Understanding OpenAI Agent SDK:

What is an Agent?

An Agent is like a smart helper (A Mini AI) that can think, decide and take actions based on your instructions.

Example:

Imagine you say to a smart assistant:

"Book a flight, send an email, and tell me the weather."

An **Agent** will plan each step:

1. Find flights.
2. Choose the best one.
3. Write and send the email.
4. Get the weather from the internet.

You don't have to tell it **how** to do each step — it does that itself. That's what makes it an agent — it acts **on your behalf**.

What is OpenAI Agent SDK?

The **OpenAI Agent SDK** is a toolkit made by OpenAI that helps you **build your own AI agents** easily.

With it, you can:

. Create agents that can **read files**, **browse the web**, **use tools**, or even **talk to other apps**.

. Give your agent memory, so it can **remember things**.

. Let your agent take actions step by step using AI models like GPT.

✅ ***Example:***

Let's say you're building a smart assistant that helps with your homework.
Using OpenAI Agent SDK, you can make it:

- Read your question
- Search the internet
- Write the answer
- Save it in a file

You don't have to write every step manually — the agent plans and does it.

Why is the Agent class defined as a dataclass?

What is a `dataclass`?

*A `dataclass` in Python is a shortcut that helps you **create classes quickly** when your class is mostly just **storing data** (like names, goals, or tools).Instead of writing long code, `@dataclass` makes it neat and automatic.*

Definition of Agent class:

*An **Agent class** is a structure that holds important information about an AI agent, like its **name**, **goal**, **tools it can use**, and sometimes its **memory**. It doesn't always do actions directly but stores the data needed to help the agent perform tasks.*

Why use `@dataclass` for Agent?

Here are the main reasons:

1. **Less Code**: You don't have to manually write the `__init__()` method to set values.
2. **Easy to Read**: The class looks clean and simple.
3. **Better Printing**: You can print the agent object and see all its values easily.
4. **Easy to Compare**: You can compare two agents directly using `==`.
5. **Best for Storing Info**: Since the Agent class mainly stores data, `dataclass` is a perfect fit.

Why is the Agent class defined as a `dataclass`?

*The `Agent` class is defined as a **dataclass** because it is only used to **store information** like the agent's name, goal, tools, and memory.*
*Using `@dataclass` makes the code **clean, short, and easy** to use.*
*It **automatically creates** useful things like the constructor (`__init__`) so we don't have to write extra code.*

Example:

Without @dataclass:

```
class Agent:
def ___init___(self, name, goal):
self.name = name
self.goal = goal

agent1 = Agent("HelperBot", "Answer questions")
print(agent1.name) # Output: HelperBot
```

With @dataclass:

```
from dataclasses import dataclass

@dataclass
class Agent:
```

name: str
goal: str

agent1 = Agent(“HelperBot”, “Answer questions”)
print(agent1)
# Output: Agent(name=’HelperBot’, goal=’Answer questions’)

What is the user prompt? Why is it passed in the `run()` method of Runner?

What is a user prompt?

A user prompt is the input or question that the user gives to the agent.
It tells the agent what the user wants — like a message saying: “Please do this for me.”

Example:
If you type:

```
"Suggest a recipe using eggs and tomatoes"
```

That is the user prompt — the agent will read it and give an answer.

Why is it passed in the `run()` method?

Because the `run()` method is what starts the agent and tells it:
“Okay, here’s the user’s question — now go and answer it.”

So, the user prompt is passed to `run()` like this:

runner.run(user_prompt=”Tell me a joke”)

What is the purpose of the `Runner` class?

The `Runner` class is like a controller or manager.
It is used to:

1. Start the agent
2. Handle the flow (prompt in →agent thinks →response out)
3. Manage memory, tools, or context if needed.

Example:
Think of the agent like a car
The Runner is the driver who starts the car, steers it, and controls where it goes.

runner = Runner(agent=your_agent)
runner.run(“What’s the weather today?”)

What are Generics in Python? Why do we use `TContext`?

What are Generics?

Generics in Python let you write flexible code that works with any type of data.
Instead of fixing a type (like `int`, `str`, etc.), you write a placeholder.

Example:
Think of a box ☐ that can hold anything — toys, books, clothes.
That box is generic — it doesn't care what's inside.

Why do we use `TContext`?

`TContext` is just a custom name for a generic type. It represents context information passed between steps of the agent (like memory, user info, preferences, etc.).

Using `TContext` makes the code flexible — you can define what the context looks like later.

Simple Words:
`TContext` is like an empty box ☐ where we'll put things like user name, preferences, or history — but we'll decide *what* later.
It keeps the code clean and reusable.

Written by Duaa Pirzada