Who Am I?

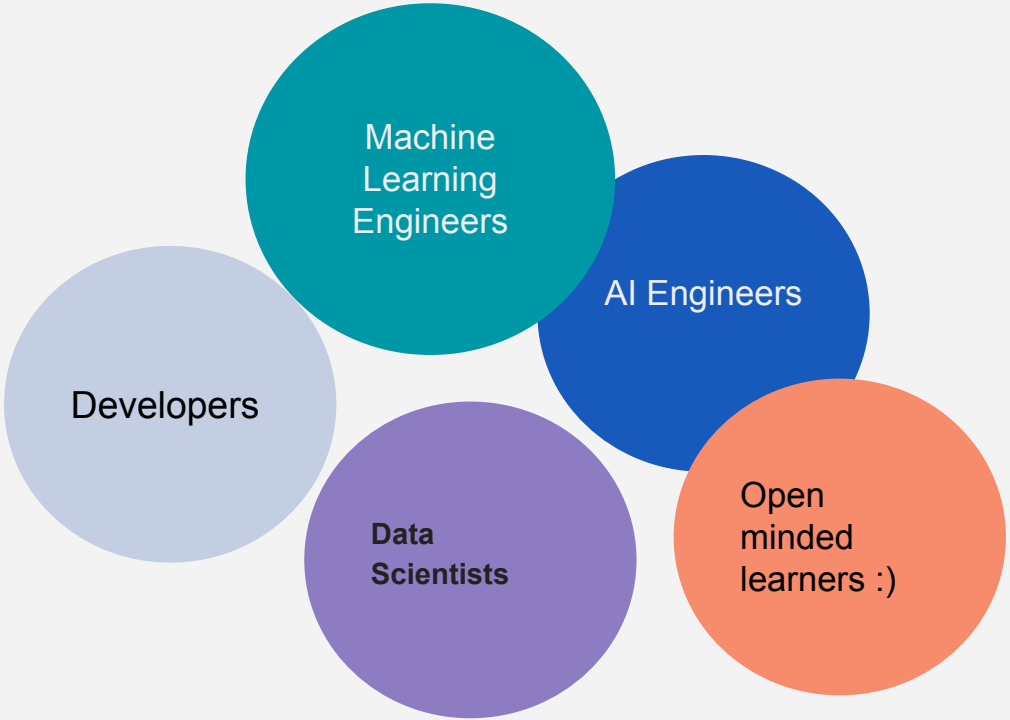
Paulo Dichone

Software, Cloud, AI Engineer
and Instructor

What Is This Course About?

- AI Agentic Patterns
  - Deep dive into the 4 main Agentic Patterns
    - Reflection
    - Tool Use
    - Planning
    - Multi-agent Pattern
  - Hands-on and use cases

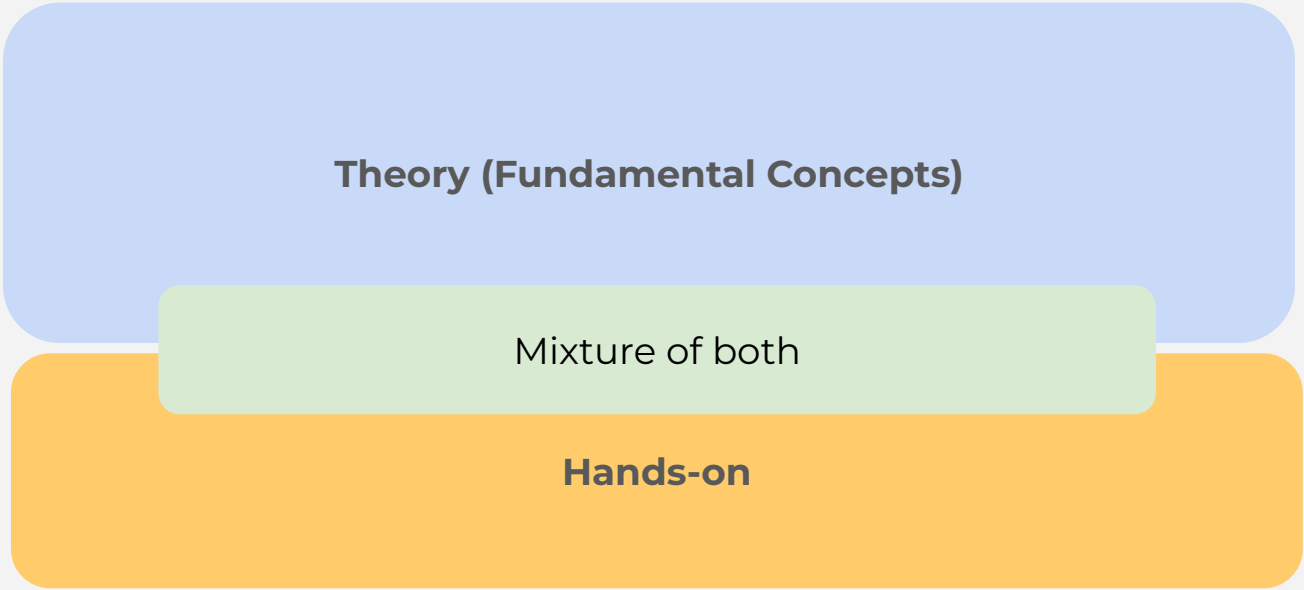
Who Is This Course For



Course Prerequisites

1. Know programming (highly *preferred… at least the basics)*
    a. *We will be using Python*
2. Basics of AI, ML, LLM, AI Agents
3. *This is not a programming course*
4. Willingness to learn :)

Course Structure



Development Environment setup

- Python
- VS Code (or any other code editor)
- OpenAI Account and an OpenAI API Key

Set up Ollama on Win and MacOS

Follow instructions here:
https://medium.com/@sridevi17j/step-by-step-guide-setting-up-and-running-ollama-in-windows-macos-linux-a00f21164bf3

Dev Environment Setup

Python (Win, Mac, Linux)

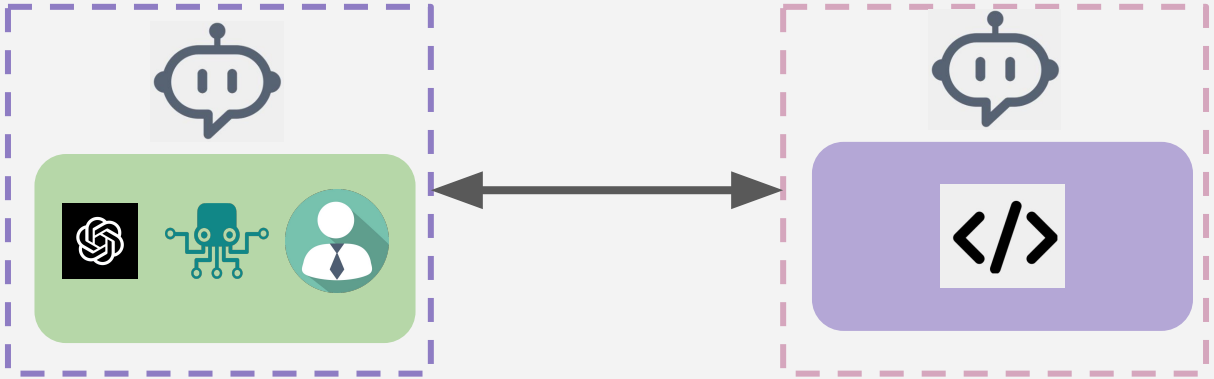
https://kinsta.com/knowledgebase/install-python/

AutoGen Deep Dive

- What is it?
- Why (motivation)?
- How it works?
- Key concepts

AutoGen

An open-source programming framework used for building AI agents that can **communicate** and **cooperate** with other agents to solve tasks.



