# Openai-agents SDK:

**openai-agents SDK** aik powerful Python package hai jo tumhein **AI Agents** banane mein help karta hai — **especially Chainlit, LiteLLM, aur OpenAI models ke sath**.

## ✅ Simple Definition (Roman Urdu):

openai-agents aik Python library hai jo tumhein AI agents banane, unhein run karne, aur unse conversation karwane ka system deti hai.

- Sochna sikhaata hai (Agent)
- Baat karna sikhaata hai (Runner)
- Zubaan (model) set karta hai (OpenAIChatCompletionsModel)
- Aur OpenAI ya Gemini se contact karne deta hai (AsyncOpenAI)

# LLM vs Agent — Simple Logic:

## LLM (Large Language Model):

**Dimagh (Brain)** ki tarah hota hai. Jaise: GPT, Gemini, Claude. **Sirf jawab deta hai**, lekin **action** khud nahi leta.

## Agent kya karta hai?:

**Agent ek** action lene wala system **hota hai jo LLM ka use karta hai sochne ke liye, aur** khud kaam karta hai.

```
LLM = Dimagh (Sirf sochta hai, jawab deta hai)
Agent = Amal karne wala insan (LLM ka use karke action leta hai)

LLM + Agent = Ek complete smart AI jo kaam bhi karta hai, sirf bolta nahi!
```

# UV:

uv ek **fast Python package manager + environment tool** hai — jo tumhare agent project ko build, install aur run karne mein help karta hai.

1st:

➞ Ye tumhare **agent project ka start point** hota hai.

```
uv init
```

## 🎁 Real Life Example (Kitchen Style 🧿):

**Tum ek nayi recipe (agent) banana chahti ho.**
Uske liye pehle kya karti ho?

✅ Ek **naya kitchen shelf** setup karti ho:

- Shelf bana leti ho
- Ingredients rakhne ki jagah set karti ho
- Tools (knife, pan, spoon) rakh leti ho

## 💻 Programming mein:

uv init --package agent_project

✅ Ye **ek naya project ka shelf** banata hai:

- src/ → Tumhara code yahan hoga
- .venv/ → Tumhara Python environment
- pyproject.toml → Tumhare project ki recipe likhi hogi

2nd:

```
uv venv
```

Ye command ek **virtual environment (venv)** banata hai,
jisme sirf tumhare project ke liye tools (packages) install honge —
baaki system Python ya doosre projects par koi asar nahi hoga.

3rd :

```
uv run main.py
```

Tumhare virtual environment ko activate karta hai
 Uske andar jo packages (chainlit, openai-agents, etc.) hain unka use karta hai
 Aur tumhari main.py wali file run karta hai

apni likhi hui recipe (code) ko apne kitchen (venv) mein chalao"

4th :

```
uv add openai-agents
```

Apne project (kitchen) mein openai-agents naam ka ek naya masala (tool/package) add karo."OpenAI, Gemini, Claude jaise LLMs ko connect karne mein
 Multi-agent workflows banane mein
 Chainlit ke saath connect karne mein

5th :

```
Import os
```

✅ **import os**
Ye Python ka **built-in module** hai jo **Operating System (OS)** ke sath kaam karta hai.

💡 **Kya karta hai?**
Tum .env file se **secret keys** ya **API keys** uthana chahti ho, to os.getenv() ka use karte ho.

🗣️ **Asaan lafzon mein**:
os ka kaam hai tumhari **hidden locker** (jaise .env file) se **important cheezein** nikalna 🗝️

6th :

```
import os
from dotenv import load_dotenv
from agents import Agent, Runner, OpenAIChatCompletionsModel

load_dotenv()
```

✅ **from dotenv import load_dotenv**

dotenv ek chhota sa **Python package** hai jo .env file ko load karta hai.
💡 .env file wo file hoti hai jisme tum **secret data** rakhti ho,

🗣️ **Asaan lafzon mein**:
load_dotenv() ka kaam hai .env file ko **khulne layak** banana, taake os usme se cheezein read kar sake.

Agar tum `load_dotenv()` na likho, to `.env` file se `api_key` kabhi bhi read nahi ho payega.

