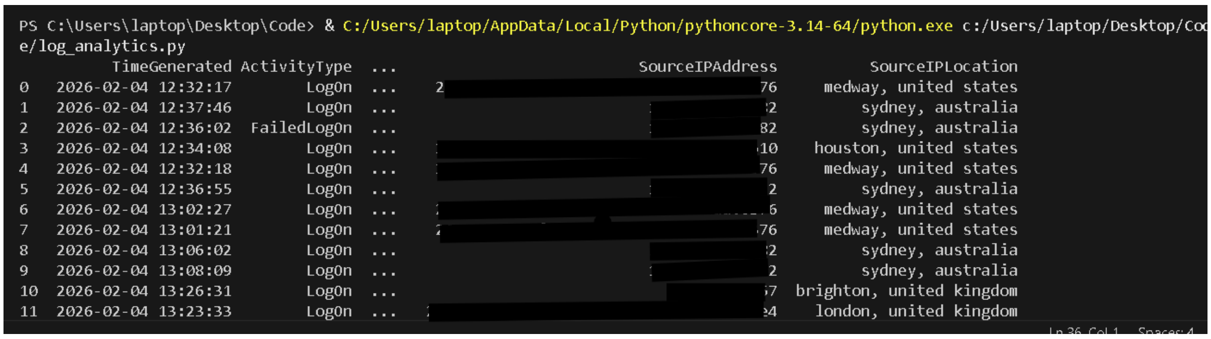


```
PS C:\Users\laptop\Desktop\Code> & C:/Users/laptop/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/laptop/Desktop/Cod
e/log_analytics.py
         TimeGenerated AccountName     ActionType                                          DeviceName       RemoteIP
0  2026-02-04 13:13:01        root    LogonFailed  linux-scan-erick.z[redacted]ı.cx...  [redacted]3.144
1  2026-02-04 13:12:11       dwm-1  LogonAttempted                                  vm-final-lab-pr              -
2  2026-02-04 13:12:09      umfd-0    LogonSuccess                                  vm-final-lab-pr
3  2026-02-04 13:12:09      umfd-0  LogonAttempted                                  vm-final-lab-pr              -
4  2026-02-04 13:12:09      umfd-1    LogonSuccess                                  vm-final-lab-pr
5  2026-02-04 13:12:03        root    LogonFailed  linux-scan-erick.z[redacted].cx...  [redacted]3.144
6  2026-02-04 13:08:38        root    LogonFailed  linux-scan-erick.z[redacted]ı.cx...  [redacted].144
7  2026-02-04 13:07:47        root    LogonFailed  linux-scan-erick.z[redacted].cx...  [redacted].144
8  2026-02-04 13:05:09        root    LogonFailed  linux-scan-erick.z[redacted]a.cx...  [redacted]3.144
9  2026-02-04 12:59:53        root    LogonFailed  linux-scan-erick.z[redacted].cx...  [redacted]8.144
PS C:\Users\laptop\Desktop\Code> 
```

We can be even more specific with our searches:



```python
kql_query = f'''
BehaviorAnalytics
| project TimeGenerated, ActivityType, ActionType, UserName, EventSource, SourceIPAddress, SourceIPLocation
| where isnotempty (ActionType) and isnotempty (UserName)
| take 25
'''
```



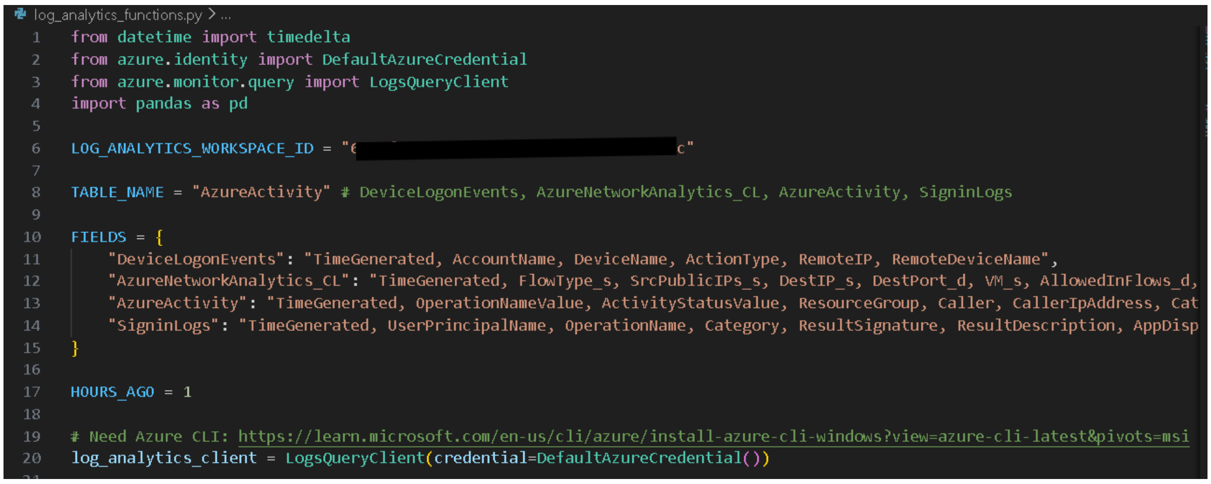
```
PS C:\Users\laptop\Desktop\Code> & C:/Users/laptop/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/laptop/Desktop/Coc
e/log_analytics.py
          TimeGenerated ActivityType  ...                   SourceIPAddress          SourceIPLocation
0   2026-02-04 12:32:17        LogOn  ...    2[redacted]76      medway, united states
1   2026-02-04 12:37:46        LogOn  ...              [redacted]2       sydney, australia
2   2026-02-04 12:36:02  FailedLogOn  ...              [redacted]82       sydney, australia
3   2026-02-04 12:34:08        LogOn  ...    [redacted]10     houston, united states
4   2026-02-04 12:32:18        LogOn  ...    [redacted]76      medway, united states
5   2026-02-04 12:36:55        LogOn  ...              [redacted]2       sydney, australia
6   2026-02-04 13:02:27        LogOn  ...    [redacted]6      medway, united states
7   2026-02-04 13:01:21        LogOn  ...    2[redacted]76      medway, united states
8   2026-02-04 13:06:02        LogOn  ...              [redacted]2       sydney, australia
9   2026-02-04 13:08:09        LogOn  ...              [redacted]2       sydney, australia
10  2026-02-04 13:26:31        LogOn  ...               [redacted]57   brighton, united kingdom
11  2026-02-04 13:23:33        LogOn  ...   [redacted]24     london, united kingdom
```

Adding the Agent

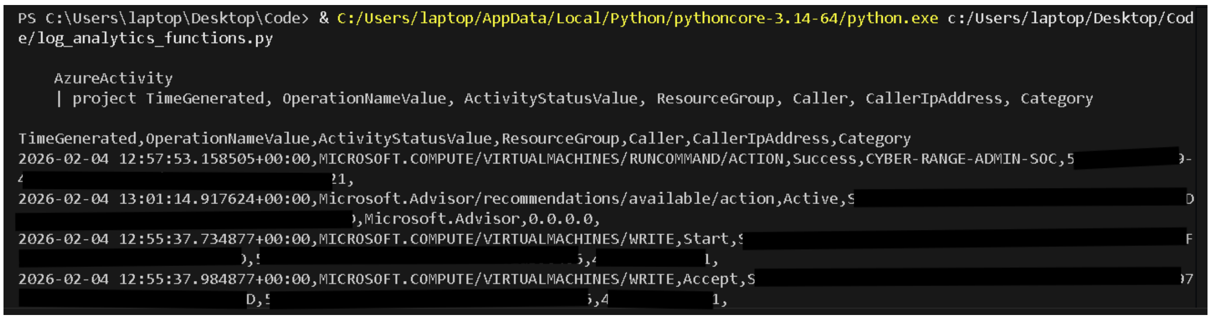
Step 1. The Groundwork for AI Functionality

At this stage, the focus shifts from environment setup to expanding the program's functionality and formally introducing the AI agent into the workflow.

Initially, the agent itself is not yet active within the program. Instead, the groundwork is laid by extending how data is retrieved from the Azure Log Analytics workspace. Additional tables are pulled into the application and exposed as variables within a dedicated function. This abstraction simplifies how the program requests data and maintains synchronization with the Log Analytics workspace.

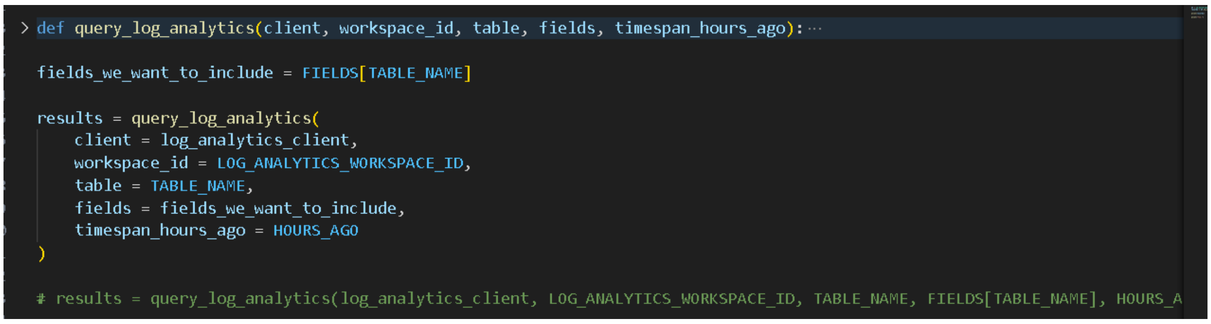


```
log_analytics_functions.py > ...
from datetime import timedelta
from azure.identity import DefaultAzureCredential
from azure.monitor.query import LogsQueryClient
import pandas as pd

LOG_ANALYTICS_WORKSPACE_ID = "6[redacted]c"

TABLE_NAME = "AzureActivity" # DeviceLogonEvents, AzureNetworkAnalytics_CL, AzureActivity, SigninLogs

FIELDS = {
    "DeviceLogonEvents": "TimeGenerated, AccountName, DeviceName, ActionType, RemoteIP, RemoteDeviceName",
    "AzureNetworkAnalytics_CL": "TimeGenerated, FlowType_s, SrcPublicIPs_s, DestIP_s, DestPort_d, VM_s, AllowedInFlows_d,
    "AzureActivity": "TimeGenerated, OperationNameValue, ActivityStatusValue, ResourceGroup, Caller, CallerIpAddress, Cat
    "SigninLogs": "TimeGenerated, UserPrincipalName, OperationName, Category, ResultSignature, ResultDescription, AppDisp
}

HOURS_AGO = 1

# Need Azure CLI: https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-windows?view=azure-cli-latest&pivots=msi
log_analytics_client = LogsQueryClient(credential=DefaultAzureCredential())
```

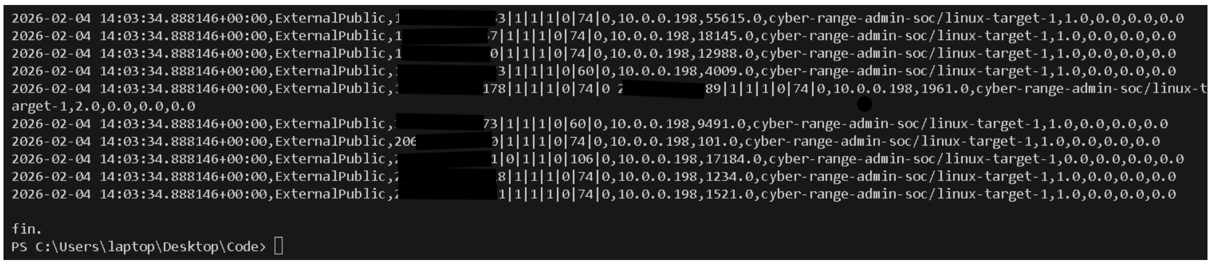


```
PS C:\Users\laptop\Desktop\Code> & C:/Users/laptop/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/laptop/Desktop/Cod
e/log_analytics_functions.py

    AzureActivity
    | project TimeGenerated, OperationNameValue, ActivityStatusValue, ResourceGroup, Caller, CallerIpAddress, Category

TimeGenerated,OperationNameValue,ActivityStatusValue,ResourceGroup,Caller,CallerIpAddress,Category
2026-02-04 12:57:53.158505+00:00,MICROSOFT.COMPUTE/VIRTUALMACHINES/RUNCOMMAND/ACTION,Success,CYBER-RANGE-ADMIN-SOC,5[redacted]9-
4[redacted]21,
2026-02-04 13:01:14.917624+00:00,Microsoft.Advisor/recommendations/available/action,Active,S[redacted]D
[redacted]),Microsoft.Advisor,0.0.0.0,
2026-02-04 12:55:37.734877+00:00,MICROSOFT.COMPUTE/VIRTUALMACHINES/WRITE,Start,S[redacted]F
[redacted]),5[redacted]5,4[redacted]1,
2026-02-04 12:55:37.984877+00:00,MICROSOFT.COMPUTE/VIRTUALMACHINES/WRITE,Accept,S[redacted]07
[redacted]D,5[redacted]5,4[redacted]1,
```

