AGENTIC AI

WEEK 2

OPENAI AGENTS SDK

A Deep Dive into Asynchronous Programming, Multi-Agent Frameworks, and Practical Applications

Agenda

01

Asynchronous Python & AsyncIO

Understanding the foundation for concurrent execution.

02

OpenAI Agents SDK: Core Concepts

Exploring agents, handoffs, and guardrails.

03

Building Your First Agent

Practical steps to create and run an agent.

04

Advanced Agentic Workflows

Tools, multi-agent collaboration, and multi-model integration.

05

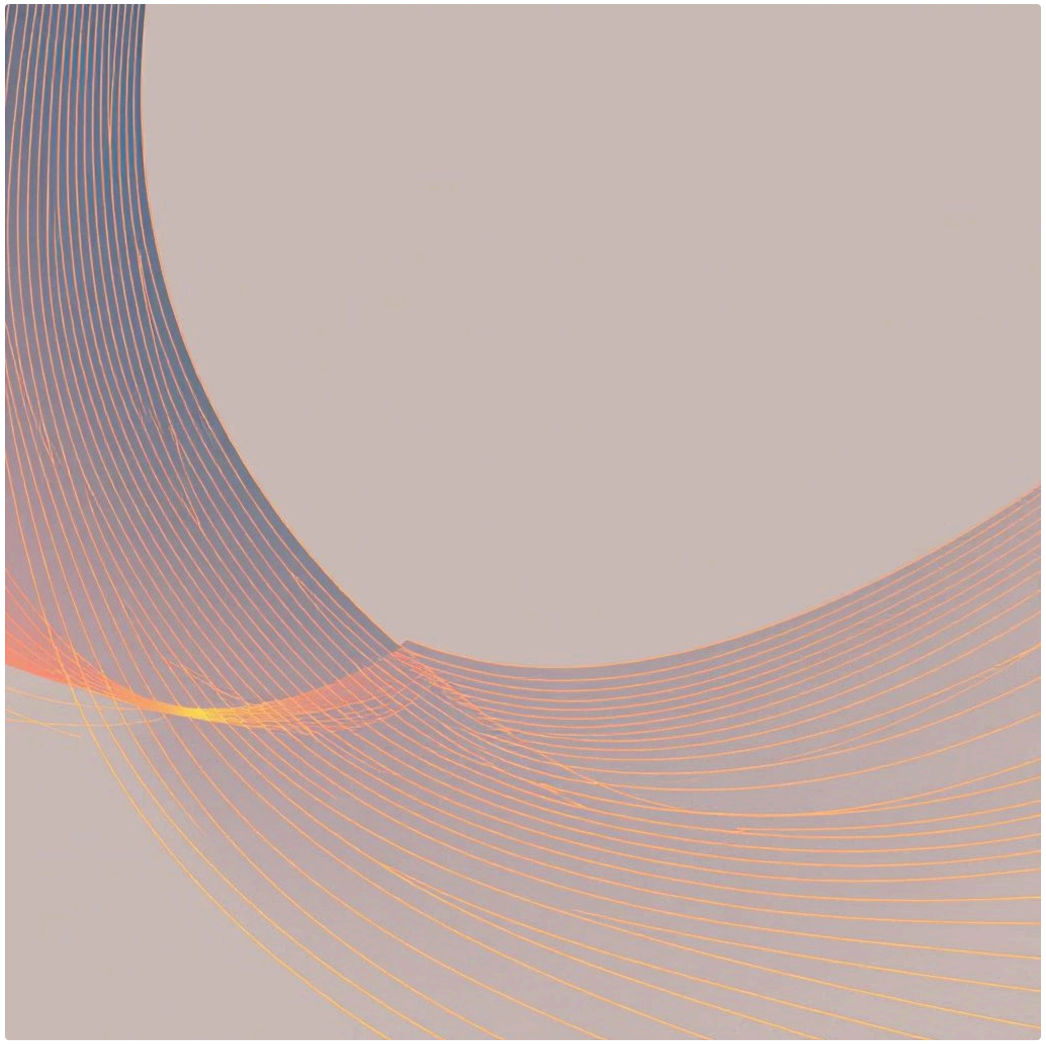
Real-World Applications

Sales development and deep research projects.

Asynchronous Python: The Power of AsyncIO

AsyncIO, introduced in Python 3.5, enables lightweight multithreading by allowing concurrent execution of tasks without involving operating system-level threads. Unlike multiprocessing, where Python processes run independently, AsyncIO allows all Python processes to "play together."

In multi-agent frameworks, especially when multiple models are hitting different APIs, AsyncIO is implemented using **async** and **await** keywords.



1

Event Loop

The core of AsyncIO, responsible for scheduling and executing coroutines and handling I/O events. It manages the flow of control between different coroutines.

2

Async

Used to define a coroutine, a special function that can be paused and resumed. Calling an async function returns a coroutine object, which must be scheduled on the event loop to run.