AutoGen Main Features

Enables building next-gen LLM applications easier- we can build these applications based on **multi-agent conversations**

- Simplifies:
  - The orchestration
  - Automation
  - Optimization of complex LLM workflows
  - Heightens LLM models performance
  - Overcomes their weaknesses

Supports diverse conversation patterns (for complex workflows):

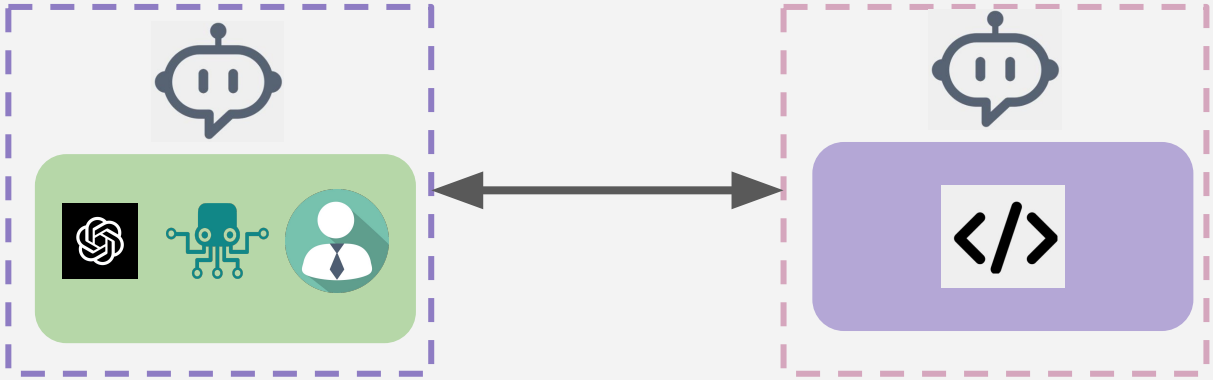
- Has customizable and conversational agents
- Developers can use AutoGen to build various types of conversation patterns

Provides a collection of working systems already:

- Developers can tap into systems that span a wide range of applications

AutoGen Building Blocks

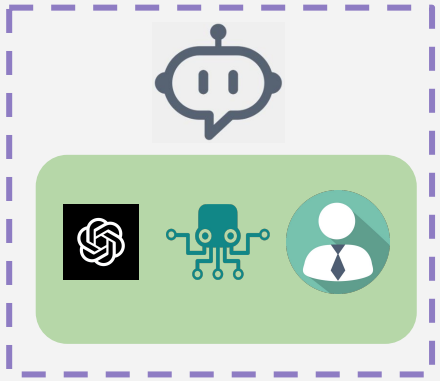
AutoGen **Agent** - an entity that can send and receive message to and from other agents.



AutoGen Building Blocks

Example of a AutoGen agent - *ConversableAgent*

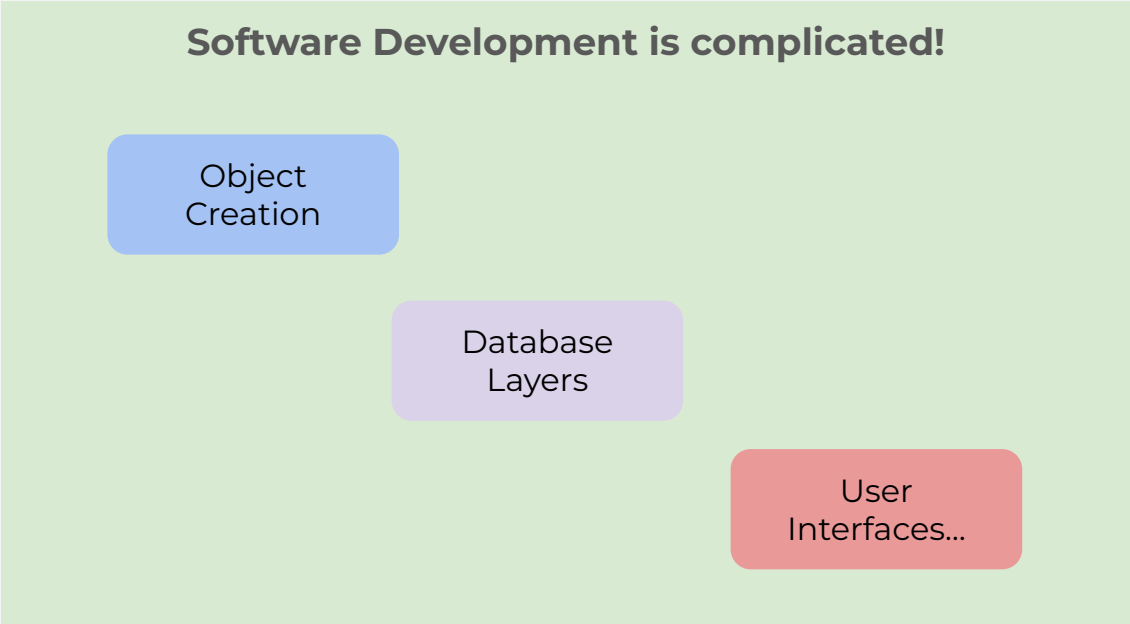
ConversableAgent



Has the following components:

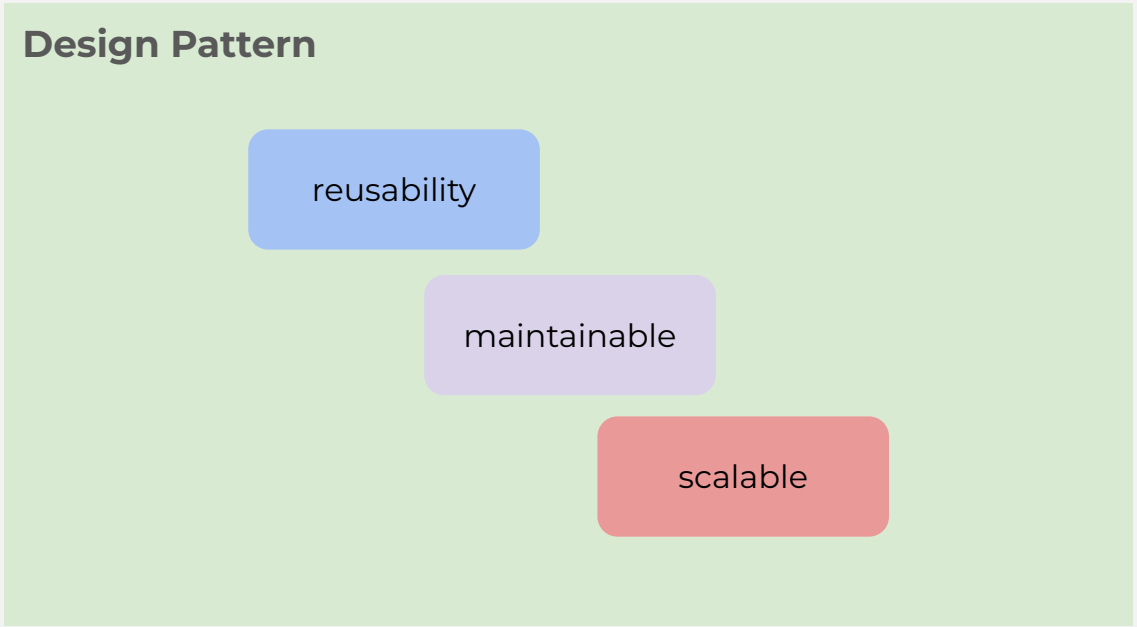
- A list of LLMs
- A code executor
- A function and tool executor
- A component for keeping human-in-the-loop

Design Patterns - Why?



Design Patterns help solve common problems efficiently

Design Patterns - Why?



No need to reinvent the wheel.

