1.) The Agent class has been defined as a dataclass why?

In Python, a **dataclass** is a shortcut for creating a class that mainly holds **data** (like settings or options). It's a **shortcut** to writing classes that are primarily used to **store data**.

dataclass automatically adds:

- __init__() method
- __repr__() for printing
- __eq__() for comparing
- And more (like __hash__, etc., optionally)

```
from dataclasses import dataclass
@dataclass
class Person:
    name: str
    age: int
```

Agent is a class that represents an **AI assistant** (like ChatGPT) that can:

- Use tools
- Generate replies
- Delegate tasks to sub-agents

- Handle input/output logic
- Be customized with hooks and behaviors

The Agent class is used to store things like:

- instructions (how the agent should behave),
- tools it can use,
- and other settings.

Why is the Agent class defined as a @dataclass?

Imagine you're making a robot (an **agent**) and you want to give it:

- Some instructions (what it should do),
- Some tools (like calculator, browser, etc.),
- And other info.

You need a way to **store all this information** in one place — like a **box**.

🔷 A **class** in Python is like a box you can use to keep related things together.

🔷 A **dataclass** is a special kind of box that helps you **create and organize** those things **automatically** — without writing a lot of code.

✅ **Why use a dataclass for Agent?**

Because the Agent just **holds information** (like instructions

and tools), and a dataclass helps do that in a clean and simple way.

🟦 **Benefits of using @dataclass here:**

Feature	Benefit for Agent class
Auto-generated __init__	No need to manually write constructor for all fields
Auto __repr__	Easier debugging/logging
Field defaults	Cleaner code using default values or factories
Type hints + immutability	Safer structure, better with editors and mypy

So rather than manually writing an initializer for 10+ fields, @dataclass does it automatically and cleanly.