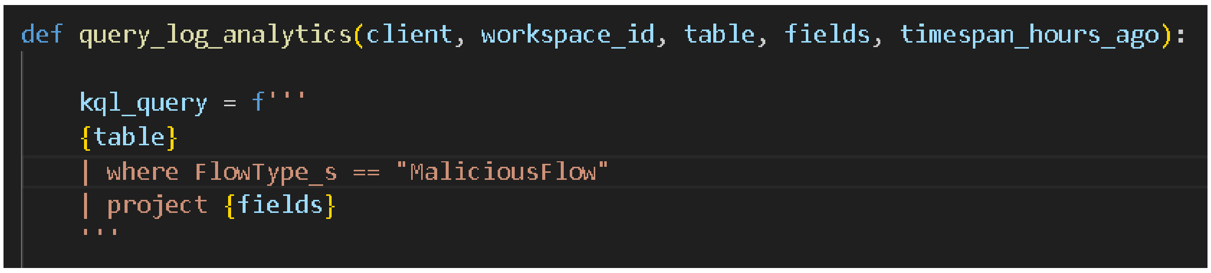
By structuring data access in this way, specific fields can be predefined, filtered, and manipulated directly from the derived tables returned by the function. This design choice allows the program to remain **modular and adaptable**, making it easy to tailor the agent's inputs to different environments, datasets, or investigative use cases without rewriting core logic.



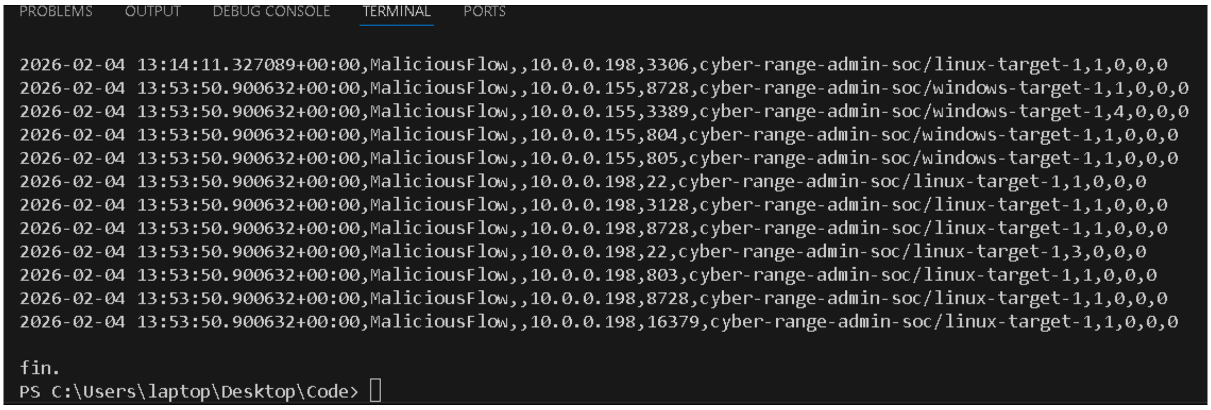
```
> def query_log_analytics(client, workspace_id, table, fields, timespan_hours_ago):...

fields_we_want_to_include = FIELDS[TABLE_NAME]

results = query_log_analytics(
    client = log_analytics_client,
    workspace_id = LOG_ANALYTICS_WORKSPACE_ID,
    table = TABLE_NAME,
    fields = fields_we_want_to_include,
    timespan_hours_ago = HOURS_AGO
)

# results = query_log_analytics(log_analytics_client, LOG_ANALYTICS_WORKSPACE_ID, TABLE_NAME, FIELDS[TABLE_NAME], HOURS_A
```



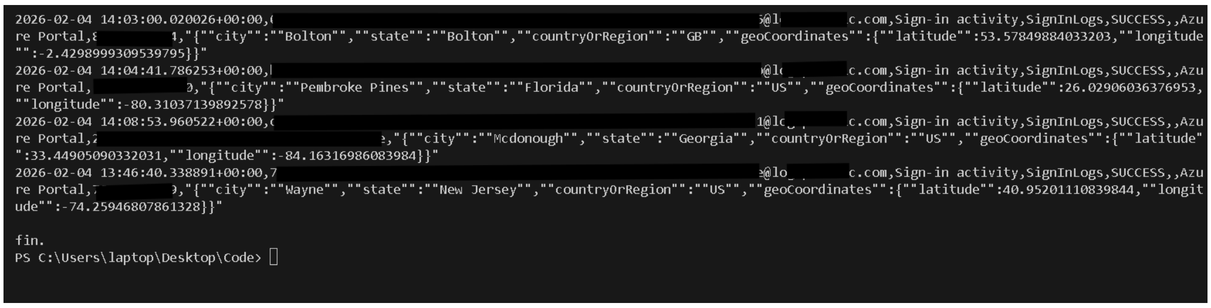
```
2026-02-04 14:03:34.888146+00:00,ExternalPublic,1[redacted]3|1|1|1|0|74|0,10.0.0.198,55615.0,cyber-range-admin-soc/linux-target-1,1.0,0.0,0.0,0.0
2026-02-04 14:03:34.888146+00:00,ExternalPublic,1[redacted]7|1|1|1|0|74|0,10.0.0.198,18145.0,cyber-range-admin-soc/linux-target-1,1.0,0.0,0.0,0.0
2026-02-04 14:03:34.888146+00:00,ExternalPublic,1[redacted]0|1|1|1|0|74|0,10.0.0.198,12988.0,cyber-range-admin-soc/linux-target-1,1.0,0.0,0.0,0.0
2026-02-04 14:03:34.888146+00:00,ExternalPublic,1[redacted]3|1|1|1|0|60|0,10.0.0.198,4009.0,cyber-range-admin-soc/linux-target-1,1.0,0.0,0.0,0.0
2026-02-04 14:03:34.888146+00:00,ExternalPublic,1[redacted]178|1|1|1|0|74|0 2[redacted]89|1|1|1|0|74|0,10.0.0.198,1961.0,cyber-range-admin-soc/linux-t
arget-1,2.0,0.0,0.0,0.0
2026-02-04 14:03:34.888146+00:00,ExternalPublic,1[redacted]73|1|1|1|0|60|0,10.0.0.198,9491.0,cyber-range-admin-soc/linux-target-1,1.0,0.0,0.0,0.0
2026-02-04 14:03:34.888146+00:00,ExternalPublic,206[redacted]0|1|1|1|0|74|0,10.0.0.198,101.0,cyber-range-admin-soc/linux-target-1,1.0,0.0,0.0,0.0
2026-02-04 14:03:34.888146+00:00,ExternalPublic,2[redacted]1|0|1|1|0|106|0,10.0.0.198,17184.0,cyber-range-admin-soc/linux-target-1,0.0,0.0,0.0,0.0
2026-02-04 14:03:34.888146+00:00,ExternalPublic,2[redacted]8|1|1|1|0|74|0,10.0.0.198,1234.0,cyber-range-admin-soc/linux-target-1,1.0,0.0,0.0,0.0
2026-02-04 14:03:34.888146+00:00,ExternalPublic,2[redacted]1|1|1|1|0|74|0,10.0.0.198,1521.0,cyber-range-admin-soc/linux-target-1,1.0,0.0,0.0,0.0

fin.
PS C:\Users\laptop\Desktop\Code> 
```



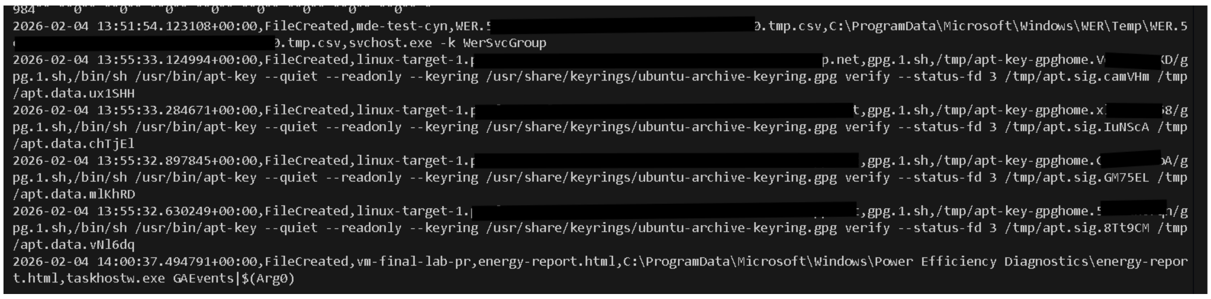
```
def query_log_analytics(client, workspace_id, table, fields, timespan_hours_ago):

    kql_query = f'''
    {table}
    | where FlowType_s == "MaliciousFlow"
    | project {fields}
    '''
```



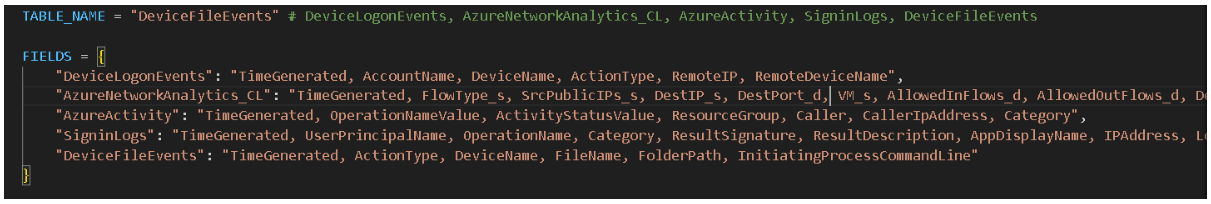
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

