Class 03:

instructions →

Yeh agent ko batata hai ke iska kaam kya hai.

"Tum kya ho aur tumhara role kya hai?"

Handoff_description →

Yeh batata hai ke yeh agent kis condition mein dusre agent ko handover kiya ja sakta hai.

"Agar mujhe kaam na mile, to mujhe kis tarah ka kaam dia ja sakta hai?"

Agar tum `triage_agent` use kar rahi ho, to wo `handoff_description` dekh kar decide karta hai kis agent ko forward kare. ✅

Lekin tum **handoff_description hi nahi likhti**, to **triage_agent ko ye samajh hi nahi aayega** ke English waale user ko kahan bhejna hai.

◆ Pehle `if __name__ == "__main__"` kya karta hai?

(jab ham apni file ko async banty hein tu o sky ander function ko run krny ky liye hm asyncio ko import krwty hen jo bydefult python huta hy or is ko run hm asyncio.run sy krwty hen)

◆ `asyncio.run(main())` ka matlab kya hai?

Python mein agar koi function `async def` se bana ho, to use **direct run nahi kar sakte**.
Usay **run karne ke liye event loop chahiye hota hai**.

`asyncio.run(main())`:

- Ek **event loop** banata hai
- Aapka async function `main()` **us loop mein run karta hai**
- Jab task complete ho jata hai, loop **automatically close** ho jata hai

✅ Final Real-life Analogy:

Socho `main()` aik delivery ka kaam hai, lekin ye kaam async hai — yaani rider ko wait bhi karna padta hai.

- `async def main()` rider ka kaam define karta hai

- `asyncio.run(main())` us rider ko bike dekar usay delivery pe bhejta hai 🚴‍♂️
- Aur jab delivery ho jaye, to rider wapas aa jata hai (event loop band)
- Python **har file mein** ek hidden variable `__name__` automatically set karta hai:

✨ Why is it important?

- Agar aap async function ko run karna chahte ho to `asyncio.run()` **zaroori** hai
- Warna error aayega: `RuntimeWarning: coroutine was never awaited`

❌ Kyun?

- `__name__` ek **Python ka built-in variable** hai.
- `"__main__"` bhi ek **exact string** hai.
- Agar aap spelling ya format change karti ho:
  - `__nmae__` → galat variable
  - `"__start__"` → galat string

✅ Sahi tarika:

```
if __name__ == "__main__":
    print("This file is running directly")
```

🔧 Function Tools in Simple Words:

Agent sirf jawab dene wala chatbot nahi hota — agar aap chahen ke wo koi **action perform kare** (jaise weather check karna, file read karna, calculation karna, ya API call karna), to aap uske liye ek **tool** banate ho.

✅ Real-Life Example (Simple):

User: "What is the weather in Karachi?"

- Agent ko pata hai is sawal ka jawab usko nahi aata — isliye wo tool call karta hai: `get_weather("Karachi")`
- Tool real weather API call karta hai aur result agent ko deta hai.
- Agent final answer user ko deta hai.

Function Tools Python Agent SDK (jaise `openai-agents`, `CrewAI`, etc.) mein is liye use kiye jaate hain **taake agents sirf text reply na dein, balkay actions bhi le saken**, jaise:

✅ 1. 💡 Yeh Tool Kab Call Hoga?

Ye tabhi call hoga **jab user ka input us tool se relevant ho**
Aapka agent smart hai, wo dekhta hai:

“Kya mujhe iska jawab khud pata hai ya mujhe kisi tool se help leni chahiye?”

Agar user bolta hai:
👉 `"What is the weather in Lahore?"`
To agent ko lagta hai:

“Hmm... mujhe to nahi pata. Lekin mere paas `get_weather()` tool hai!”

To agent yeh tool **automatically** call karega.

Jan ham handoffs kry gey tu wo hamra agent completely tabqeel hojyega jis agent sy sawal ho ga wohi jwb day ga lekin agr ham kisi function ko as a tool use kry gey tu is ka jwb wohi ho ga jo main agent ho ga

✅ LLM (jaise GPT-4 ya Gemini) sab kuch nahi jaanta

❌ LLM ke limitations kya hain?

LLM:

- **Offline trained model** hota hai.
- Yeh **Internet se live data nahi la sakta**.
- Sirf wo cheezein jaanta hai **jo uski training ke waqt available thi** (jaise 2023 tak ki knowledge).

LLM **sochta hai**:

“Yeh info meri knowledge mein nahi hai... mujhe kisi tool se puchhna padega.”

Phir agent:

- Us **function tool ko call karta hai**
- Tool se real data aata hai
- LLM us data ko use karke answer deta hai ✅

- Tool kab use hota hai?

Jab LLM ko live data chahiye hota hai