Class 02: *

Class (AGENT) – Dimagh (Brain) of AI:

📌 Definition:

Agent aik aisa logical unit hai jo LLM (like GPT-4, Gemini etc.) ke sath interact karta hai. Ye decide karta hai kya karna hai, kaise karna hai, aur kis tool se karna hai.

Jese aik insaan ka "dimagh" hota hai — jo decide karta hai ke kisi sawal ka jawab khud de ya kisi tool (calculator, search engine, etc.) ka use kare — waise hi Agent bhi ye kaam karta hai.

✅ Real-Life Example:

Tumhare sawal ka jawab agent tumhari taraf se Gemini ko deta hai, aur uska reply tumhe la kar deta hai — jaise ek **messenger** ya **assistant**.

Class (Runner) – Driver ya Engine:

📌 Definition:

Runner aik helper class hai jo Agent ko chalata hai. Ye LLM ko input bhejta hai aur response wapas lata hai.

Samjho:

Jese Agent driver hai, Runner uska car engine hai. Agent sochta hai, Runner uski baat LLM tak pohcha kar answer wapas lata hai.

Decorator (@dataclass) – Yeh Kya Hai?

@dataclass Python ka aik **decorator** hai jo manually __init__ function likhne ki zarurat nahi hoti — wo khud ban jata hai jab tum @dataclass lagate ho. Function ka behaver change krta hy bina code ko change kiye

@dataclass
 	
class Person:
 	
name: str
 	
age: int
	class Person:
	
	def __init__(self, name, age):
	
	self.name = name
	
	self.age = age

✅ Field = Class ka Variable:

Jab tum @dataclass use karte ho to jitne bhi variables likhte ho (e.g. name, age) — unhe **fields** kehte hain.

💡 Generic Kya Hota Hai? – Most Important

✅ Simple Definition:

Generic" ka matlab hota hai **general-purpose** ya **har type ke data pe kaam karne wali cheez**.

Example:

Tumhara **shopping bag** generic hota hai:

- Kabhi usme **fruit** daalti ho
- Kabhi **books**
- Kabhi **bottles**

Ek hi bag, lekin har cheez carry kar sakta hai = **Generic**

Generic se hum aisi class ya function banate hain jo **multiple types ke data** ke sath kaam kare.

Generic is liye use karte ho taake class ya function **flexible** ban jaye — woh kisi bhi type ke object ke sath kaam kar sake.

Viva Tip:

@dataclass aik decorator hai jo class ko smart bana deta hai. Jab isko use karte hain, to **init** method khud ban jata hai. Fields us class ke variables hote hain. Generic use karte hain jab hum flexible code likhna chahte hain — jo kisi bhi data type ke liye kaam kare."

```
from typing import TypeVar, Generic

T = TypeVar("T")  # Placeholder for any type

class Box(Generic[T]):
    def __init__(self, item: T):
        self.item = item
```

🔶 2) Runner Class (Runner.run()) – Agent Ko Chalanay Wali Machine 🏃‍♂️

Runner ek **helper engine** hai jo tumhare Agent ko run karta hai — yaani user input le ke, agent ko call karta hai, aur LLM (like Gemini) se **final output** la ke deta hai.

Runner:

- Agent ko chalata hai
- LLM ko query bhejta hai aur jawab laata hai

Runner.run Asynchronous Function – Kya Karta Hai?

await Runner.run(

Runer.run asyncones kam krta hy. Asyncorne bolta hy wait kroon mian sra kam kr ky aap ko jb wb dun ga os sy phlly nhi

📌 Samjho:

- Ye **asynchronous** hai: await lagana **zaroori** hota hai.
- Jab tak LLM (e.g., Gemini) ka complete response nahi milta — ye **ruk jata hai**.
- Jab answer mil jaye, wo result variable me **object form** me return karta hai.

🎓 Viva:

"Runner.run() ek async function hai jo agent ko input deta hai, aur jab LLM se full response mil jata hai to wo result object return karta hai. Is object se hum string answer, steps, tool use, aur input sab dekh sakte hain."

Method	Type	Kaam Kya Karta Hai
Runner.run()	async	Asynchronous call, full LLM reply deta hai
Runner.run_sync()	sync	Synchronous version — await nahi lagana padta
Runner.run_streamed()	async	Streaming reply deta hai — har part slowly milta hai

✅ Output:

- Default: **string** milta hai (result.final_output)
- Lekin agar config mein output_type="object" do to tum full detailed object bhi paa sakti ho.

2. Runner.run_sync() → Jab tum async nahi use karna chahti

Yeh simple function hai Same Runner.run() ki tarah hi kaam karta hai Lekin ye synchronous hai. bina await lagaye tum normal tarike se run kar sakti ho.
Aise projects ke liye best hai jo basic Python script hain (like main.py).

🔷 3. Runner.run_streamed() → Jab tum streaming form mein answer chaho:

Yeh method tumhein **response line by line** ya **chunk by chunk** de sakti hai — jese tum Chainlit mein "typing..." effect dekhte ho.

Ye bhi async hai. Lekin LLM ka response **slowly line by line** stream karta hai.

- Real-time feeling deta hai (jaise ChatGPT slowly likhta hai).

Runner.run(input) ————▶ Wait karega... ▶ Full response in object form

Runner.run_sync() ————▶ Same, but sync ▶ Used in normal scripts

Runner.run_streamed()———▶ Stream karta hai ▶ Real-time response (step-by-step)

✅ Ye ek Module Import Statement hai.

- agents → module (yaani Python ka code file ya package)
- Agent → us module ke andar ek class

✅ "Module" kya hota hai?

Python ka **.py file ya folder jisme code ho**, usay module kehte hain.
Ye tumhare project ko **organized** banata hai.

✅ Modules ke Faide:

- Reusability (bar bar use kar sakti ho)
- Clean structure
- Code reuse across files

Concept	Meaning
Agent	LLM ke sath baat karne wala assistant
Runner	Agent ko chalanay ka engine
Module	Python file ya package jisme code hota hai
Decorator	Function ko enhance karne wala @ symbol