2026-02-04 13:14:11.327089+00:00,MaliciousFlow,,10.0.0.198,3306,cyber-range-admin-soc/linux-target-1,1,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.155,8728,cyber-range-admin-soc/windows-target-1,1,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.155,3389,cyber-range-admin-soc/windows-target-1,4,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.155,804,cyber-range-admin-soc/windows-target-1,1,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.155,805,cyber-range-admin-soc/windows-target-1,1,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.198,22,cyber-range-admin-soc/linux-target-1,1,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.198,3128,cyber-range-admin-soc/linux-target-1,1,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.198,8728,cyber-range-admin-soc/linux-target-1,1,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.198,22,cyber-range-admin-soc/linux-target-1,3,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.198,803,cyber-range-admin-soc/linux-target-1,1,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.198,8728,cyber-range-admin-soc/linux-target-1,1,0,0,0
2026-02-04 13:53:50.900632+00:00,MaliciousFlow,,10.0.0.198,16379,cyber-range-admin-soc/linux-target-1,1,0,0,0

fin.
PS C:\Users\laptop\Desktop\Code>
```



```
2026-02-04 14:03:00.020026+00:00,([redacted]5@l[redacted]c.com,Sign-in activity,SignInLogs,SUCCESS,,Azu
re Portal,8[redacted]4,"{""city"":""Bolton"",""state"":""Bolton"",""countryOrRegion"":""GB"",""geoCoordinates"":{""latitude"":53.57849884033203,""longitude
"":-2.4298999309539795}}"
2026-02-04 14:04:41.786253+00:00,[redacted]@l[redacted]c.com,Sign-in activity,SignInLogs,SUCCESS,,Azu
re Portal,[redacted]0,"{""city"":""Pembroke Pines"",""state"":""Florida"",""countryOrRegion"":""US"",""geoCoordinates"":{""latitude"":26.02906036376953,
""longitude"":-80.31037139892578}}"
2026-02-04 14:08:53.960522+00:00,([redacted]1@l[redacted]c.com,Sign-in activity,SignInLogs,SUCCESS,,Azu
re Portal,2[redacted]e,"{""city"":""Mcdonough"",""state"":""Georgia"",""countryOrRegion"":""US"",""geoCoordinates"":{""latitude
"":33.44905090332031,""longitude"":-84.16316986083984}}"
2026-02-04 13:46:40.338891+00:00,7[redacted]e@lo[redacted]c.com,Sign-in activity,SignInLogs,SUCCESS,,Azu
re Portal,7[redacted]9,"{""city"":""Wayne"",""state"":""New Jersey"",""countryOrRegion"":""US"",""geoCoordinates"":{""latitude"":40.95201110839844,""longit
ude"":-74.25946807861328}}"

fin.
PS C:\Users\laptop\Desktop\Code>
```



```
2026-02-04 13:51:54.123108+00:00,FileCreated,mde-test-cyn,WER.5[redacted]0.tmp.csv,C:\ProgramData\Microsoft\Windows\WER\Temp\WER.5
[redacted]0.tmp.csv,svchost.exe -k WerSvcGroup
2026-02-04 13:55:33.124994+00:00,FileCreated,linux-target-1.p[redacted]p.net,gpg.1.sh,/tmp/apt-key-gpghome.V[redacted]KD/g
pg.1.sh,/bin/sh /usr/bin/apt-key --quiet --readonly --keyring /usr/share/keyrings/ubuntu-archive-keyring.gpg verify --status-fd 3 /tmp/apt.sig.camVHm /tmp
/apt.data.ux1SHH
2026-02-04 13:55:33.284671+00:00,FileCreated,linux-target-1.p[redacted]t,gpg.1.sh,/tmp/apt-key-gpghome.x[redacted]8/g
pg.1.sh,/bin/sh /usr/bin/apt-key --quiet --readonly --keyring /usr/share/keyrings/ubuntu-archive-keyring.gpg verify --status-fd 3 /tmp/apt.sig.IuNScA /tmp
/apt.data.chTjEl
2026-02-04 13:55:32.897845+00:00,FileCreated,linux-target-1.p[redacted],gpg.1.sh,/tmp/apt-key-gpghome.[redacted]A/g
pg.1.sh,/bin/sh /usr/bin/apt-key --quiet --readonly --keyring /usr/share/keyrings/ubuntu-archive-keyring.gpg verify --status-fd 3 /tmp/apt.sig.GM75EL /tmp
/apt.data.mlKhRD
2026-02-04 13:55:32.630249+00:00,FileCreated,linux-target-1.[redacted],gpg.1.sh,/tmp/apt-key-gpghome.5[redacted]n/g
pg.1.sh,/bin/sh /usr/bin/apt-key --quiet --readonly --keyring /usr/share/keyrings/ubuntu-archive-keyring.gpg verify --status-fd 3 /tmp/apt.sig.8Tt9CM /tmp
/apt.data.vNl6dq
2026-02-04 14:00:37.494791+00:00,FileCreated,vm-final-lab-pr,energy-report.html,C:\ProgramData\Microsoft\Windows\Power Efficiency Diagnostics\energy-repor
t.html,taskhostw.exe GAEvents|$(Arg0)
```



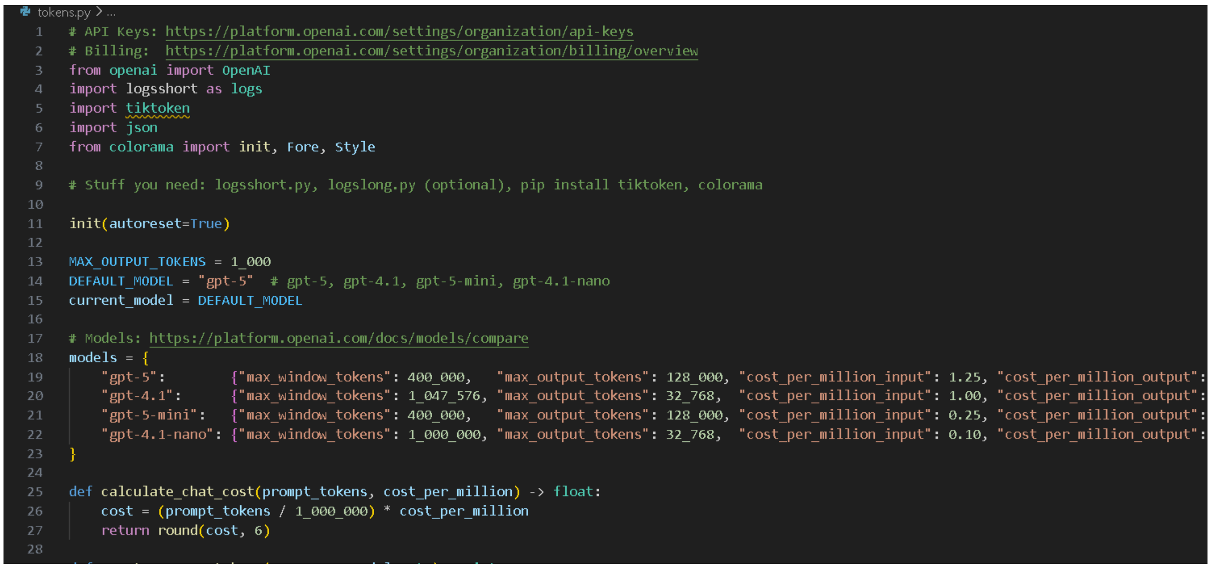
```
TABLE_NAME = "DeviceFileEvents" # DeviceLogonEvents, AzureNetworkAnalytics_CL, AzureActivity, SigninLogs, DeviceFileEvents

FIELDS = {
    "DeviceLogonEvents": "TimeGenerated, AccountName, DeviceName, ActionType, RemoteIP, RemoteDeviceName",
    "AzureNetworkAnalytics_CL": "TimeGenerated, FlowType_s, SrcPublicIPs_s, DestIP_s, DestPort_d, VM_s, AllowedInFlows_d, AllowedOutFlows_d, De
    "AzureActivity": "TimeGenerated, OperationNameValue, ActivityStatusValue, ResourceGroup, Caller, CallerIpAddress, Category",
    "SigninLogs": "TimeGenerated, UserPrincipalName, OperationName, Category, ResultSignature, ResultDescription, AppDisplayName, IPAddress, Lo
    "DeviceFileEvents": "TimeGenerated, ActionType, DeviceName, FileName, FolderPath, InitiatingProcessCommandLine"
}
```

Step 2. Introducing AI Functionality to the program

With the data layer in place, AI capabilities are then integrated into the program. Initial testing focuses on **cost awareness and model selection**, ensuring the solution remains practical for ongoing use.

The program is designed to support multiple AI models, allowing the user to specify which model should be used at runtime. Cost-efficient models are defined explicitly within the code, giving flexibility to balance performance and operational expense.

For this implementation, **GPT-5** is selected as the active model.



tokens.py > ...

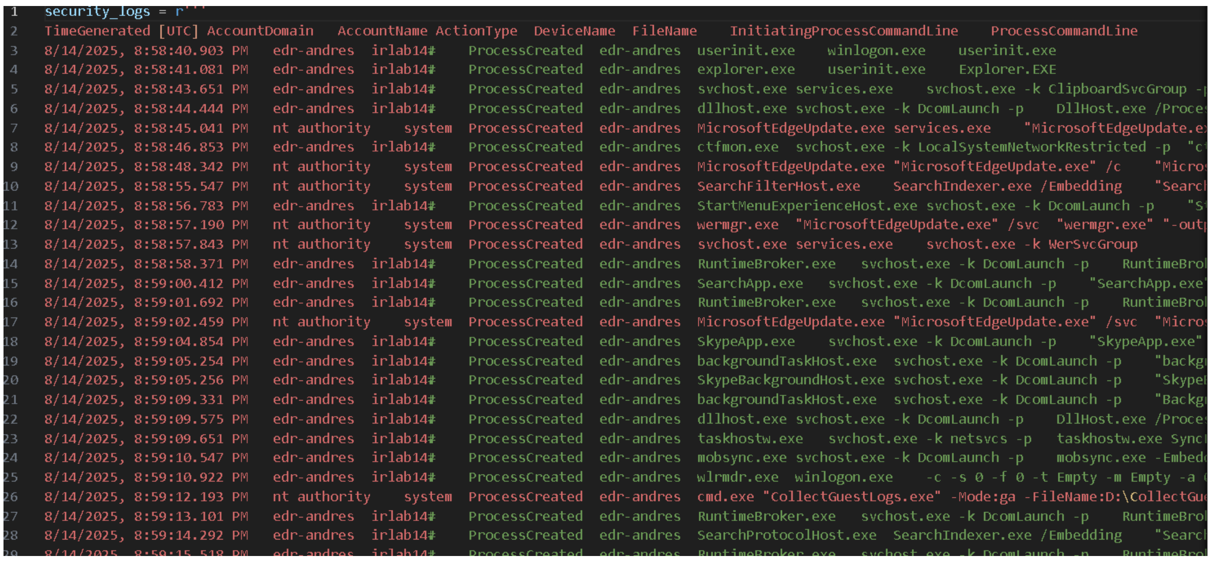
```python
# API Keys: https://platform.openai.com/settings/organization/api-keys
# Billing:  https://platform.openai.com/settings/organization/billing/overview
from openai import OpenAI
import logsshort as logs
import tiktoken
import json
from colorama import init, Fore, Style

# Stuff you need: logsshort.py, logslong.py (optional), pip install tiktoken, colorama

init(autoreset=True)

MAX_OUTPUT_TOKENS = 1_000
DEFAULT_MODEL = "gpt-5"  # gpt-5, gpt-4.1, gpt-5-mini, gpt-4.1-nano
current_model = DEFAULT_MODEL

