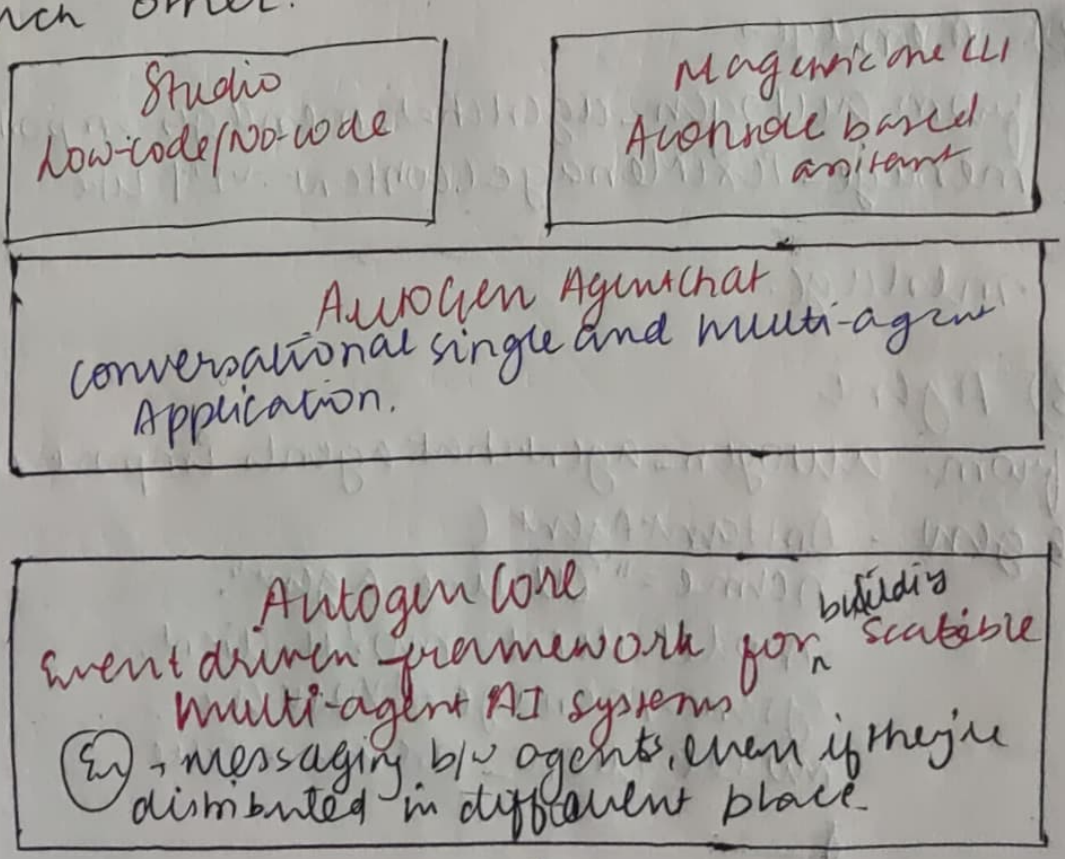
Week 5 → Microsoft Autogen 0.5.6

⇒ Autogen 0.4 released January 2025
→ "from-the-ground-up rewrite adopting an asynchronous, event driven architecture to address issues such as observation, flexibility, interactive control and scale.
→ This 0.4 version is an replacement for version 0.2.0.
→ several of the original developers of Autogen left microsoft and split off in late 2024 to create a fork of Autogen → called AG2 or Agent OS 2, compatible with earlier Autogen 0.2.
→ Whenever we do "pip install autogen", we get AG2 instead of Autogen
→ These are the bit of pieces that make Autogen and they are build on top of each other.



Core concept / or building blocks →

→ Models → similar to llm

→ Messages → represents messages b/w agents →

① b/w agent

② events that happens within an agents interaction.

→ Agents

→ Teams → groups of agent that can interact to achieve some goal.

→ Creating Tools and Database Integration →

→ Concept

① model →

```
from autogen_ext.model.openai import
                                OpenAIChatCompletion
model_client = OpenAIChatCompletion(model="gpt-4")
```

② message →

```
from autogen_agentchat.messages import TextMessage
message = TextMessage(content="I'd like ...", source=
                                "user")
message.
```

③ Agent

```
from autogen_agentchat.agents import AssistantAgent
agent = AssistantAgent(
    name="airline_agent",
    model_client=model_client,
    system_message="You are ....",
    model_client_stream=True,
)
```

→ New Concept
  → Multi-model
  → Structured Output
  → Tools from Langchain
  → Teams

→ Semantic kernel →

→ Semantic kernel is an open-source SDK by Microsoft that helps you build AI-powered apps. It acts like a bridge between LLM models and your own code, data, services. Think of it as a toolkit that makes it easier to combine :-

① AI reasoning (LLMs)
② Traditional Programming Language C# / Python / Java
③ Your data and APIs

Q What is Autogen core?

Sol → An Agent orientation framework.
  - An Agnostic to Agent's abstraction.
  → Similar positioning like Langgraph
  → But focus is on managing interactions between distributed and diverse Agents

Fundamental principles-

→ Decouples an Agents logic from how messages are delived.
→ framework handles creation & communication.
→ The Agents are responsible for their logic - that is not the remit of autogen core.

Two types of Runtime-

① Standalone → run on your system in a single way.
② Distributed → it runs in a way that allows remote agents to interact with each.

Coding →

@dataclass → decorator // for creating class to store data.

#define our own object

class Message:
    content : str

Define our Agent in Autogen core →

→ The agent created in ~~Message~~ messenger is different from that of Agent core.

→ In agent core → (management)

→ In this Every Agent has an unique id.

→ Every Agent has an AgentID which has 2 components.

① agent.id.type - describes the kind of agent

② agent.id.key - gives it its identifier

→ ⓐ message_handler → is an decorator method on agent that defines how the agent processes and responds to ~~it~~ incoming messages.

→ ⓑ On_message() → When an agent receives a message, it then invokes the agent on_message() method is responsible for implementing the logic

→ Distributed Runtime → Distributed Runtime is referred as experimental stated by Microsoft.

→ It is not ready for production system.

→ It is now used for getting ideas ~~for~~ for architecture