Want to structure a system for object creation? -- use the ***Factory Pattern***

Also we provide a shared Vocabulary and best practices

Design Patterns - in summary

Blueprints for coding/design best practices

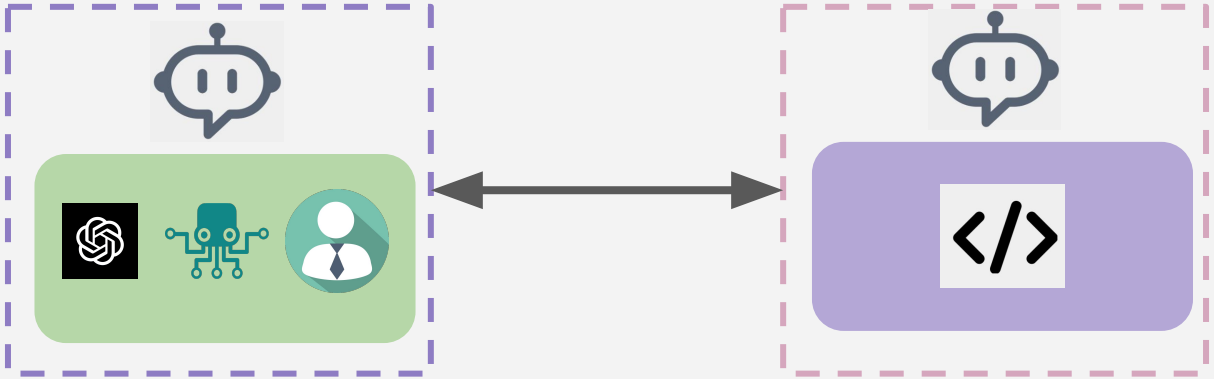
Save time

Reduce errors

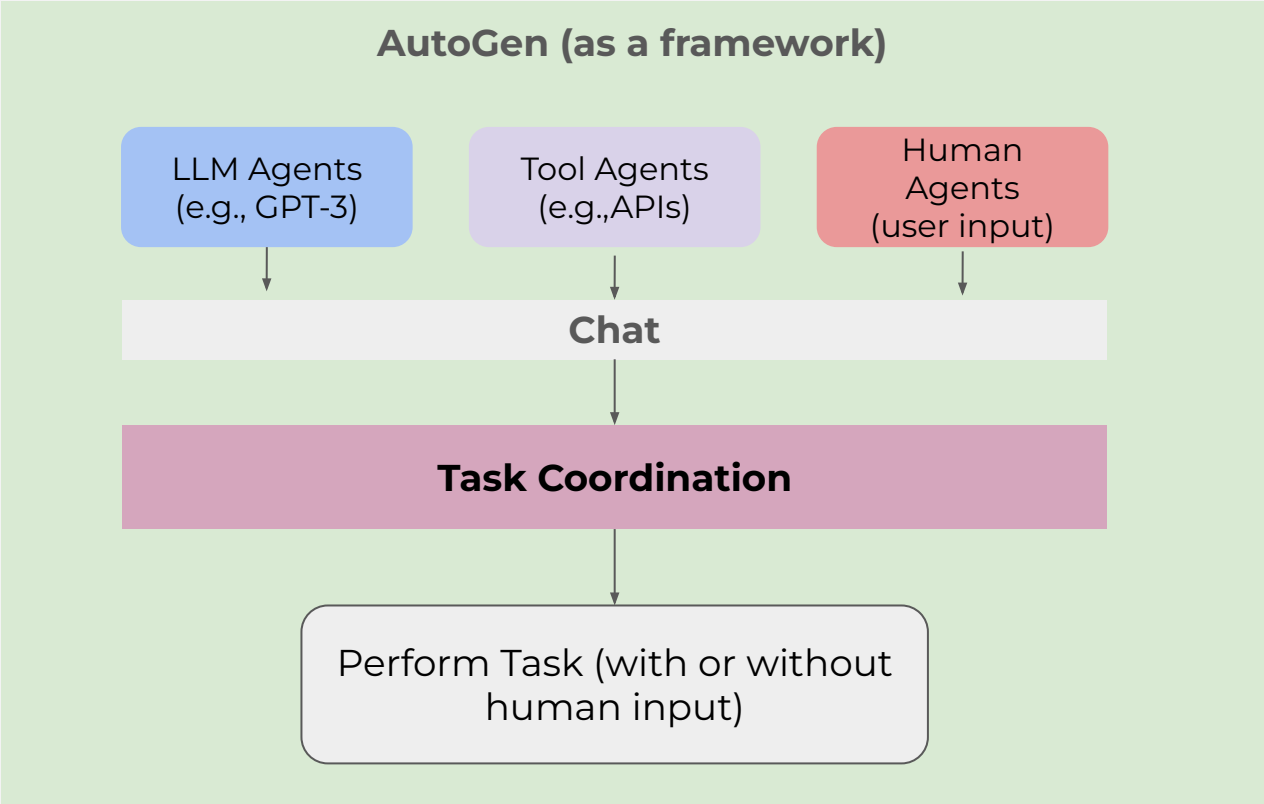
Provide structure

AutoGen Building Blocks

Multi-agents conversations
Conversation patterns



Multi-agents conversations



AutoGen

...makes it easy to create advanced applications using multiple agents, including AI models, tools and humans.

- **Organize and automate** - easily set up, automate, improve complex workflows that use LLMs
- **Boost performance -** enhances performance of LLMs and helps overcome their weaknesses
- **Easy to use -** minimal effort to build next-gen LLM applications
- **Flexible conversations -** support various conversation styles
- **Versatile Systems** - provides ready-to-use systems

Agents in AutoGen

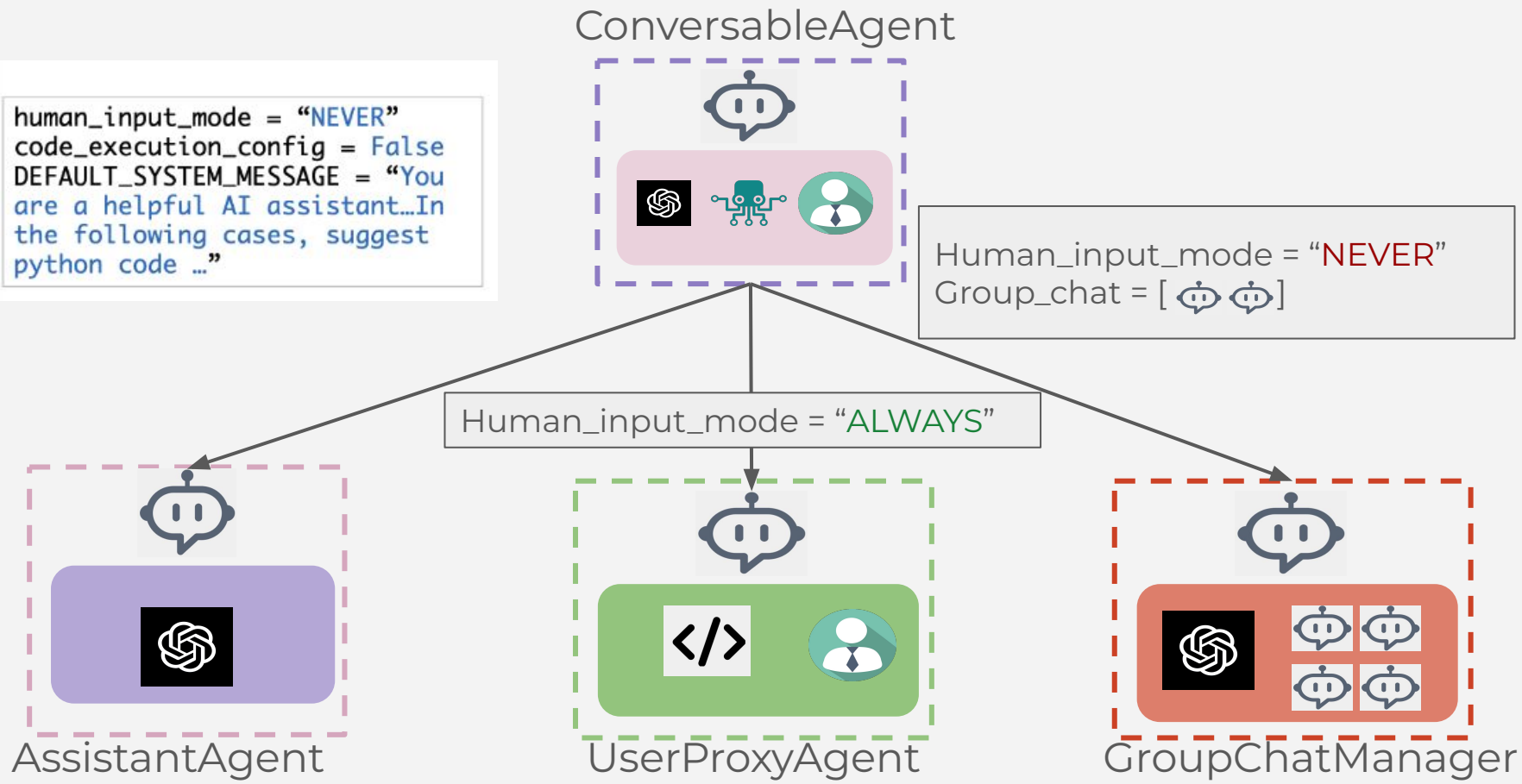
AutoGen abstracts out all intricacies about agents and how they work, and implements agents that communicate with each other to solve tasks.

These agents have two features:

- **Conversable** - agents can send and receive messages
- **Customizable** - agents can integrate with AI models, tools, humans or a combination of all.