1. os = Operating System se hidden cheezein nikalne wala module hai
2. dotenv = .env file load karne wala helper hai
3. load_dotenv() = .env ko active karta hai taake os use kar sake
4. Api_key = os.getenv("OPENAI_API_KEY")

```
api_key = "GEMINI_API_KEY"
if not api_key:
    raise RuntimeError("KEY NOT FOUND")
```

## Agent Loop kya hota hai?

"Ek aisa cycle jahan ek agent user se input leta hai, us input ko LLM (Large Language Model) ko bhejta hai, aur phir LLM ke jawab ke basis pe action leta hai."

```
model = OpenAIChatCompletionsModel(
    model="gpt-4.1-nano-2025-04-14",  # https://platform.openai.com/docs/models/gpt-4.1
    openai_client=api_key
)
```

Agent loop kia huta hy???

Jab bhi strt ho ga 2 cheain lay ga 1 agent lay ga dusra user input lay ga agr hm ye dono chezain ly gey tu wo req send kry  dy gal lm ko jaisy hi hmri req llm ko jye gi wo dekhy ga agr koi normal swal howa tu wo khod jwb dy ga agr koi difficult ho gat u wo phir sy agent ky pa aye ga or agent koi tool use kry ga o sky bad final output nzar aajye ga

```
Agent = Agent(
   name="assistant",
   instructions="you are a helpful assistant"
)

result = Runner.run_sync(agent, "Who is the founder of Pakistan?")
print(result.final_output)
```

`Agent = Agent(...)`

Yeh ek **agent object** bana raha hai, jiska naam `"assistant"` rakha gaya hai.

- **name="assistant"** → Agent ka naam rakha gaya hai.
- **instructions="you are a helpful assistant"** → Agent ko yeh guide kiya gaya hai ke uska kaam madad karna hai.

 *Ye agent AI model ke sath kaam karega aur user ke sawaalon ka jawab dega.*

`Runner.run_sync(...)` → Yeh agent ko **sync mode** mein chala raha hai (yaani bina `async` ke seedha chala raha hai).

## `final_output` kya hota hai?

**Yeh koi aapka variable nahi hai** — yeh ek **built-in property** hai us `result` object ki jo `Runner.run_sync()` return karta hai.

- ⬆️ `final_output` mein **LLM se aaya hua final answer** hota hai.

**Sochiye:** Agent ne sawal bheja: `"Who is the founder of Pakistan?"`
LLM (e.g., Gemini, GPT) ne jawab diya: `"The founder of Pakistan is Muhammad Ali Jinnah."`
To yeh jawab `result.final_output` mein hota hai.

- `result` ➤ variable ka naam hai
- `Runner.run_sync(...)` ➤ **function hai jo ek object return karta hai**
- Is liye `result` ke andar jo cheez hai, **wo object hota hai**

**`result` aik variable hai, jo aik object ko hold karta hai.** Aur us object ke andar `.final_output` ek attribute hai jisme final answer hota hai jo agent ne diya hota hai.

Ab hm gemi sy bnaye gey :

```
external_client = AsyncOpenAI(
    api_key = api_key,
    base_url = "https://generativelanguage.googleapis.com/v1beta/openai/"
)
```

`AsyncOpenAI(...)`

Yeh ek **class/function** hai jo Google Gemini API se connect hone ka **client** banata hai

### 🩹 Real Life Example:

Socho aap Pizza order karna chahti ho.

- `AsyncOpenAI(...)` = pizza company ka phone number milana
- `api_key` = aapka name/password
- `base_url` = jis branch ko aap call kar rahi ho

Aur:

- `external_client` = woh phone line jisme aap ab call kar sakti ho

```
model = OpenAIChatCompletionsModel(
    model="gemini-2.0-flash",
    openai_client=external_client
)
```

```
Agent = Agent(
    name = "assistant",
    instruction = "you are a helpful assistant",
    model = model
)

result = Runner.run_sync(
    Agent,
    "tell me about piaic.",
    run_config = config
)

print(result.final_output)
```

### 1️⃣ Agent = Agent(...)

Yeh ek **AI agent** bana raha hai — socho jaise ek **robot** jise:

- Naam diya gaya: `"assistant"`
- Kaam bataya gaya: `"you are a helpful assistant"`
- Dimaag diya gaya: `model`

📌 Socho yeh **ek insan banaya gaya hai**:

- `name`: uska naam (assistant)
- `instruction`: kya kaam karega
- `model`: kis AI model se baat karega (Gemini ya GPT)

- Gemini = **Sochne wala AI (LLM)**
- Agents SDK = **Us sochne wale AI ko chalane wala system (logic)**

---