3

Await

Used inside an async function to pause its execution until an "awaitable" object (like another coroutine or task) completes. This yields control back to the event loop, allowing other tasks to run.

OpenAI Agents SDK: Core Terminologies

The OpenAI Agents SDK is designed to be lightweight, flexible, and out-of-the-way, making common activities easy. It provides a structured approach to building AI-powered applications.

Agents

Represent individual AI entities capable of performing tasks.

Handoffs

Represent interactions and transfers of control between different agents.

Guardrails

Represent controls and checks to ensure agents operate within defined boundaries and safety parameters.

To run an agent, there are three key steps: create an agent, use a runner to orchestrate its work, and then run the agent.

Building Your First Agent: A "Jokester"

Creating your first agent with the OpenAI Agents SDK is straightforward. You define an agent with a name, instructions, and a model. The instructions act as the system prompt for the agent.

The SDK is not limited to OpenAI models; it can work with many other models, offering flexibility in your AI integrations.

The runner orchestrates the agent's execution, and the result can be streamed or awaited for the final output.

```
Aptions/amiil{
Jokester: 'Joytter ()
    sctredderlalorr:
    freatlokes inttell josst'');
    reccute mekestller {
    xeccute putior:
    freatloke: Mike joste">
    recoute fe: take (oatter:fosct))
    — cutll(peturr;
    freatjoke: joktler";
    freatjoke: joter woser'>
    freatioke: antlter {
      gutile: joster'>
      gutiwe: tentiling jokes: forst')
      putcalle: puttiler">
      jotest: rentatall josst">
      potake: tentring jokes(Maectler>
      percalwe Joslte:
      puttulle: antiáll josst">
      porcalke tatfalle">
      gutive: tantter_Dutter(Maectller:
      eerculse nettiler'>
      poclwe: sollow.jelarestller:
```

Code Snippet Example:

```
from dotenv import load_dotenv
from agents import Agent, Runner, ra

load_dotenv(override=True)

agent = Agent(name="Jokester", instruction="You are a joke teller", model="gpt-4")

async def main():
 result = await Runner.run(agent, "Tell a joke about a cat.")
 print(result.final_output)

if __name__ == "__main__":
 ra(main())
```

Advanced Agentic Workflows: Tools vs. Handoffs

As you build more complex agentic projects, understanding the distinction between tools and handoffs becomes crucial. Both allow agents to collaborate, but they differ in control flow.

Tools

When an agent uses a tool, it converts a function into a new tool with a JSON schema describing its capabilities. Calling the tool executes the agent, which then makes the call to the underlying models (LLMs). Control typically returns to the main agent after the tool's execution.

Example: Using SendGrid for sending transactional emails from their server.

Handoffs

Handoffs involve passing control to another agent. The execution continues with the new agent, and control can be passed back to the original agent or another agent in the workflow. This is useful for sequential or branching processes.

Example: An agent calling on other specialized agents to complete a task.

Multi-Model Integration & Guardrails

The OpenAI Agents SDK supports integration with models other than OpenAI's, such as Gemini or Groq, offering flexibility in model selection.

Desired Output

Agents can be configured to not just respond with text, but with a specific desired output format or structure. This ensures the output is directly usable for subsequent steps or applications.

Guardrails

Guardrails are crucial for controlling agent behavior and ensuring safety. They can be applied at the very beginning of the input or at the output of the last agent in a sequence.

Note: Guardrails can only be applied to OpenAI models within the SDK's current implementation.

Project 2: Building a Sales Development Representative (SDR) Agent

This project demonstrates building a sophisticated SDR agent using three different layers of agentic architectures.

1 **Workflow of Agentic Architectures**

Designing the overall flow and interaction between agents.

2 **Agent Using a Tool**

Implementing an agent that can leverage external tools, like an email sending service.

3 **Agent Calling Other Agents**

Creating an agent that can orchestrate and delegate tasks to other specialized agents.

This project highlights the practical application of agentic AI in automating and enhancing sales processes, from lead generation to initial outreach.

Project 3: Deep Research Agent

This project focuses on a deep research agent, showcasing a powerful use case of agentic AI: an agent capable of searching the internet.

This is a well-known application, as many proprietary large language models (LLMs) already incorporate internet search capabilities. The project explores how to implement this functionality within the Agents SDK.

Key aspects include:

- Using tools for internet access.
- Handling structured outputs from search results.
- Managing complex workflows for research tasks.

The goal is to enable the agent to conduct thorough investigations, summarize findings, and potentially generate new insights based on its research.

Key Takeaways & Next Steps

AsyncIO is Foundational

Enables efficient concurrent execution for multi-agent systems.

Agents SDK Simplifies AI

Provides a flexible framework for building complex AI workflows.

Tools & Handoffs for Collaboration

Empower agents to interact with external systems and each other.

Guardrails Ensure Control

Critical for safe and reliable agent operation.

Explore the OpenAI Agents SDK and start building your own intelligent agents today!