Built-in Agents in AutoGen

```
human_input_mode = "NEVER"
code_execution_config = False
DEFAULT_SYSTEM_MESSAGE = "You are a helpful AI assistant…In the following cases, suggest python code …"
```



AI Agentic Patterns

- Key concepts
- Detailed breakdown
- Hands-on

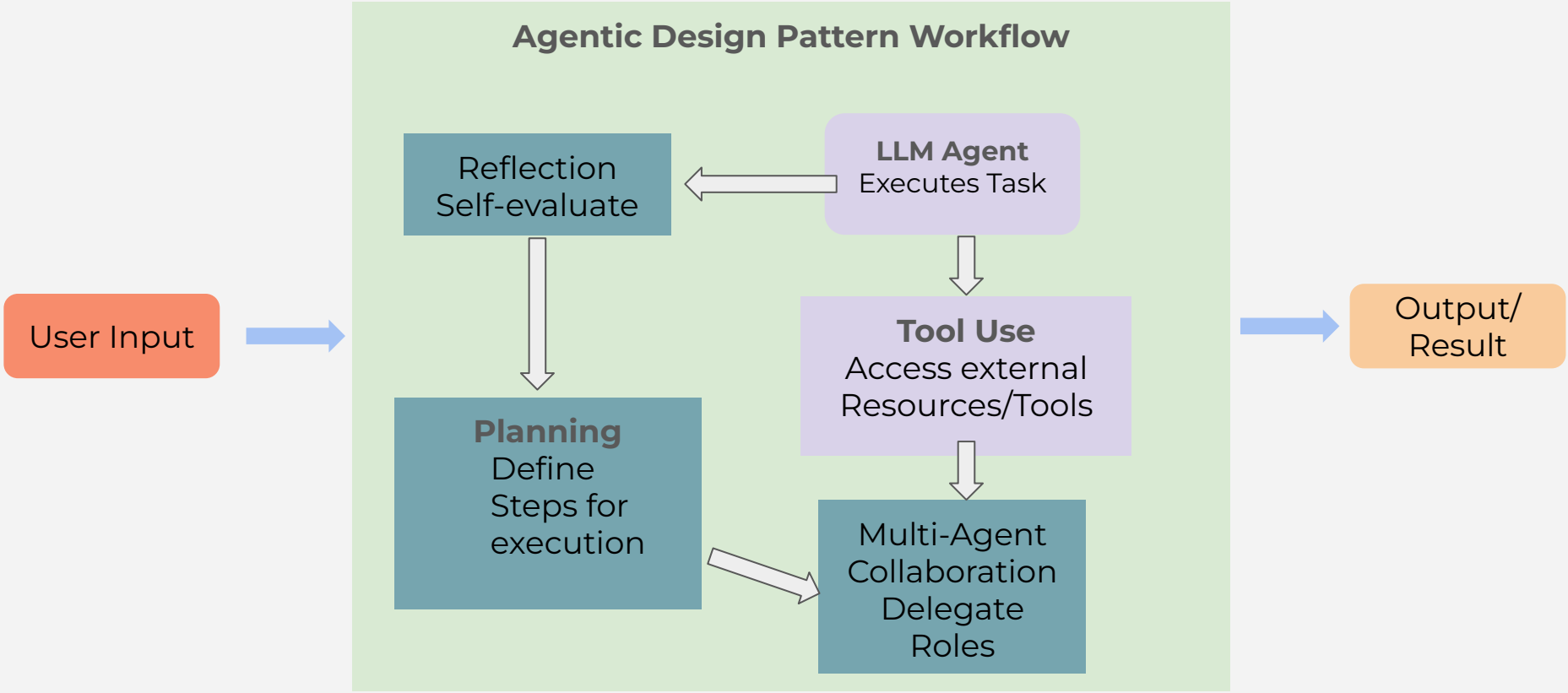
AI Agentic Design Patterns

Strategies that enhance the capabilities of LLMs by **structuring** their workflows into **iterative** and **collaborative** processes.

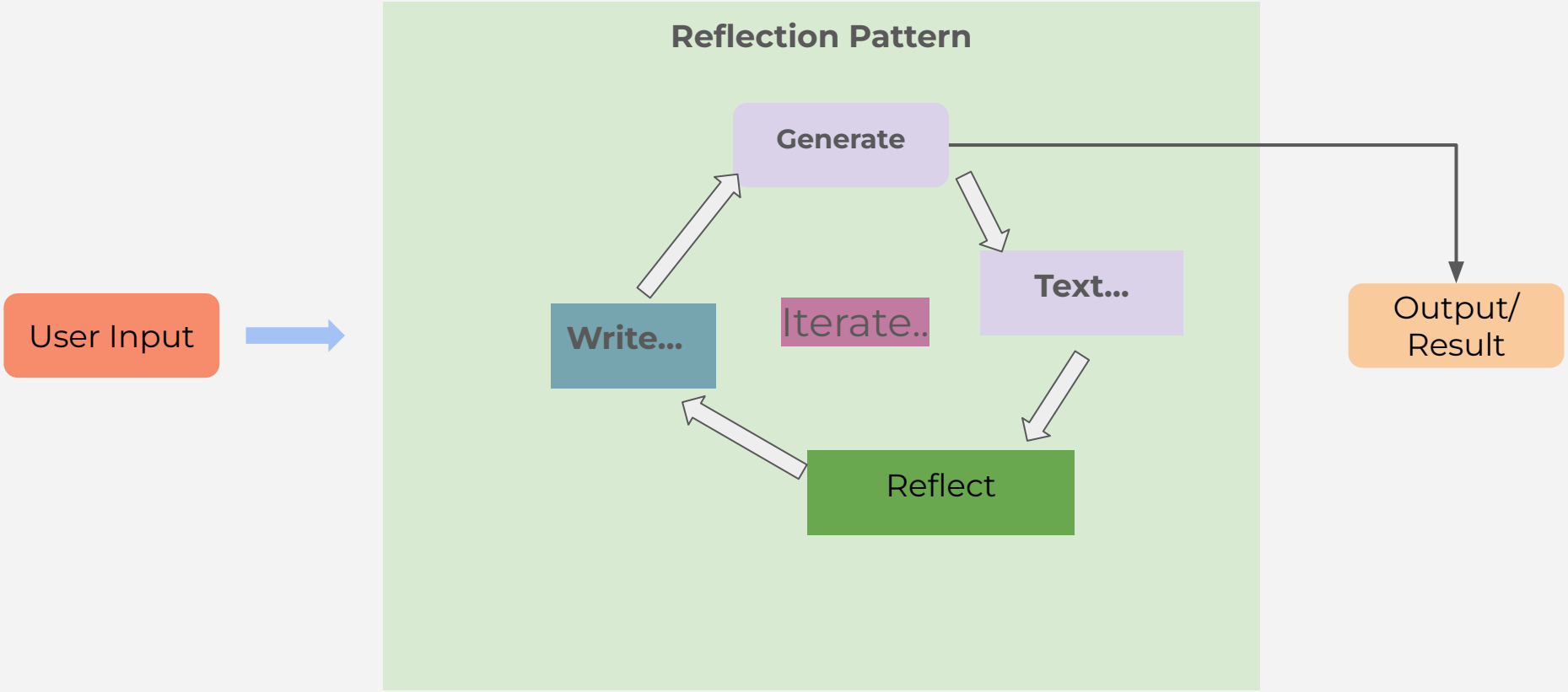
AI Agentic Design Patterns

Strategies that enhance the capabilities of LLMs by **structuring** their workflows into **iterative** and **collaborative** processes.

