✅ SDK kya hota hai:*

SDK = Software Development Kit
Ye ek **toolbox** hota hai jisme ready-made **functions, classes, aur tools** hote hain jo programmer ki coding **asan** bana dete hain.

```
!pip install -uq openai-agents
```

✅ Ye line ka matlab hai: **"Chup chaap latest Agent SDK install kar lo."**

Ye extra cheezain jo installion mian nzr ati hy os ko hide krta hy (uq)

📄 Example: OpenAI SDK mein kya kya hota hai?

```
openai.ChatCompletion     # ChatGPT se baat karne ka tareeqa

openai.Image.create       # DALL·E se image banwana

openai.Audio.transcribe   # Audio ko text mein badalna

openai.File.upload        # File ko upload karna
```

Ek Line Difference:

"OpenAI SDK sirf GPT se baat karne ke liye hota hai – jese chatbot. Lekin Agent SDK GPT ko aik real assistant banata hai – jo tools use karta hai, sochta hai, aur kaam karta hai."

🎁 Short Point Notes (Class ya Presentation ke liye):

- 🔹 **OpenAI SDK** = GPT se chat karna (text in → text out)
- 🔹 **Agent SDK** = GPT + tools + logic → kaam bhi karwana
- 🔹 GPT sirf text model hai — agent **task solver** ban jata hai
- 🔹 Agent SDK real-time tool use kar sakta hai (e.g., weather check, API call)

🔹 **OpenAI SDK = Bada toolbox**
🔹 **Agents SDK = Toolbox ka ek naya intelligent section**

Ek Line Difference:

"OpenAI SDK se tum GPT-4 se sirf baat karti ho, Agent SDK se tum GPT-4 ko kaam karwati ho."

🔹 "Agent" ka matlab kya hota hai?

Ek aisa AI ka program jo khud se sochta hai, decide karta hai, aur kaam karta hai.

Example:

- Tum ek agent bnao jo **user ka weather poochhne par** khud **web se data le aye**.
- Ya ek agent bnao jo **calculator tool** use kar k answer nikaalay.

Next.asynio 1 python library hai jo allow krti hy next.asyncevent loop chanlny ky liye.

“next.asyncio.apply() ek async method hai jo AI agent ko chalaata hai aur await karta hai jab tak agent se proper jawab na mil jaye.”

Agentloop ky ander ham 2 chezain pas karty hein 1 name 2 instruction. Ye 2 chezain jab ham agent ko dain gey tu agent apni req llm ko day ga. Llm har kam kr skta hy bas os ko smjhna parta hai. Llm ky jany kay pas 1 condtion huti hy jaise agr jo is ky pas req aye hy wo osky mutabiq hy tu wo os ka jwb day ga tu yahan py khtm ho gya hmra agent loop jis ko hm khty hen finaloutput.

Or agr aisa swal aya jo ky llm ka na howa tu wo tool call kray ga phir hmra llm boly ga kay tool call kro Tool aik **extra power** hoti hai jo action perfom krwny main madad deti hy.tool ka req peramtr dety howy hm tool call kry gey.

🛠 Tool kya hota hai AI agent ke liye?

Tool aik **extra power** hoti hai jo agent use kar sakta hai kaam complete karne ke liye. 1huta hy callable or 1 huta hy awaitable

Jaise:

- calculator chalana
- browser open karke search karna
- Python code run karna
- kisi external API ko call karna
- ya tumhara custom function use karna

Agent bina tool ke **sirf text reply** karta hai, lekin tool milne ke baad **kaam bhi kar sakta hai**.