# Models: https://platform.openai.com/docs/models/compare
models = {
    "gpt-5":        {"max_window_tokens": 400_000,   "max_output_tokens": 128_000, "cost_per_million_input": 1.25, "cost_per_million_output":
    "gpt-4.1":      {"max_window_tokens": 1_047_576, "max_output_tokens": 32_768,  "cost_per_million_input": 1.00, "cost_per_million_output":
    "gpt-5-mini":   {"max_window_tokens": 400_000,   "max_output_tokens": 128_000, "cost_per_million_input": 0.25, "cost_per_million_output":
    "gpt-4.1-nano": {"max_window_tokens": 1_000_000, "max_output_tokens": 32_768,  "cost_per_million_input": 0.10, "cost_per_million_output":
}

def calculate_chat_cost(prompt_tokens, cost_per_million) -> float:
    cost = (prompt_tokens / 1_000_000) * cost_per_million
    return round(cost, 6)
```

This modular approach to model selection allows the program to evolve over time, making it straightforward to switch models or adjust configurations as requirements change.

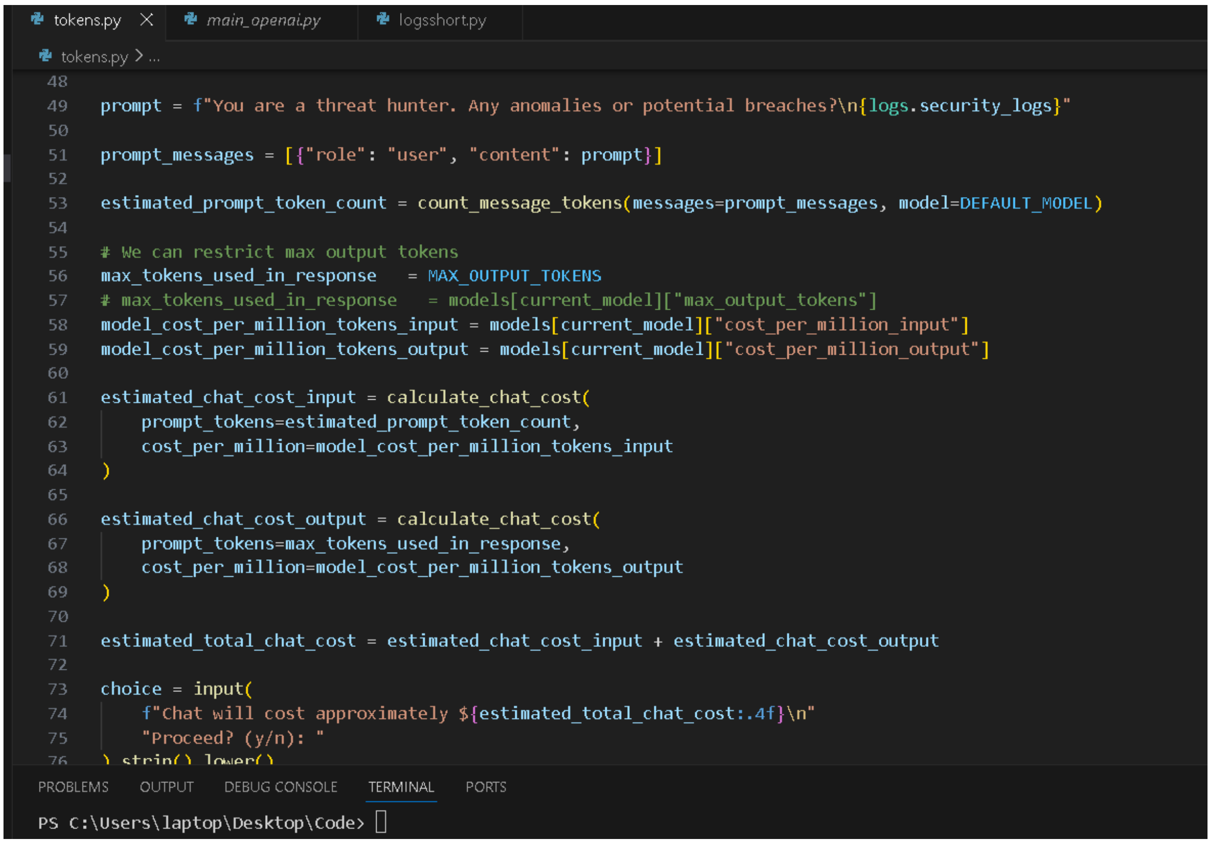
Testing the Agent

Step 1. Evaluating Inputs and Managing Cost

With the AI model integrated, the next step is providing the agent with data to analyze.



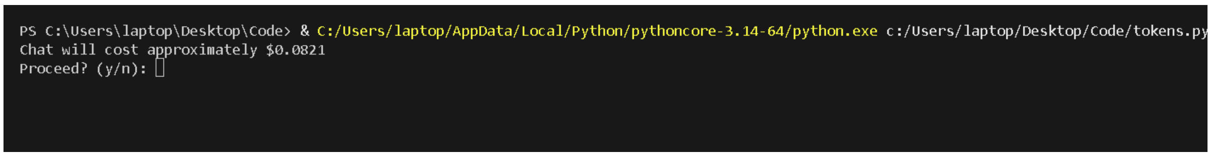
```
security_logs = r'''
TimeGenerated [UTC] AccountDomain   AccountName ActionType  DeviceName  FileName    InitiatingProcessCommandLine    ProcessCommandLine
8/14/2025, 8:58:40.903 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  userinit.exe    winlogon.exe    userinit.exe
8/14/2025, 8:58:41.081 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  explorer.exe    userinit.exe    Explorer.EXE
8/14/2025, 8:58:43.651 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  svchost.exe services.exe    svchost.exe -k ClipboardSvcGroup -
8/14/2025, 8:58:44.444 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  dllhost.exe svchost.exe -k DcomLaunch -p    DllHost.exe /Proce
8/14/2025, 8:58:45.041 PM   nt authority    system  ProcessCreated  edr-andres  MicrosoftEdgeUpdate.exe services.exe    "MicrosoftEdgeUpdate.e
8/14/2025, 8:58:46.853 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  ctfmon.exe  svchost.exe -k LocalSystemNetworkRestricted -p  "c
8/14/2025, 8:58:48.342 PM   nt authority    system  ProcessCreated  edr-andres  MicrosoftEdgeUpdate.exe "MicrosoftEdgeUpdate.exe" /c    "Micro
8/14/2025, 8:58:55.547 PM   nt authority    system  ProcessCreated  edr-andres  SearchFilterHost.exe    SearchIndexer.exe /Embedding    "Searc
8/14/2025, 8:58:56.783 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  StartMenuExperienceHost.exe svchost.exe -k DcomLaunch -p    "S
8/14/2025, 8:58:57.190 PM   nt authority    system  ProcessCreated  edr-andres  wermgr.exe  "MicrosoftEdgeUpdate.exe" /svc  "wermgr.exe" "-out
8/14/2025, 8:58:57.843 PM   nt authority    system  ProcessCreated  edr-andres  svchost.exe services.exe    svchost.exe -k WerSvcGroup
8/14/2025, 8:58:58.371 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  RuntimeBroker.exe   svchost.exe -k DcomLaunch -p    RuntimeBro
8/14/2025, 8:59:00.412 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  SearchApp.exe   svchost.exe -k DcomLaunch -p    "SearchApp.exe
8/14/2025, 8:59:01.692 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  RuntimeBroker.exe   svchost.exe -k DcomLaunch -p    RuntimeBro
8/14/2025, 8:59:02.459 PM   nt authority    system  ProcessCreated  edr-andres  MicrosoftEdgeUpdate.exe "MicrosoftEdgeUpdate.exe" /svc  "Micro
8/14/2025, 8:59:04.854 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  SkypeApp.exe    svchost.exe -k DcomLaunch -p    "SkypeApp.exe"
8/14/2025, 8:59:05.254 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  backgroundTaskHost.exe  svchost.exe -k DcomLaunch -p    "backg
8/14/2025, 8:59:05.256 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  SkypeBackgroundHost.exe svchost.exe -k DcomLaunch -p    "Skype
8/14/2025, 8:59:09.331 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  backgroundTaskHost.exe  svchost.exe -k DcomLaunch -p    "Backg
8/14/2025, 8:59:09.575 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  dllhost.exe svchost.exe -k DcomLaunch -p    DllHost.exe /Proce
8/14/2025, 8:59:09.651 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  taskhostw.exe   svchost.exe -k netsvcs -p   taskhostw.exe Sync
8/14/2025, 8:59:10.547 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  mobsync.exe svchost.exe -k DcomLaunch -p    mobsync.exe -Embed
8/14/2025, 8:59:10.922 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  wlrmdr.exe  winlogon.exe    -c -s 0 -f 0 -t Empty -m Empty -a
8/14/2025, 8:59:12.193 PM   nt authority    system  ProcessCreated  edr-andres  cmd.exe "CollectGuestLogs.exe" -Mode:ga -FileName:D:\CollectGu
8/14/2025, 8:59:13.101 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  RuntimeBroker.exe   svchost.exe -k DcomLaunch -p    RuntimeBro
8/14/2025, 8:59:14.292 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  SearchProtocolHost.exe  SearchIndexer.exe /Embedding    "Searc
8/14/2025, 8:59:15.518 PM   edr-andres  irlab14#    ProcessCreated  edr-andres  RuntimeBroker.exe   svchost.exe -k DcomLaunch -p    RuntimeBro
```



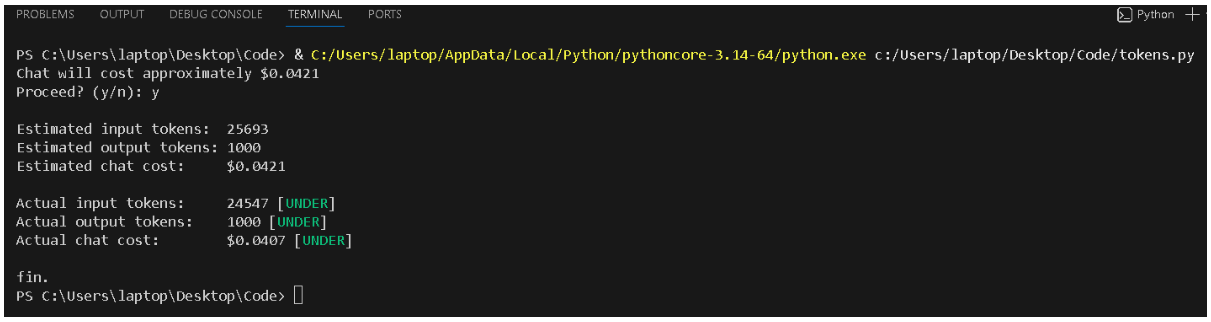
```python

prompt = f"You are a threat hunter. Any anomalies or potential breaches?\n{logs.security_logs}"

prompt_messages = [{"role": "user", "content": prompt}]

estimated_prompt_token_count = count_message_tokens(messages=prompt_messages, model=DEFAULT_MODEL)

# We can restrict max output tokens
max_tokens_used_in_response    = MAX_OUTPUT_TOKENS
# max_tokens_used_in_response    = models[current_model]["max_output_tokens"]
model_cost_per_million_tokens_input = models[current_model]["cost_per_million_input"]
model_cost_per_million_tokens_output = models[current_model]["cost_per_million_output"]

estimated_chat_cost_input = calculate_chat_cost(
    prompt_tokens=estimated_prompt_token_count,
    cost_per_million=model_cost_per_million_tokens_input
)

estimated_chat_cost_output = calculate_chat_cost(
    prompt_tokens=max_tokens_used_in_response,
    cost_per_million=model_cost_per_million_tokens_output
)

estimated_total_chat_cost = estimated_chat_cost_input + estimated_chat_cost_output

choice = input(
    f"Chat will cost approximately ${estimated_total_chat_cost:.4f}\n"
    "Proceed? (y/n): "
).strip().lower()
```