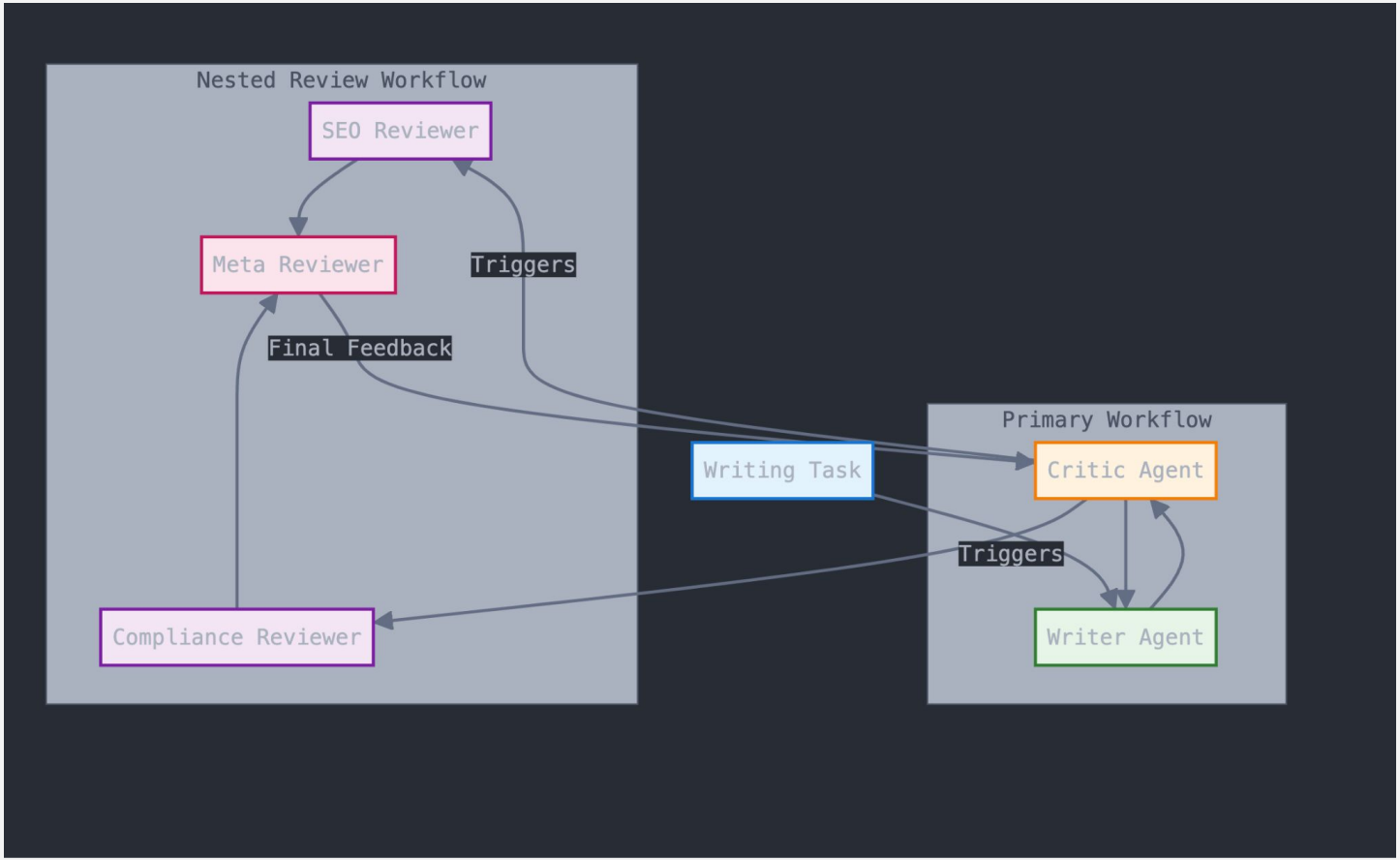
AI Agentic Design Patterns



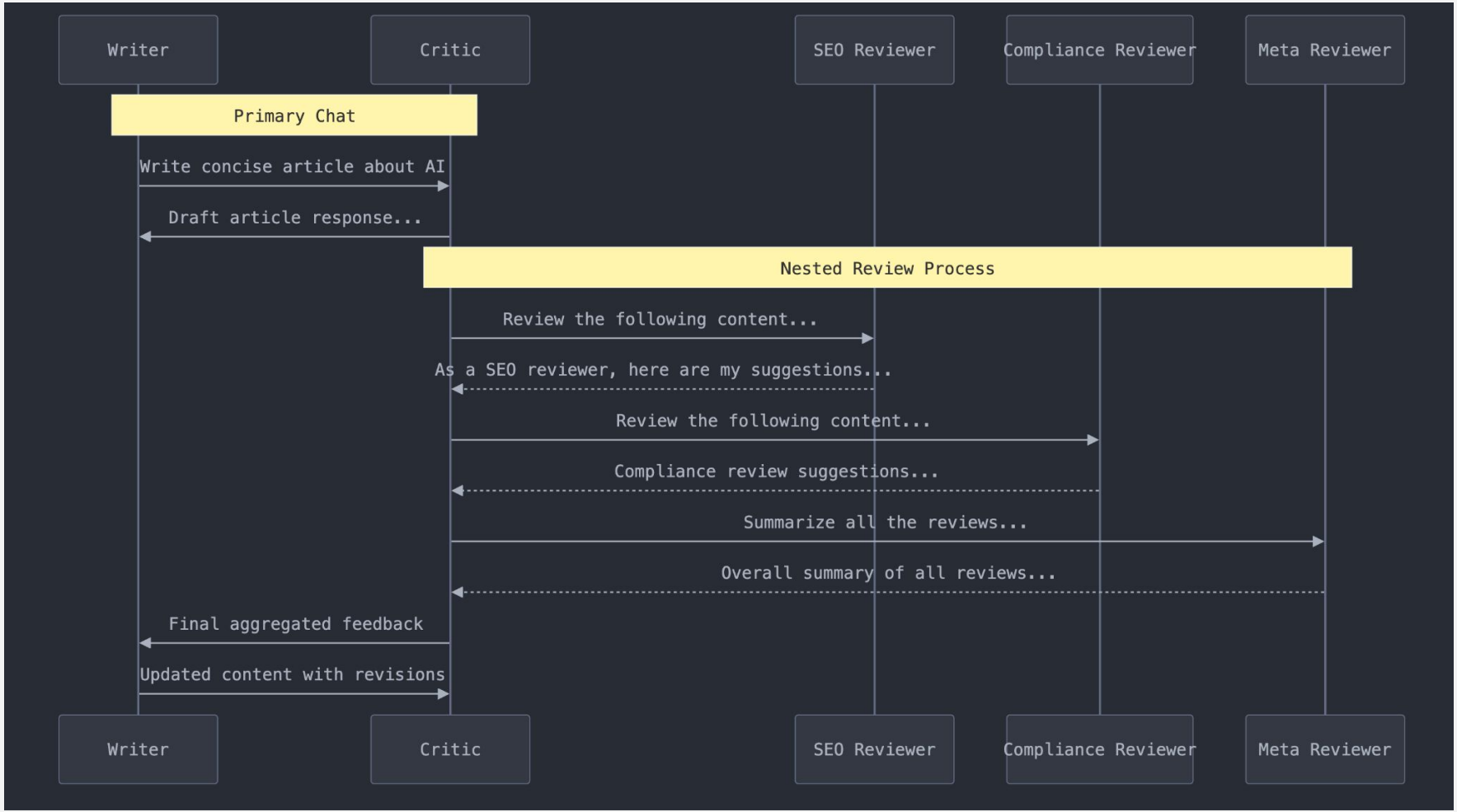
Reflection Pattern



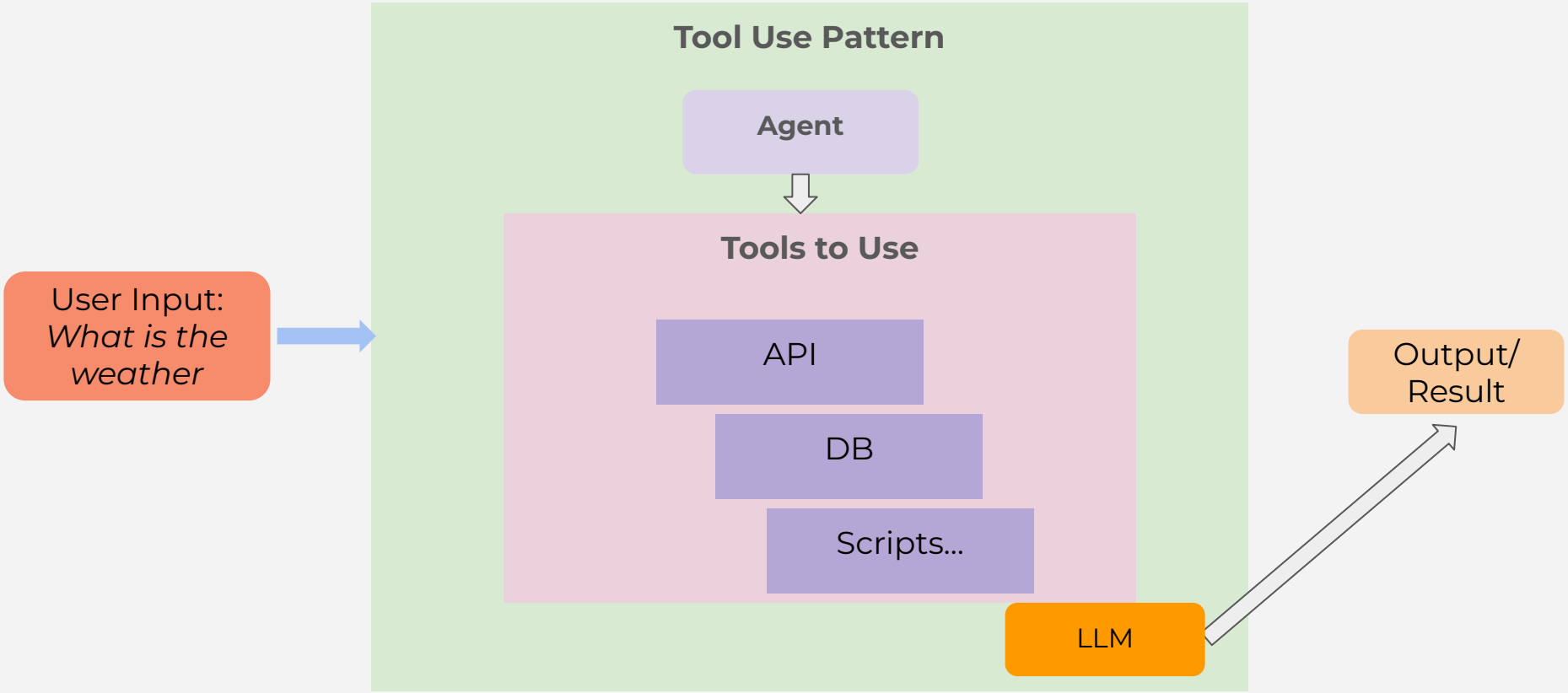
Reflection Pattern - Hands-on



Reflection Pattern - Hands-on



Tool Use Pattern



Tool Use Pattern - Key concepts