Type	Naam	Kya hota hai?
1️⃣	**Hosted Tools**	Ye OpenAI ne khud diye hain – built-in tools jaise **code interpreter**, **browser**, **DALL·E**, **Python runner**, **file search**
2️⃣	**Function Tools (Custom Tools)**	Tum khud Python function bna kar agent ko doge, aur agent use karega
3️⃣	**Agent-as-a-Tool**	Tum ek agent ko doosre agent ka tool bana do – ek agent dusre se kaam le sakta hai

1. **learn-agentic-ai**
2. /01_ai_agents_first
3. /15_run_lifecycle (pic)

output_type

Agent ka answer kis structure mein aaye (JSON ya object form)

```
class Person(BaseModel):
    name: str
    age: int

agent = Assistant(
    name="PakAgent",
    instructions="Answer with name and age of famous people",
    output_type=Person
)

# Tum pucho:
"Who is the founder of Pakistan?"

# Agent pehle reply karega:
Thought: "I will find who founded Pakistan..."

# Phir output dega:
{
  "name": "Muhammad Ali Jinnah",
  "age": 71}
```

Yeh object hoga PersonInfo class ke structure ke mutabiq — jise hum kehte hain **structured output** ya **tool-use compatible output**.

"Agent jab structured output type milta hai, to pehle reasoning karta hai aur phir JSON/object format mein jawab deta hai jise hum output_type kehte hain – yeh response code ke liye easy hota hai."

from agents.run import RunConfig

agent hmry pas 1 building hy agent ky ander hmry pas posion waise classes rkhi hoi hy ab main agent ko ghr sy bulaon ya posion sy at 1 hy

✅ 1. Agent aik data class hoti hai:

Jab hum kehte hain **“agent ek data class hai”**, iska matlab hai ke:

- Uske andar **fixed attributes (yaani properties)** hote hain
- Wo class **Python mein banayi gayi hoti hai @dataclass ya pydantic.BaseModel** ki tarah

🔹 Example:

```
from openai import Assistant

agent = Assistant(
   name="MonaBot",
   instructions="Help with questions",
   tools=[...],
   max_turns=10,
   output_type=MySchema
)
```

- name, instructions, tools, max_turns, output_type
  ☞ ye **sab attributes hain**.

💠 max_turns Kya Hoti Hai?

"Agent ko maximum kitni baar sochnay + jawab denay ka moka diya jaye."
Yani agent ko **kitne "steps"** milenge apna kaam complete karne ke liye.

🌀 Ek "turn" kya hoti hai?

Ek **turn** = agent ka aik reasoning + tool use + jawab dena.Example:

- Tum ne agent se pucha: "Find who is the founder of Pakistan"
- Agent sochta hai → tool use karta hai → jawab deta hai

☞ Ye **1 turn** complete hui.

Lekin kaam abhi bhi complete na ho
To agent **stop ho jata hai** aur error ya incomplete jawab deta hai.

Sir ne jo kaha – Wo sahi kyu hai?

- Default max_turns=10 hoti hain.
- Agar tumhara agent complex kaam kar raha hai (multi-step tools, function calls, chaining), to use zyada turns chahiye hoti hain.
- Tab tum max_turns=20 ya 30 manually set kar sakti ho.
- **"Max turns batata hai ke agent ko apna kaam complete karne ke liye kitni baar reasoning ya tool use karne ki ijazat hai. Default 10 hoti hai, lekin agar kaam complex ho to hum manually zyada set kar sakte hain."**

✅ 2. Result Till ya Continuous Running – iska matlab kya hai?

Jab tak **final result** na mil jaye (ya koi break condition na ho), tab tak agent chalta rahega.

"Koi na koi condition lagi honi chahiye warna agent kabhi rukega nahi."

🌀 run() jab agent ko chalaata hai to wo loop mein chalta hai

```
from openai import run

result = await run(agent, input="Find weather and explain it.")
```

To internally run():

- agent ko repeatedly **think → act → respond** karwata hai
- jab tak koi **condition complete na ho jaye**
- ya agent max_turns tak na pohanch jaye
- ya agent finish_reason = stop na kar de

🗣️ Viva Line:

"Agent ek data class hota hai jisme name, instructions jaise attributes hote hain. Jab agent run hota hai, to wo turn-by-turn sochta hai, jab tak koi result mil na jaye ya koi condition usay na roke – jese max_turns ya finish_reason."

🛡️ Guardrails kya hoti hain?