```
PS C:\Users\laptop\Desktop\Code>
```

Before any analysis is performed, the program evaluates the projected cost of the request. This ensures that each investigation is assessed for resource usage prior to execution.

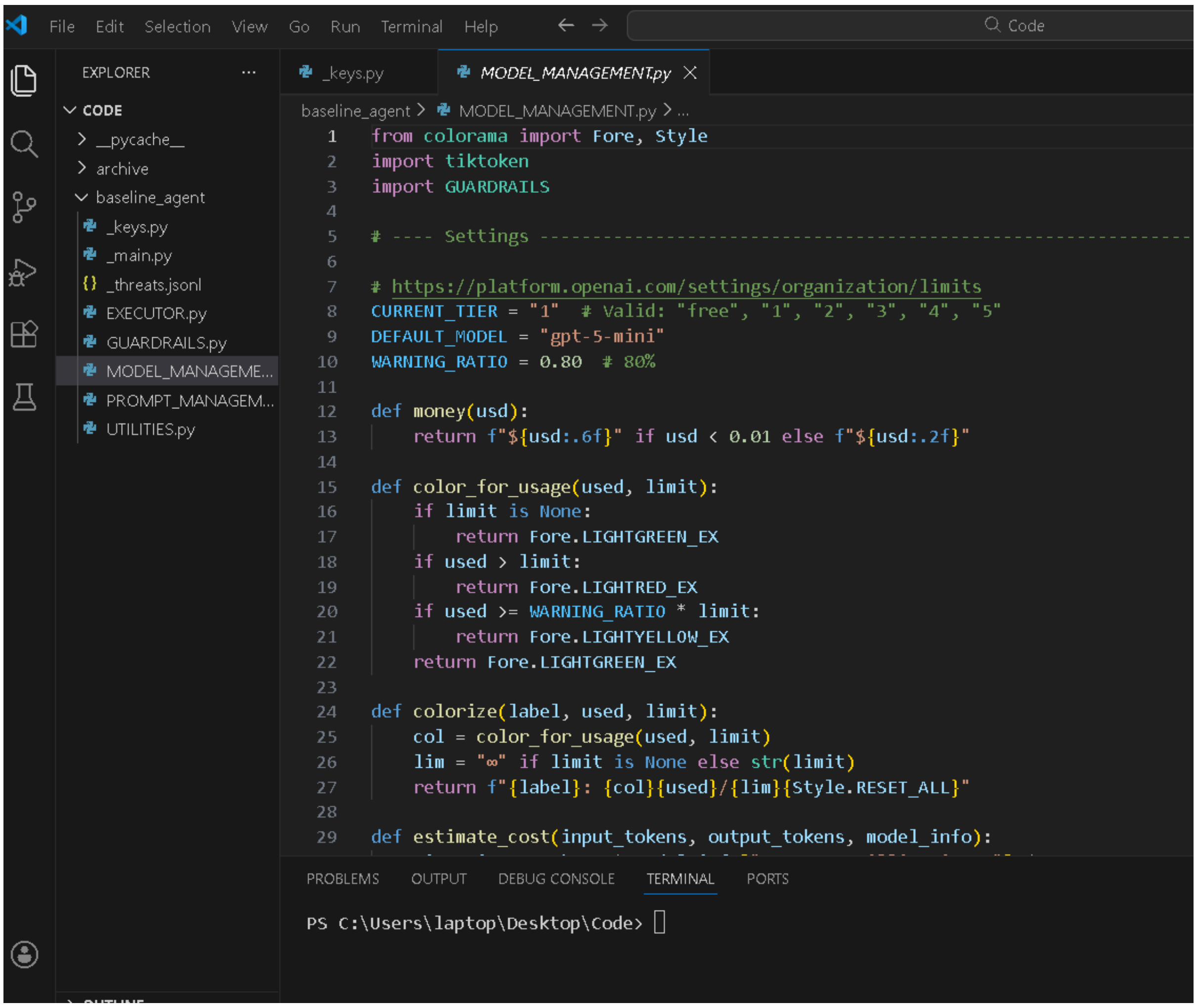


```
PS C:\Users\laptop\Desktop\Code> & C:/Users/laptop/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/laptop/Desktop/Code/tokens.py
Chat will cost approximately $0.0821
Proceed? (y/n):
```



```
PS C:\Users\laptop\Desktop\Code> & C:/Users/laptop/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/laptop/Desktop/Code/tokens.py
Chat will cost approximately $0.0421
Proceed? (y/n): y

Estimated input tokens:  25693
Estimated output tokens: 1000
Estimated chat cost:     $0.0421

Actual input tokens:     24547 [UNDER]
Actual output tokens:    1000 [UNDER]
Actual chat cost:        $0.0407 [UNDER]

fin.
PS C:\Users\laptop\Desktop\Code>
```

When token usage is maximized, the cost of a single investigation is approximately **$0.08**. To prevent unexpected or unnecessary spending, guardrails are implemented by explicitly defining a maximum token limit. In this case, a limit of **1,000 tokens** is later applied—an amount that proves sufficient for the intended use case while keeping costs consistently low.



```python
from colorama import Fore, Style
import tiktoken
import GUARDRAILS

# ---- Settings ---------------------------------------------------------------

# https://platform.openai.com/settings/organization/limits
CURRENT_TIER = "1"  # Valid: "free", "1", "2", "3", "4", "5"
DEFAULT_MODEL = "gpt-5-mini"
WARNING_RATIO = 0.80  # 80%

def money(usd):
    return f"${usd:.6f}" if usd < 0.01 else f"${usd:.2f}"

def color_for_usage(used, limit):
    if limit is None:
        return Fore.LIGHTGREEN_EX
    if used > limit:
        return Fore.LIGHTRED_EX
    if used >= WARNING_RATIO * limit:
        return Fore.LIGHTYELLOW_EX
    return Fore.LIGHTGREEN_EX

def colorize(label, used, limit):
    col = color_for_usage(used, limit)
    lim = "∞" if limit is None else str(limit)
    return f"{label}: {col}{used}/{lim}{Style.RESET_ALL}"

def estimate_cost(input_tokens, output_tokens, model_info):
```

```
PS C:\Users\laptop\Desktop\Code>
```

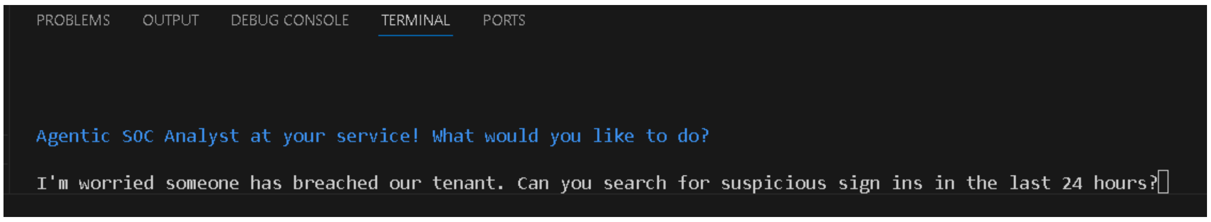
With cost controls in place, the AI agent functionality is fully enabled within the program.

Step 2. Agent Behaviour and Threat Identification

The AI is configured to operate under a defined **SOC Analyst role**, established through system-level instructions within the program. This ensures the agent's behaviour aligns with how a human analyst would approach an investigation—prioritizing context, relevance, and risk.

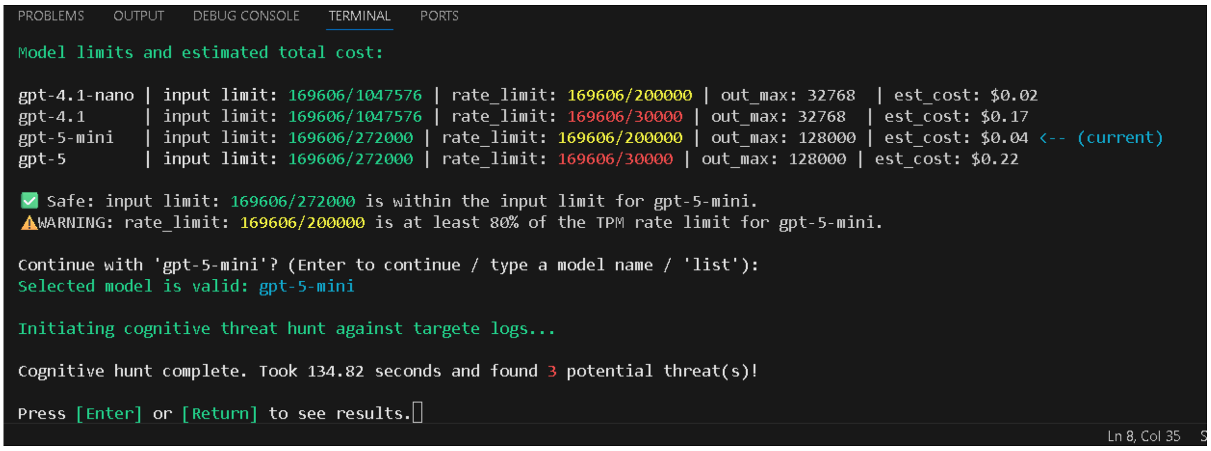
The agent connects to the Azure Log Analytics workspace through the Azure CLI, while OpenAI handles the natural language processing and reasoning behind each request. This separation allows the program to securely access enterprise telemetry while leveraging AI for interpretation and decision support.

To validate functionality, the following test prompt is issued:

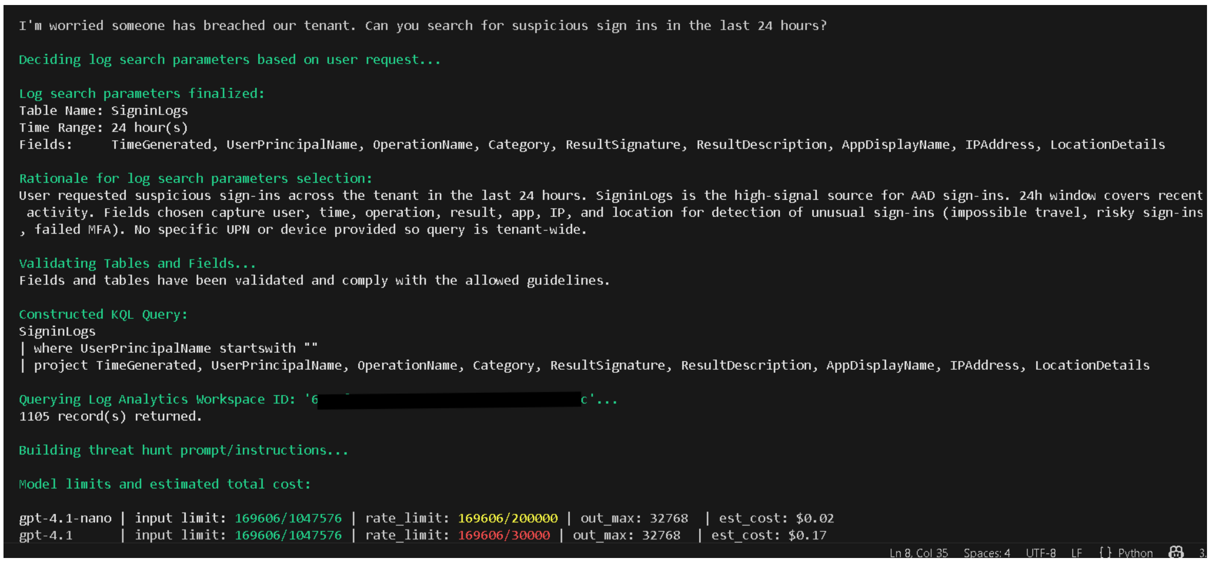


```
Agentic SOC Analyst at your service! What would you like to do?

I'm worried someone has breached our tenant. Can you search for suspicious sign ins in the last 24 hours?
```

"I'm worried someone has breached our tenant. Can you search for suspicious sign-ins in the last 24 hours?"

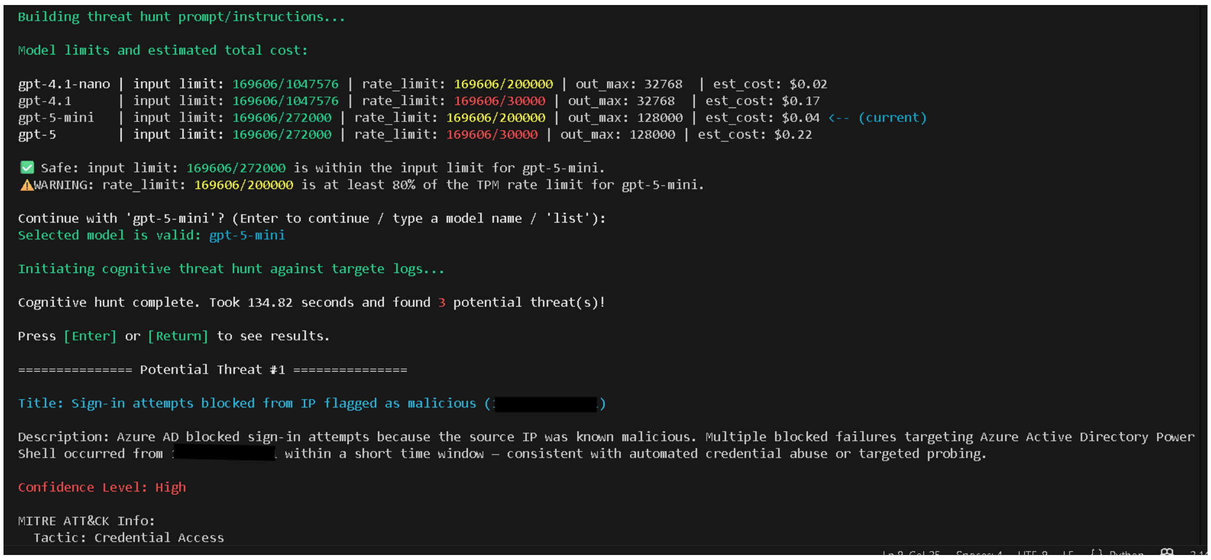


```
Model limits and estimated total cost:

gpt-4.1-nano | input limit: 169606/1047576 | rate_limit: 169606/200000 | out_max: 32768  | est_cost: $0.02
gpt-4.1      | input limit: 169606/1047576 | rate_limit: 169606/30000 | out_max: 32768  | est_cost: $0.17
gpt-5-mini   | input limit: 169606/272000 | rate_limit: 169606/200000 | out_max: 128000 | est_cost: $0.04 <-- (current)
gpt-5        | input limit: 169606/272000 | rate_limit: 169606/30000 | out_max: 128000 | est_cost: $0.22

✅ Safe: input limit: 169606/272000 is within the input limit for gpt-5-mini.
⚠️WARNING: rate_limit: 169606/200000 is at least 80% of the TPM rate limit for gpt-5-mini.

