Shaghayegh Moradirad

Cloud Native AI + Inference Day · KubeCon North America 2026

November 9, 2026 · RAG and AI agents track

An Agent Is Not a Function Call

```
result = agent.run(task)
# input in, output out
```

What the interface to an agent runtime should expose

Code shown: github.com/openai/codex · sdk/python at commit cb1bc0a (Nov 1, 2026)



Shaghayegh Moradirad

Applied Physics @ Stanford
Ex-MTS @ OpenAI

Worked on the Codex Python SDK; scientific computing and open source.

Coding agents break the function-call shape.

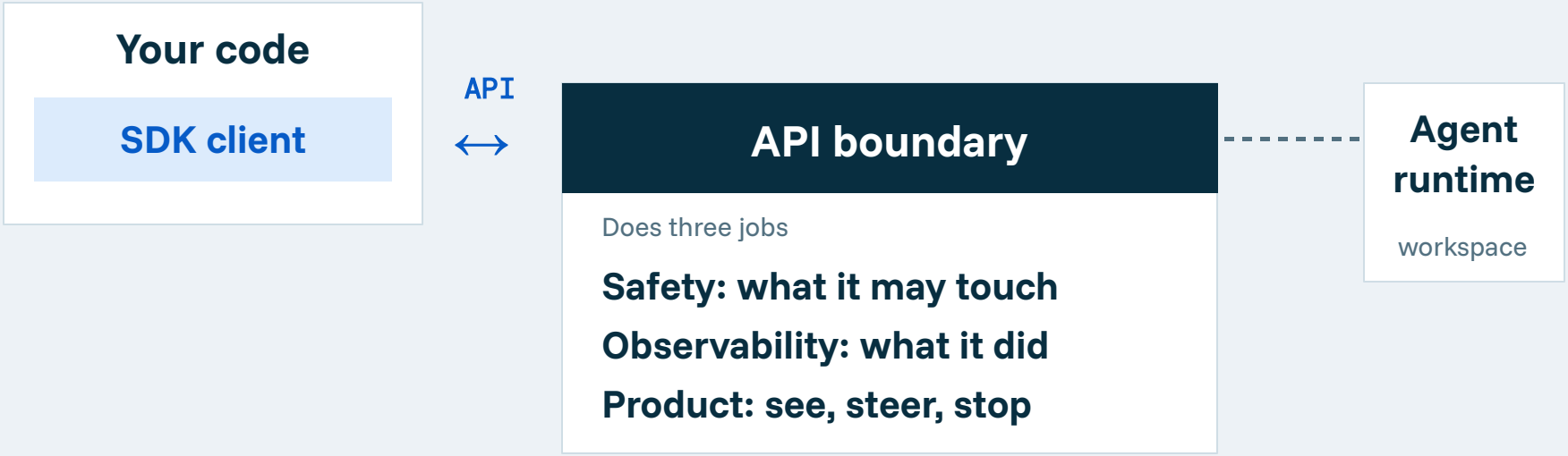
A function call shows you two moments: the call and the return.

BETWEEN THE CALL AND THE RETURN	
THE CALL	`result = agent.run("Fix the failing test.")`
WHAT A CODING AGENT DOES MEANWHILE	none of it visible to the caller
Operates over time, often for minutes	
Keeps state across turns	
Streams intermediate work	
Acts on a workspace: reads files, runs commands, edits code	
Consumes a budget	
Sometimes needs to be steered or interrupted	

The caller sees only the return value.

Once a model can act, the API boundary does three jobs at once.

Treating agents as function calls hides all three.



Five pressures, five primitives

01	Context outlives the process →	**Threads**
02	Actions take time →	**Turns**
03	Actions are invisible unless reported →	**Streams**
04	Actions have consequences →	**Sandbox policies**
05	The answer is only part of the outcome →	**Result objects**

Start from one fact: the model can act, and acting takes time. · sdk/python @ cb1bc0a

Persistent context you can resume or fork.

Pressure: the agent's context is its turn history. In process memory, it dies with the process.

PYTHON · OPENAI_CODEX

```python
with Codex() as codex:
    thread = codex.thread_start()
    thread.run("Fix the failing test.")

    later = codex.thread_resume(thread.id)
    branch = codex.thread_fork(thread.id)
    history = later.read(include_turns=True)
```

State has an ID, and it outlives your client process.

A turn is a unit of work you can steer or interrupt.

Pressure: a turn runs for minutes, so the call can’t block. You need a handle, and a handle needs verbs.

TASK	Upgrade the logging library.
YOUR CODE	WHAT HAPPENS
1 `turn = thread.turn(task)`	You hold a handle
2 `turn.steer("Keep the API stable.")`	Same turn, new input
3 `turn.interrupt()`	Ends as interrupted
4 `thread.run("Smaller change.")`	Same thread continues

A function call has no verbs between the call and the return.

The stream shows the work while it happens.

Pressure: effects happen before anything returns, so each action must be emitted as it happens.

YOUR CODE

TASK Fix the failing test.

PYTHON · TURN STREAM

```
turn = thread.turn(task)
for event in turn.stream():
    if event.method == "item/completed":
        item = event.payload.item.root

        audit_log.append(item)
```

ENDS AT turn/completed for this turn ID

items

COMMAND ITEM

command
cwd
exit_code
duration_ms
aggregated_output
status

status can be completed, failed, or **declined**

Events are routed by turn ID, so one client can watch several turns.

Workspace access is typed state, not prompt text.

Pressure: a limit in the prompt is a wish; a parameter the runtime enforces is a control. Also typed: approval_mode, ExternalMessage.

PYTHON · OPENAI_CODEX	IN EFFECT
THREAD START `codex.thread_start(sandbox= Sandbox.workspace_write)`	workspace-write
TURN `thread.run("Fix the failing test.")`	workspace-write
TURN OVERRIDE `thread.run("Review the diff.", sandbox= Sandbox.read_only)`	read-only
NEXT TURN, NO ARGUMENT `thread.run("Write a summary.")`	still read-only

A turn's sandbox carries over. Show users that state.

The final answer is one field out of nine.

Pressure: a run has status, cost, duration and side effects. Returning only the text throws them away.

Function call

RETURNS

VALUE	final_response

TurnResult

NINE FIELDS

id	
status	
error	
started_at	
completed_at	
duration_ms	
final_response	can be None
items	every completed item
usage	turn and thread, cached incl.

If a turn fails, run() raises RuntimeError; stream() still delivers the failed turn and its error.

Retry only overload errors:

```
retry_on_overload()
```

TAKEAWAYS

Ask any agent runtime three questions.

01 What was it allowed to do?

Look for: sandbox and approval as typed state; inputs that carry authority.

02 What did it actually do?

Look for: a stream per unit of work; records of every action, refusals included.

03 How can a person or a system intervene?

Look for: steer and interrupt mid-run; resume and fork later; safe retries only.

If every answer is “wait for the return value,” it’s a function call.

Thank you

Shaghayegh Moradirad

github.com/openai/codex/tree/main/sdk/python

An Agent Is Not a Function Call

SESSION FEEDBACK