Guardrails khud 1 agent huta hy. Jo hmry agent ky upper beht jta hy koi bhi input araha hy ye os ko dkehta hy. Guardrail ka matlab hota hai "suraksha lineen" – yani agent ke kaam karne ke tareeqe par control rakhna.

Agent ko **har cheez allowed nahi hoti** – kuch limits lagani padti hain:

✅ Input valid ho ya na ho
✅ Output expected format mein ho ya nahi
✅ Agent kuch galat to nahi kar raha

Ye sab **Guardrails** ka kaam hai.

🔹 1. **Input Guardrail** – Asaan Samjho

Agar koi user ne agent ko invalid input diya — jaise:

🗣 "aslkfjalkjsdf!!!@@@"
🗣 "Hacking attempt"
🗣 "NULL"

To **input guardrail** check karega:

❌ "Kya ye input sahi format mein hai?"
❌ "Kya is input mein kuch dangerous to nahi hai?"

Agar nahi hai to wo input ko **reject ya sanitize** karega.

📄 Example:

```
from openai import guardrail
```

```
@guardrail.input
def validate_input(message: str):
    if not message.endswith("?"):
        raise ValueError("Sawal question mark se khatam hona chahiye.")
```

🔹 2. **Output Guardrail** :

Agar agent ne kuch galat jawab de diya — ya output type match nahi kiya:

{"name": "Jinnah", "age": "seventy one"} ← galat (age number hona chahiye)

To **output guardrail** check karega:

❌ "Kya agent ka output tumhare schema ya expected format ke mutabiq hai?"

Agar nahi, to wo:

- ❌ Error dega
- 🔁 Ya agent ko dubara try karne ko bolega

Sir ne jo kaha – Bilkul Sahi Tha:

"Guardrail agent ke upar hoti hai – agent khud kuch bhi kare, guardrail usko control mein rakhti hai."

"Agent ka input sahi hai ya nahi – input guardrail check karta hai"
"Agent ka jawab theek hai ya nahi – output guardrail check karta hai"

Ek Line mein Definition (Viva Line):

"Guardrails agent ke input aur output par constraints lagate hain – input guardrail ensure karta hai ke user ka data sahi ho, aur output guardrail ensure karta hai ke agent ka jawab format ya rules ke mutabi

🔊 Viva Line:

"Instructions agent ke liye hoti hain, jabke handoff_description dusre system ko batata hai ke is agent ka kaam kya hai."

1. **RunResultBase** kya hoti hai?

Is ky ander hm apny main kam rkhty hein.jaise agent loop ho gya tool call ho gya handoff ho gya ye hmry pas parent class huti hhy.

Ye aik **base class** hoti hai jisme agent ke run hone ke baad jo cheezein milti hain, wo sab define hoti hain.

Yani jab tum agent ko run karti ho:

```
result = await run(agent, input="Who is Jinnah?")
```

To `result` actually ek object hota hai **RunResultBase** ya uske child class ka — jisme sab kuch hota hai:

ya `RunResultBase` ki baat kar rahi ho — jo ek **parent class** (ya base class) hoti hai OpenAI Agents SDK mein, jisme agent run ka **output structure** define hota hai.

📄 Example Socho:

```
result = await run(my_agent, input="Find age of Jinnah")
```

Ab `result` ek object hai jisme:

result.agent	✅ ← konsa agent tha (Kis agent ne kaam kiya)
result.input	✅ ← user ka sawal(Tumne agent ko kya poocha)
result.output	✅ ← agent ka final jawab (structured)(Agent ka final structured jawab)
result.messages	✅ ← agent ne har step pe kya bola
result.handoff	✅ ← kya agent ne kaam kisi aur ko diya?
result.status	✅ ← run successful ya failed?
thoughts	Agent ka reasoning flow
tool_calls	Agent ne kaunse tool use kiye

Viva Line ya Short Summary:

"RunResultBase aik base class hoti hai jo agent ke run hone ke baad ka sara data rakhti hai — jaise agent kaun tha, uska output kya tha, tool use huay ya handoff hua, etc."