Continue with 'gpt-5-mini'? (Enter to continue / type a model name / 'list'):
Selected model is valid: gpt-5-mini

Initiating cognitive threat hunt against targete logs...

Cognitive hunt complete. Took 134.82 seconds and found 3 potential threat(s)!

Press [Enter] or [Return] to see results.
```

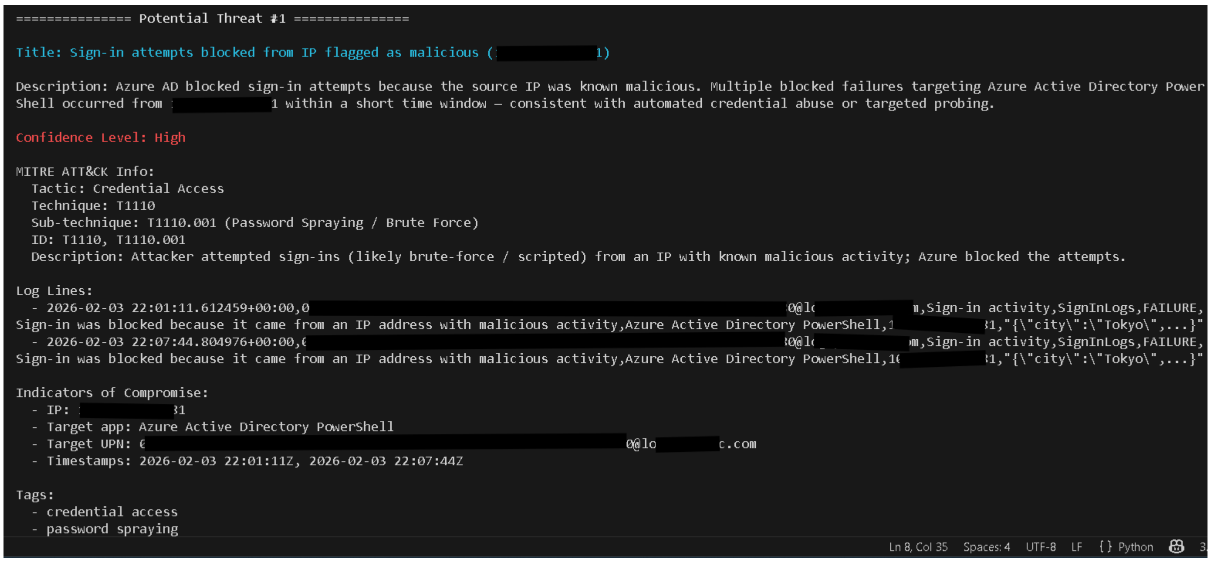


```
I'm worried someone has breached our tenant. Can you search for suspicious sign ins in the last 24 hours?

Deciding log search parameters based on user request...

Log search parameters finalized:
Table Name: SigninLogs
Time Range: 24 hour(s)
Fields:     TimeGenerated, UserPrincipalName, OperationName, Category, ResultSignature, ResultDescription, AppDisplayName, IPAddress, LocationDetails

Rationale for log search parameters selection:
User requested suspicious sign-ins across the tenant in the last 24 hours. SigninLogs is the high-signal source for AAD sign-ins. 24h window covers recent
 activity. Fields chosen capture user, time, operation, result, app, IP, and location for detection of unusual sign-ins (impossible travel, risky sign-ins
, failed MFA). No specific UPN or device provided so query is tenant-wide.

Validating Tables and Fields...
Fields and tables have been validated and comply with the allowed guidelines.

Constructed KQL Query:
SigninLogs
| where UserPrincipalName startswith ""
| project TimeGenerated, UserPrincipalName, OperationName, Category, ResultSignature, ResultDescription, AppDisplayName, IPAddress, LocationDetails

Querying Log Analytics Workspace ID: '6[redacted]c'...
1105 record(s) returned.

Building threat hunt prompt/instructions...

Model limits and estimated total cost:

gpt-4.1-nano | input limit: 169606/1047576 | rate_limit: 169606/200000 | out_max: 32768  | est_cost: $0.02
gpt-4.1      | input limit: 169606/1047576 | rate_limit: 169606/30000 | out_max: 32768  | est_cost: $0.17
```

From this prompt, the program automatically derives intent and generates an appropriate **KQL query** based on the requested investigation whilst also managing cost. Multiple query fields—such as time range, authentication context, and sign-in characteristics—are dynamically created by the AI.

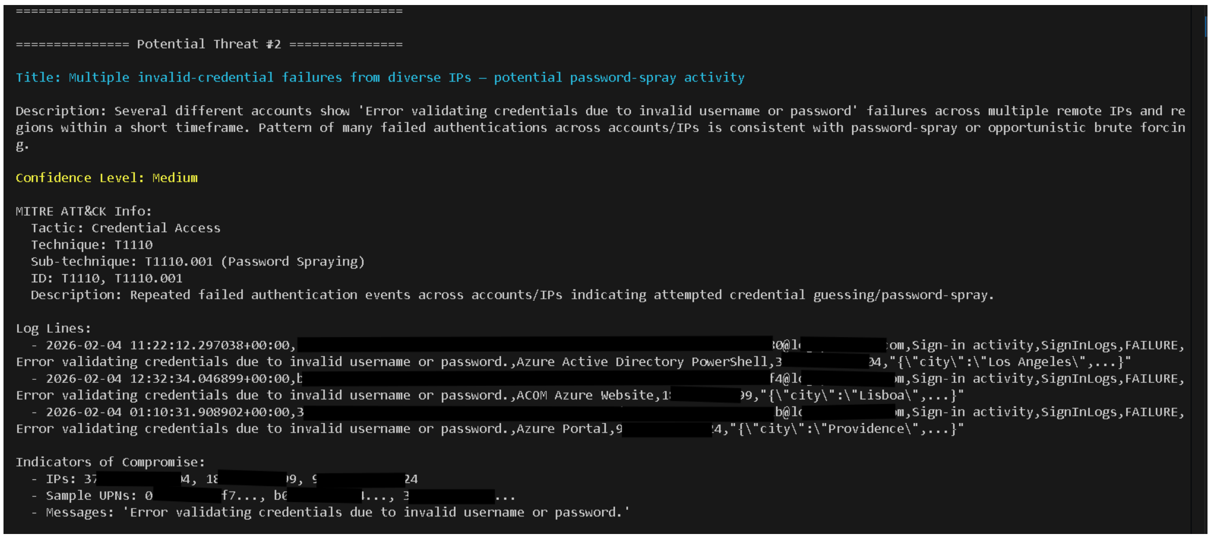


```
Building threat hunt prompt/instructions...

Model limits and estimated total cost:

gpt-4.1-nano | input limit: 169606/1047576 | rate_limit: 169606/200000 | out_max: 32768  | est_cost: $0.02
gpt-4.1      | input limit: 169606/1047576 | rate_limit: 169606/30000 | out_max: 32768  | est_cost: $0.17
gpt-5-mini   | input limit: 169606/272000 | rate_limit: 169606/200000 | out_max: 128000 | est_cost: $0.04 <-- (current)
gpt-5        | input limit: 169606/272000 | rate_limit: 169606/30000 | out_max: 128000 | est_cost: $0.22

✅ Safe: input limit: 169606/272000 is within the input limit for gpt-5-mini.
⚠️WARNING: rate_limit: 169606/200000 is at least 80% of the TPM rate limit for gpt-5-mini.

Continue with 'gpt-5-mini'? (Enter to continue / type a model name / 'list'):
Selected model is valid: gpt-5-mini

Initiating cognitive threat hunt against targete logs...

Cognitive hunt complete. Took 134.82 seconds and found 3 potential threat(s)!

Press [Enter] or [Return] to see results.

============== Potential Threat #1 ==============

Title: Sign-in attempts blocked from IP flagged as malicious ([redacted])

Description: Azure AD blocked sign-in attempts because the source IP was known malicious. Multiple blocked failures targeting Azure Active Directory Power
Shell occurred from [redacted] within a short time window — consistent with automated credential abuse or targeted probing.

Confidence Level: High

MITRE ATT&CK Info:
  Tactic: Credential Access
```



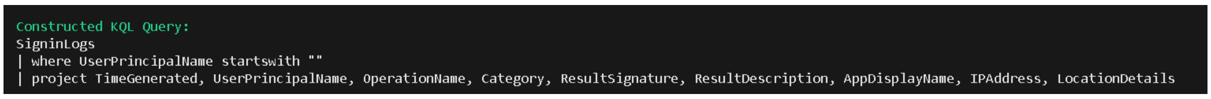
```
============== Potential Threat #1 ==============

Title: Sign-in attempts blocked from IP flagged as malicious ([redacted]1)

Description: Azure AD blocked sign-in attempts because the source IP was known malicious. Multiple blocked failures targeting Azure Active Directory Power
Shell occurred from [redacted]1 within a short time window — consistent with automated credential abuse or targeted probing.

Confidence Level: High

MITRE ATT&CK Info:
  Tactic: Credential Access
  Technique: T1110
  Sub-technique: T1110.001 (Password Spraying / Brute Force)
  ID: T1110, T1110.001
  Description: Attacker attempted sign-ins (likely brute-force / scripted) from an IP with known malicious activity; Azure blocked the attempts.

Log Lines:
  - 2026-02-03 22:01:11.612459+00:00,0[redacted]0@l[redacted]m,Sign-in activity,SignInLogs,FAILURE,
Sign-in was blocked because it came from an IP address with malicious activity,Azure Active Directory PowerShell,1[redacted]1,"{\"city\":\"Tokyo\",...}"
  - 2026-02-03 22:07:44.804976+00:00,[redacted]0@l[redacted]m,Sign-in activity,SignInLogs,FAILURE,
Sign-in was blocked because it came from an IP address with malicious activity,Azure Active Directory PowerShell,1[redacted]1,"{\"city\":\"Tokyo\",...}"

Indicators of Compromise:
  - IP: [redacted]1
  - Target app: Azure Active Directory PowerShell
  - Target UPN: [redacted]0@lo[redacted]c.com
  - Timestamps: 2026-02-03 22:01:11Z, 2026-02-03 22:07:44Z

Tags:
  - credential access
  - password spraying
```

Ln 8, Col 35   Spaces: 4   UTF-8   LF   { } Python

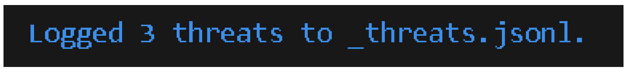


```
==================================================

============== Potential Threat #2 ==============

Title: Multiple invalid-credential failures from diverse IPs — potential password-spray activity

Description: Several different accounts show 'Error validating credentials due to invalid username or password' failures across multiple remote IPs and re
gions within a short timeframe. Pattern of many failed authentications across accounts/IPs is consistent with password-spray or opportunistic brute forcin
g.