Augmentation

LLM use external tools - extending its capabilities.

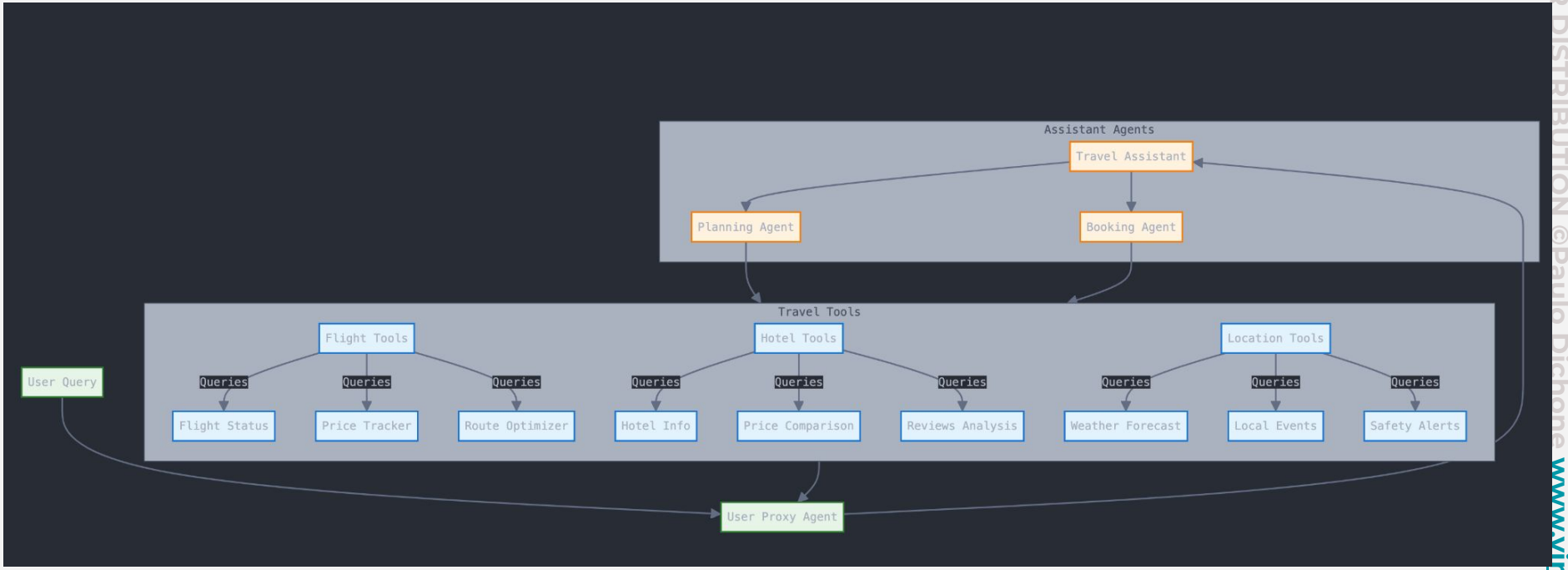
Dynamic Interaction

Real Time interaction - adaptive problem-solving.

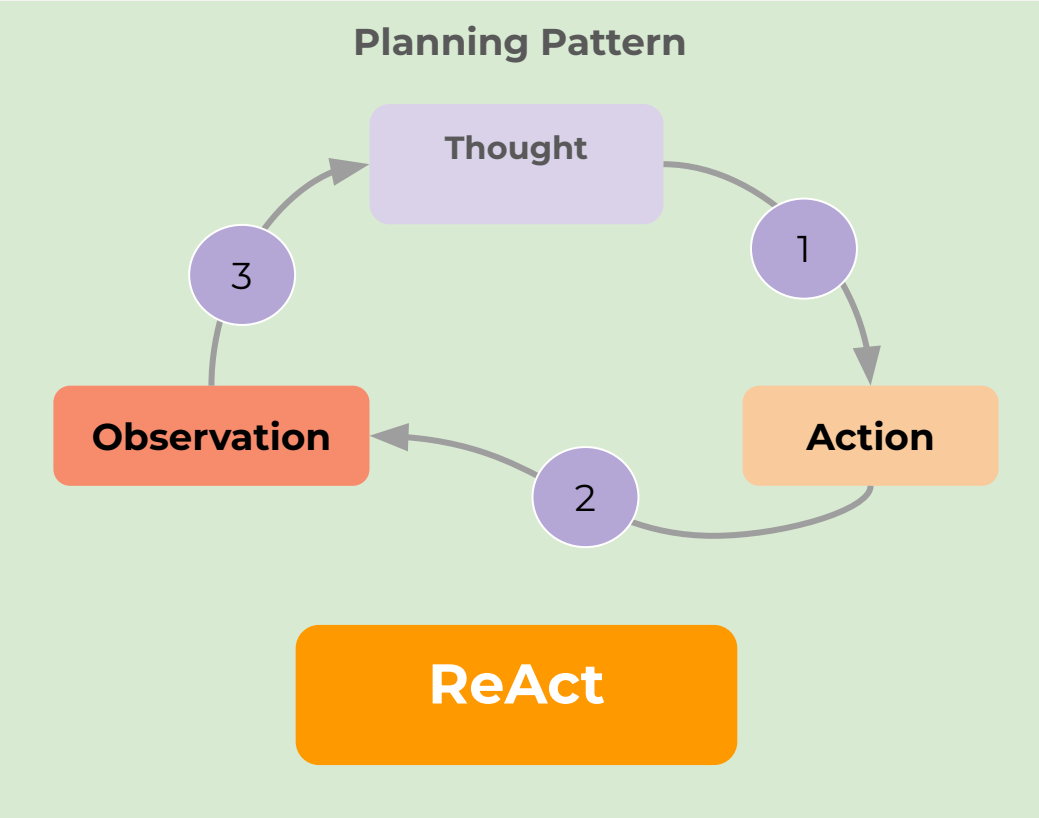
Modularity

Tools can be chosen or replaced easily.