Confidence Level: Medium

MITRE ATT&CK Info:
  Tactic: Credential Access
  Technique: T1110
  Sub-technique: T1110.001 (Password Spraying)
  ID: T1110, T1110.001
  Description: Repeated failed authentication events across accounts/IPs indicating attempted credential guessing/password-spray.

Log Lines:
  - 2026-02-04 11:22:12.297038+00:00,[redacted]0@l[redacted]om,Sign-in activity,SignInLogs,FAILURE,
Error validating credentials due to invalid username or password.,Azure Active Directory PowerShell,3[redacted]04,"{\"city\":\"Los Angeles\",...}"
  - 2026-02-04 12:32:34.046899+00:00,b[redacted]f4@l[redacted]m,Sign-in activity,SignInLogs,FAILURE,
Error validating credentials due to invalid username or password.,ACOM Azure Website,1[redacted]99,"{\"city\":\"Lisboa\",...}"
  - 2026-02-04 01:10:31.908902+00:00,3[redacted]b@l[redacted]m,Sign-in activity,SignInLogs,FAILURE,
Error validating credentials due to invalid username or password.,Azure Portal,9[redacted]4,"{\"city\":\"Providence\",...}"

Indicators of Compromise:
  - IPs: 37[redacted]4, 18[redacted]9, [redacted]24
  - Sample UPNs: 0[redacted]f7..., b0[redacted]..., 3[redacted]...
  - Messages: 'Error validating credentials due to invalid username or password.'
```

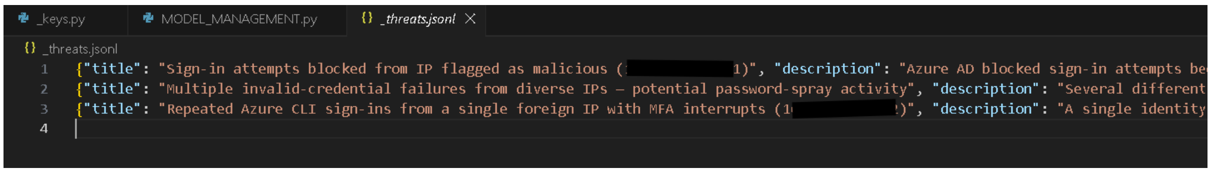
The agent then analyzes the data returned from the Log Analytics query, identifies potential threats, and records the findings in a structured **JSON file** for later review.



```
Constructed KQL Query:
SigninLogs
| where UserPrincipalName startswith ""
| project TimeGenerated, UserPrincipalName, OperationName, Category, ResultSignature, ResultDescription, AppDisplayName, IPAddress, LocationDetails
```



```
Logged 3 threats to _threats.jsonl.
```



```
{"title": "Sign-in attempts blocked from IP flagged as malicious ([redacted]1)", "description": "Azure AD blocked sign-in attempts be
{"title": "Multiple invalid-credential failures from diverse IPs – potential password-spray activity", "description": "Several different
{"title": "Repeated Azure CLI sign-ins from a single foreign IP with MFA interrupts (1[redacted])", "description": "A single identity
```

Each identified event is enriched with:

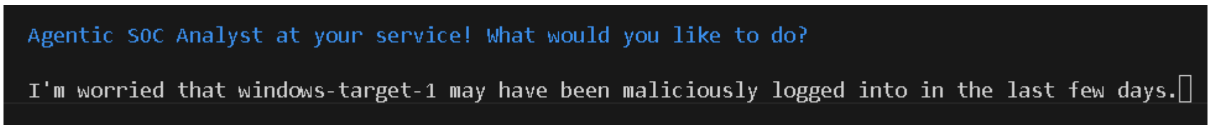
- A mapped attack category or technique
- Relevant investigative or response guidelines
- A confidence level to assist analyst judgment

This approach mirrors how SOC analysts' triage and assess alerts, enabling faster review while maintaining transparency and analyst oversight.

Step 3: Automated Threat Response

With the core investigation workflow in place, the final phase of the project introduces **automated threat response** capabilities through **Microsoft Defender for Endpoint (MDE)**. This enhancement transforms the agent from a passive analysis tool into an active participant in incident response.

To support this expanded capability, additional guardrails are implemented to ensure actions are deliberate, controlled, and auditable. These safeguards are designed to prevent overreaction, limit scope, and ensure that automated responses occur only when specific criteria are met as seen by the results.



```
Agentic SOC Analyst at your service! What would you like to do?

I'm worried that windows-target-1 may have been maliciously logged into in the last few days.
```

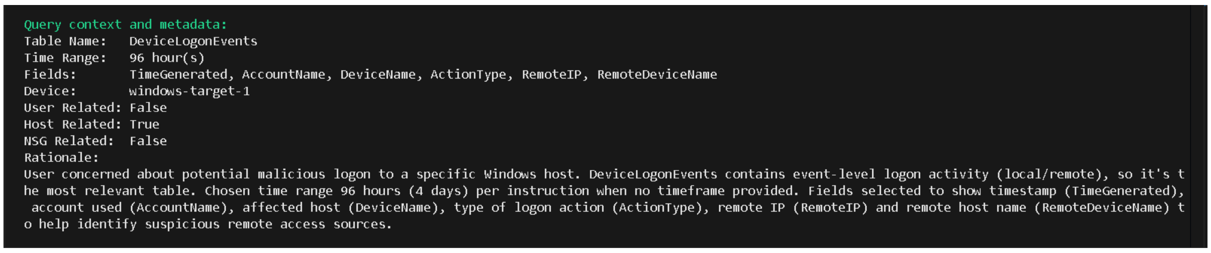
Step 4: Automated Investigation and Response Workflow

At this stage, the agent can take investigative findings and translate them directly into response actions.

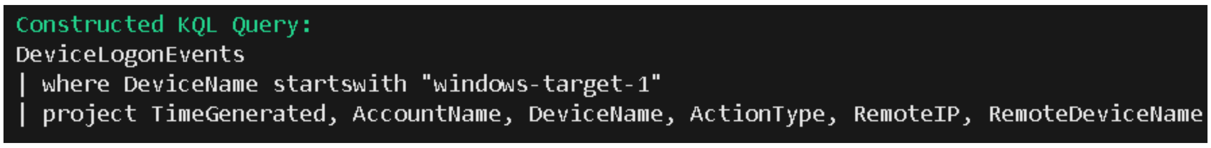
When the AI identifies suspicious or malicious activity, it now:

- Generates a **custom KQL query** based on the investigation context
- Searches historical and real-time records within the Log Analytics workspace
- Correlates affected users, sessions, and endpoints

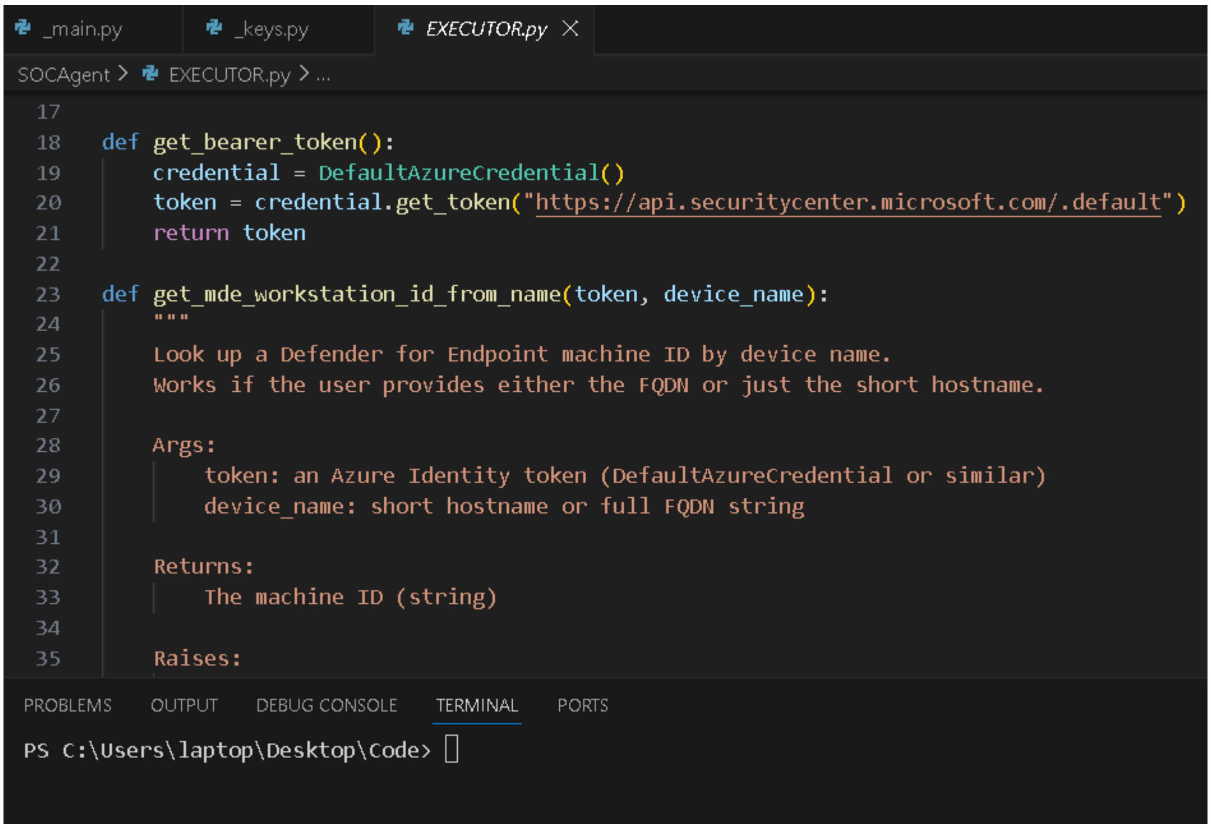
- Determines whether a response action is warranted



```
Query context and metadata:
Table Name:   DeviceLogonEvents
Time Range:   96 hour(s)
Fields:       TimeGenerated, AccountName, DeviceName, ActionType, RemoteIP, RemoteDeviceName
Device:       windows-target-1
User Related: False
Host Related: True
NSG Related:  False
Rationale:
User concerned about potential malicious logon to a specific Windows host. DeviceLogonEvents contains event-level logon activity (local/remote), so it's t
he most relevant table. Chosen time range 96 hours (4 days) per instruction when no timeframe provided. Fields selected to show timestamp (TimeGenerated),
 account used (AccountName), affected host (DeviceName), type of logon action (ActionType), remote IP (RemoteIP) and remote host name (RemoteDeviceName) t
o help identify suspicious remote access sources.
```

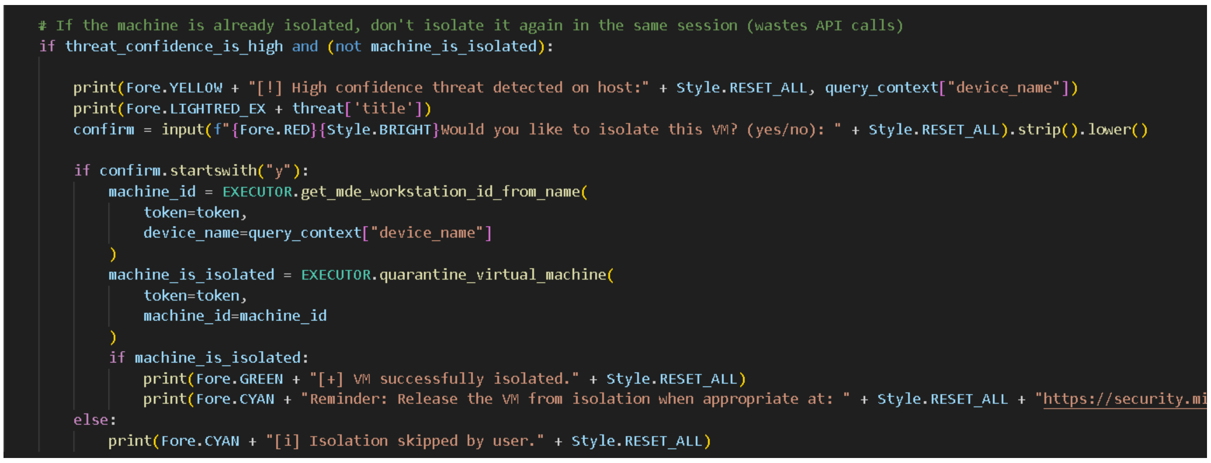


```
Constructed KQL Query:
DeviceLogonEvents
| where DeviceName startswith "windows-target-1"
| project TimeGenerated, AccountName, DeviceName, ActionType, RemoteIP, RemoteDeviceName
```



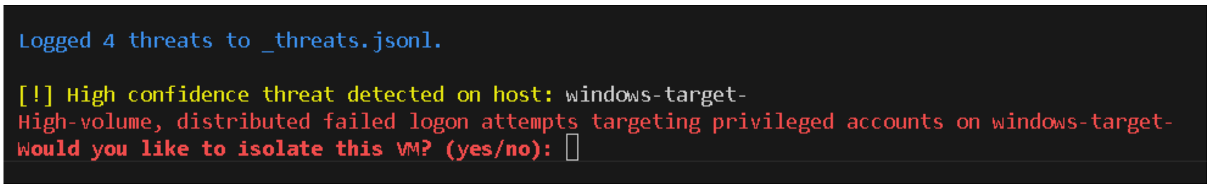
```python

def get_bearer_token():
    credential = DefaultAzureCredential()
    token = credential.get_token("https://api.securitycenter.microsoft.com/.default")
    return token

def get_mde_workstation_id_from_name(token, device_name):
    """
    Look up a Defender for Endpoint machine ID by device name.
    Works if the user provides either the FQDN or just the short hostname.

    Args:
        token: an Azure Identity token (DefaultAzureCredential or similar)
        device_name: short hostname or full FQDN string

    Returns:
        The machine ID (string)

    Raises:
```

```
PS C:\Users\laptop\Desktop\Code> 
```

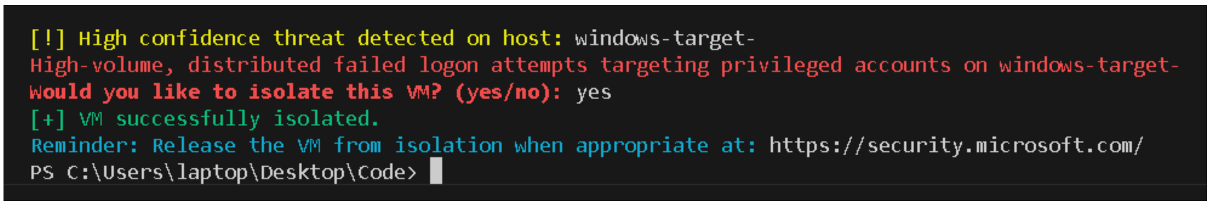


```python
# If the machine is already isolated, don't isolate it again in the same session (wastes API calls)
if threat_confidence_is_high and (not machine_is_isolated):

    print(Fore.YELLOW + "[!] High confidence threat detected on host:" + Style.RESET_ALL, query_context["device_name"])
    print(Fore.LIGHTRED_EX + threat['title'])
    confirm = input(f"{Fore.RED}{Style.BRIGHT}Would you like to isolate this VM? (yes/no): " + Style.RESET_ALL).strip().lower()

    if confirm.startswith("y"):
        machine_id = EXECUTOR.get_mde_workstation_id_from_name(
            token=token,
            device_name=query_context["device_name"]
        )
        machine_is_isolated = EXECUTOR.quarantine_virtual_machine(
            token=token,
            machine_id=machine_id
        )
        if machine_is_isolated:
            print(Fore.GREEN + "[+] VM successfully isolated." + Style.RESET_ALL)
            print(Fore.CYAN + "Reminder: Release the VM from isolation when appropriate at: " + Style.RESET_ALL + "https://security.mi
    else:
        print(Fore.CYAN + "[i] Isolation skipped by user." + Style.RESET_ALL)
```

If malicious activity is confirmed and falls within predefined thresholds, the agent initiates an automated response through Microsoft Defender for Endpoint. In this implementation, the primary response action is **isolating affected virtual machines**, effectively containing the threat while further analysis can continue.



```
Logged 4 threats to _threats.jsonl.

[!] High confidence threat detected on host: windows-target-
High-volume, distributed failed logon attempts targeting privileged accounts on windows-target-
Would you like to isolate this VM? (yes/no):
```

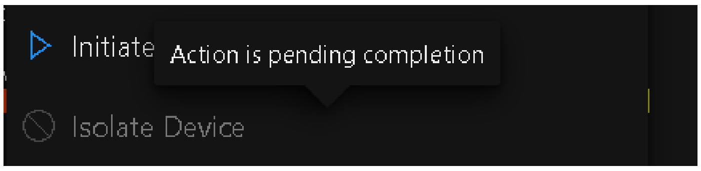


```
[!] High confidence threat detected on host: windows-target-
High-volume, distributed failed logon attempts targeting privileged accounts on windows-target-
Would you like to isolate this VM? (yes/no): yes
[+] VM successfully isolated.
Reminder: Release the VM from isolation when appropriate at: https://security.microsoft.com/
PS C:\Users\laptop\Desktop\Code>
```

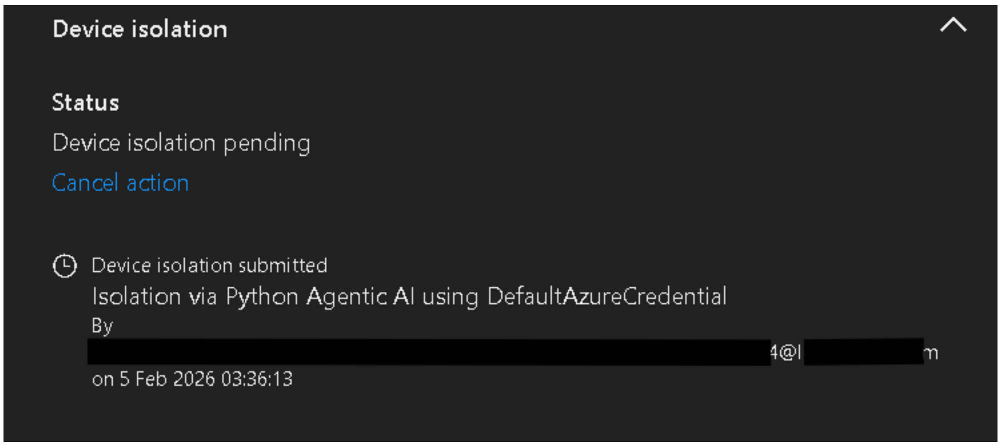
This mirrors real-world SOC escalation workflows, where containment is prioritized to limit impact while preserving evidence.

Step 5: Action Validation and Logging

Once a response action is issued, the agent monitors the status returned by MDE.



A "pending completion" status confirms that the action was successfully submitted and is in progress, providing immediate feedback that the containment step has been initiated.


Device isolation

Status

Device isolation pending

Cancel action

Device isolation submitted

Isolation via Python Agentic AI using DefaultAzureCredential

By

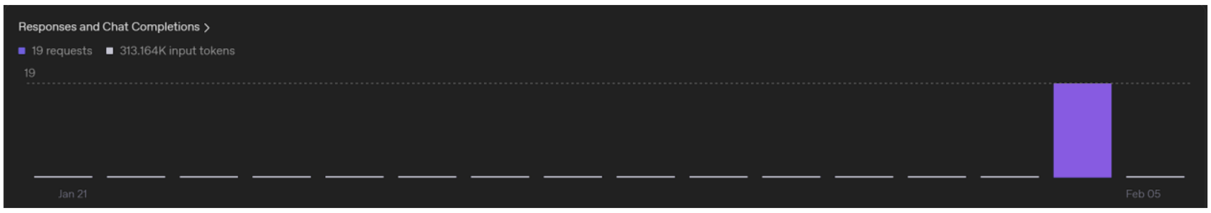
[redacted]4@l[redacted]m

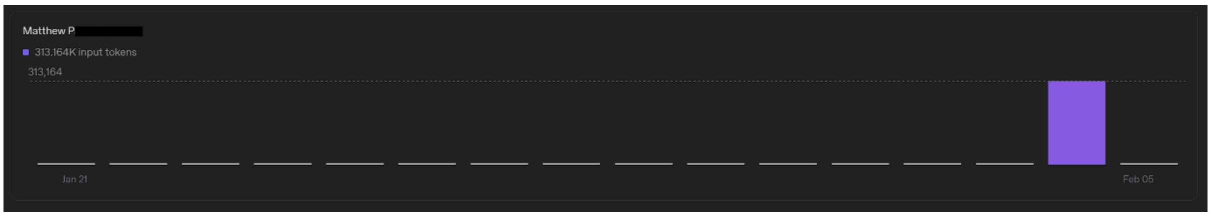
on 5 Feb 2026 03:36:13

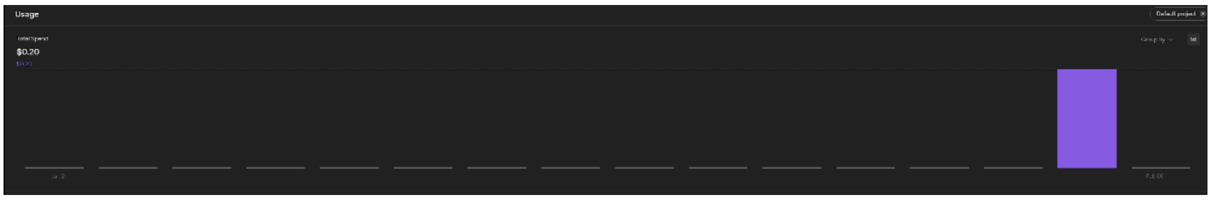
All actions taken by the agent—including the triggering indicators, investigative context, and response outcome—are logged for transparency and review. This ensures that every automated decision can be traced, audited, and validated by an analyst.

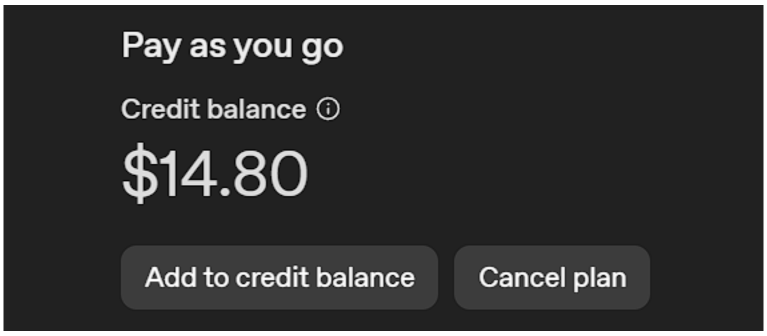
Step 6: Cost Optimization and Operational Efficiency

Throughout development, cost efficiency was treated as a core design requirement rather than an afterthought. By implementing strict token limits, selecting cost-effective models, and validating usage before execution, the average cost of an investigation was reduced to **well below one cent per run**.








Pay as you go

Credit balance ⓘ

$14.80

These controls demonstrate how AI-driven security workflows can remain financially viable at scale, even when performing repeated or automated investigations. By combining guardrails with modular model selection, the agent balances analytical depth with predictable, minimal operating costs—making it suitable for continuous SOC usage rather than isolated experimentation.

Summary

By the conclusion of this project, the AI agent is capable of:

- Interpreting natural language security concerns
- Dynamically generating KQL queries
- Analyzing enterprise security telemetry
- Identifying and classifying potential threats
- Applying cost controls and operational guardrails
- Executing automated containment actions through MDE
- Logging findings and responses for analyst review

This project demonstrates how AI agents can be responsibly integrated into SOC workflows—not as a replacement for analysts, but as a **force multiplier** for SOC Teams that accelerates investigations, reduces manual effort, and enables faster, more consistent response to security threats.

From initial detection to automated containment, the agent showcases a practical and scalable approach to modern threat hunting and remediation.