Tool Use Pattern - Hands-on

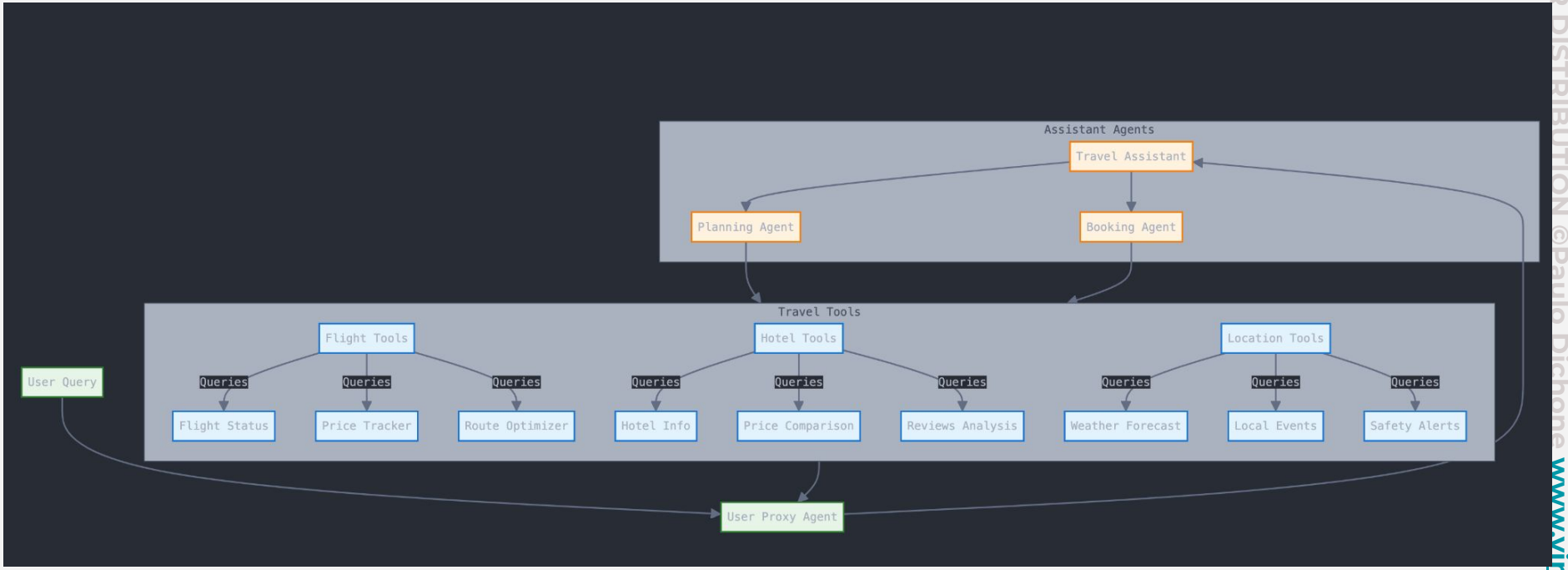


Planning Pattern -- ReAct Technique



ReAct - ***Reason and Act***

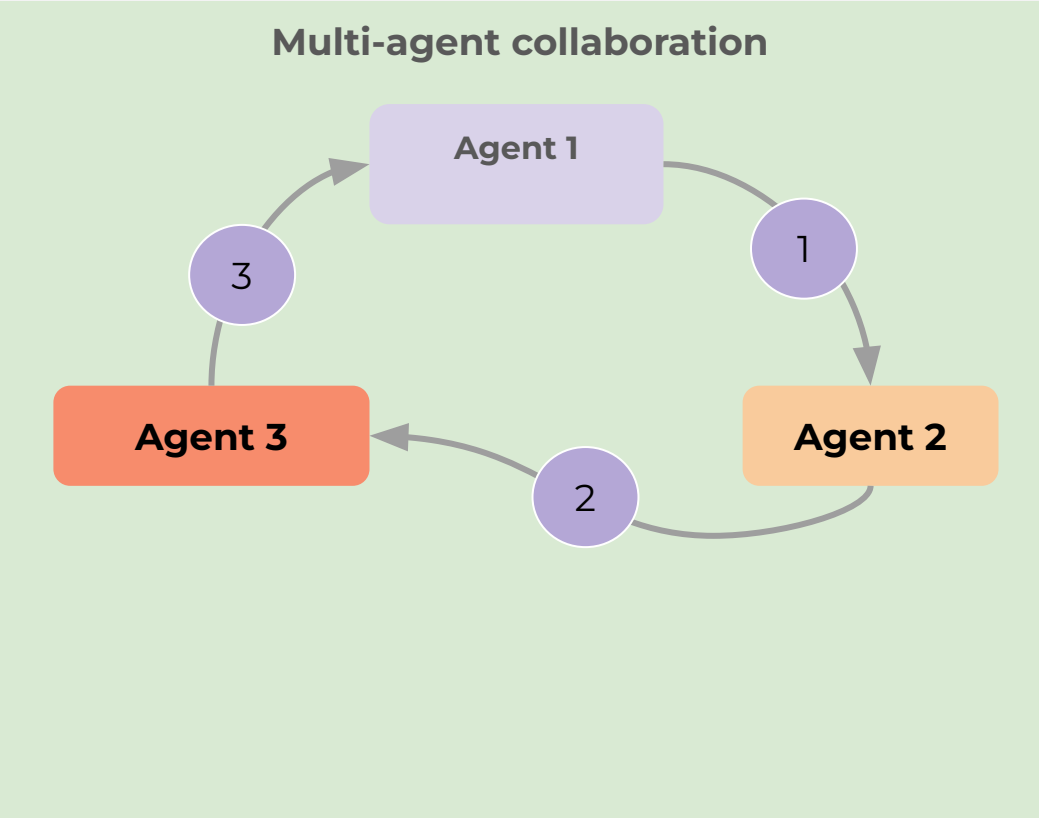
Planning Pattern (ReAct) - Hands-on



Planning Pattern - Key concepts

Task Analysis	LLM identifies the goal and steps.
Subtask Generation	Divide task into smaller subtasks.
Execution Sequence	Perform tasks in a sequence.
Iteration and Refinement	Results from a subtask may be fed into subsequent tasks.

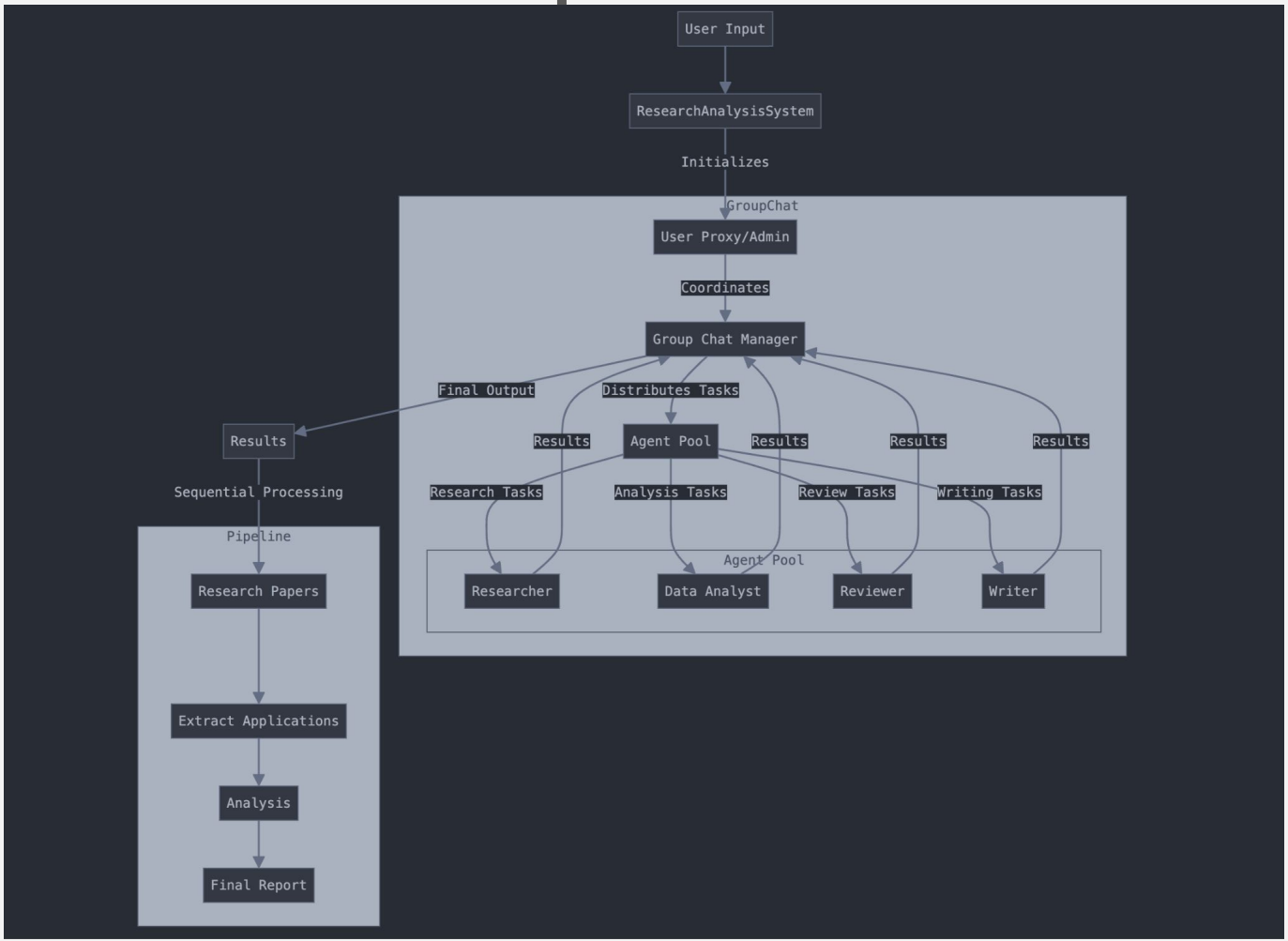
Multi-agent collaboration



Multi-agent Collaboration - Key concepts

Task division	Task breakdown
Role assignment	Assign roles to each agent
Agent communication	Agents talk with each other (share information)
Final assembly	Combine all contributions from all agents...

Hands on - Research Paper Workflow



Multi-agent Conversation Framework Flow

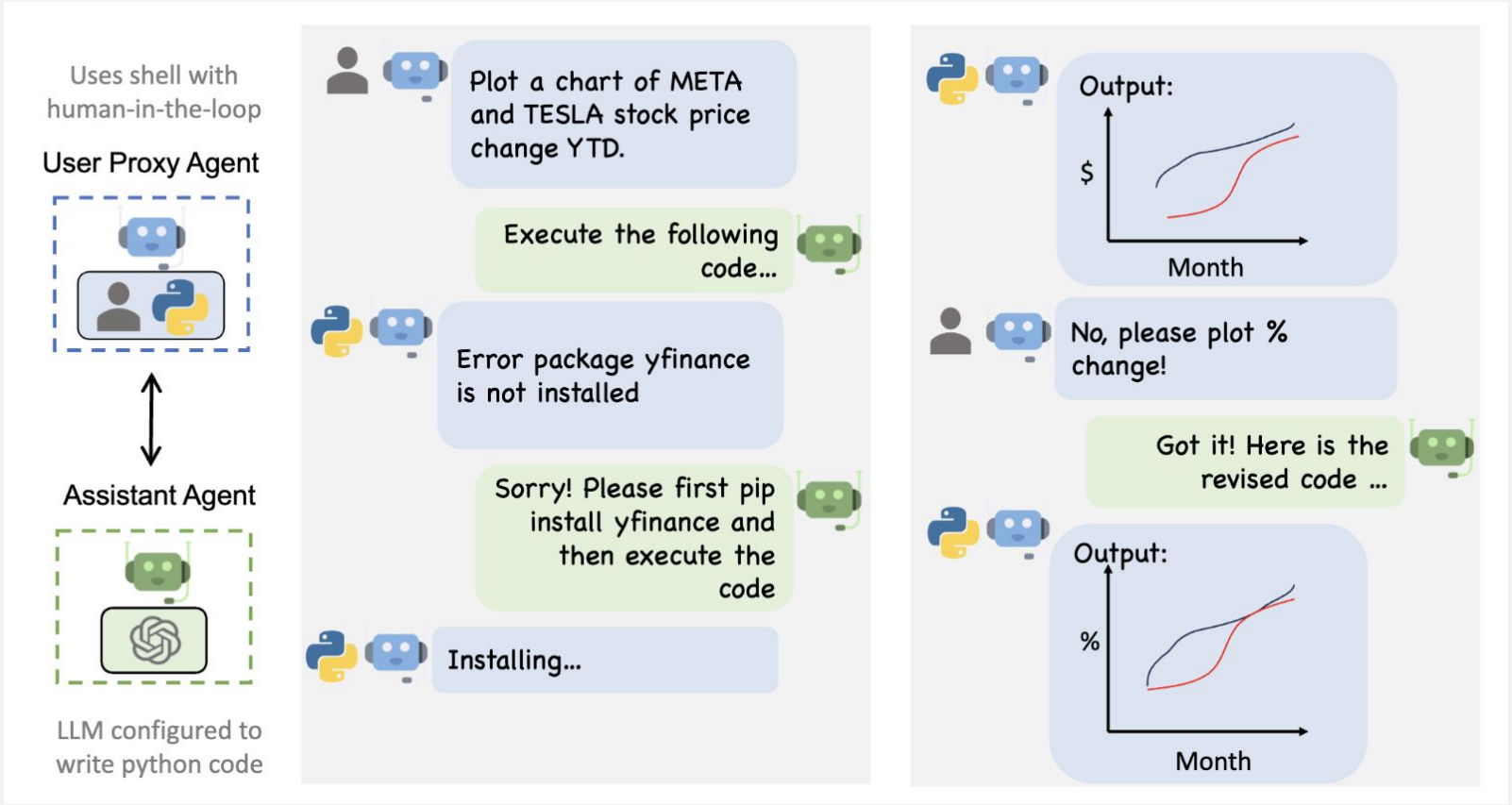


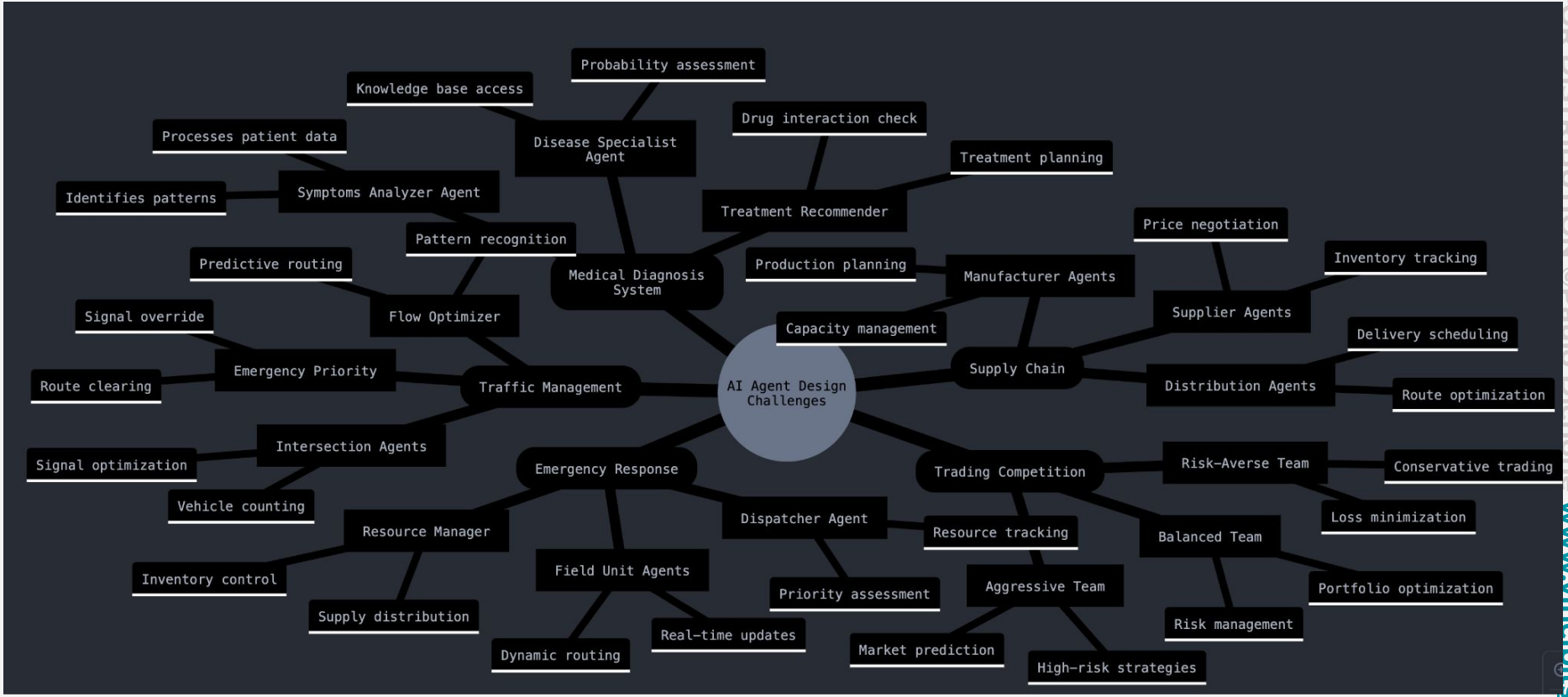
Image source: https://microsoft.github.io/autogen/docs/Getting-Started/

Congratulations!

You made it to the end!

- Next steps...

Challenge and use cases



Course Summary

- AI Agentic Design Patterns
  - Reflection Pattern
  - Tool use Pattern
  - Planning (ReAct) Pattern
  - Multi-agent Pattern (Collaboration)
- Hands-on

Wrap up - Where to Go From Here?

- Keep learning
  - Extend the projects we worked on in this course
  - Design and implement your own agents
- https://microsoft.github.io/autogen/docs/Getting-Started

Thank you!