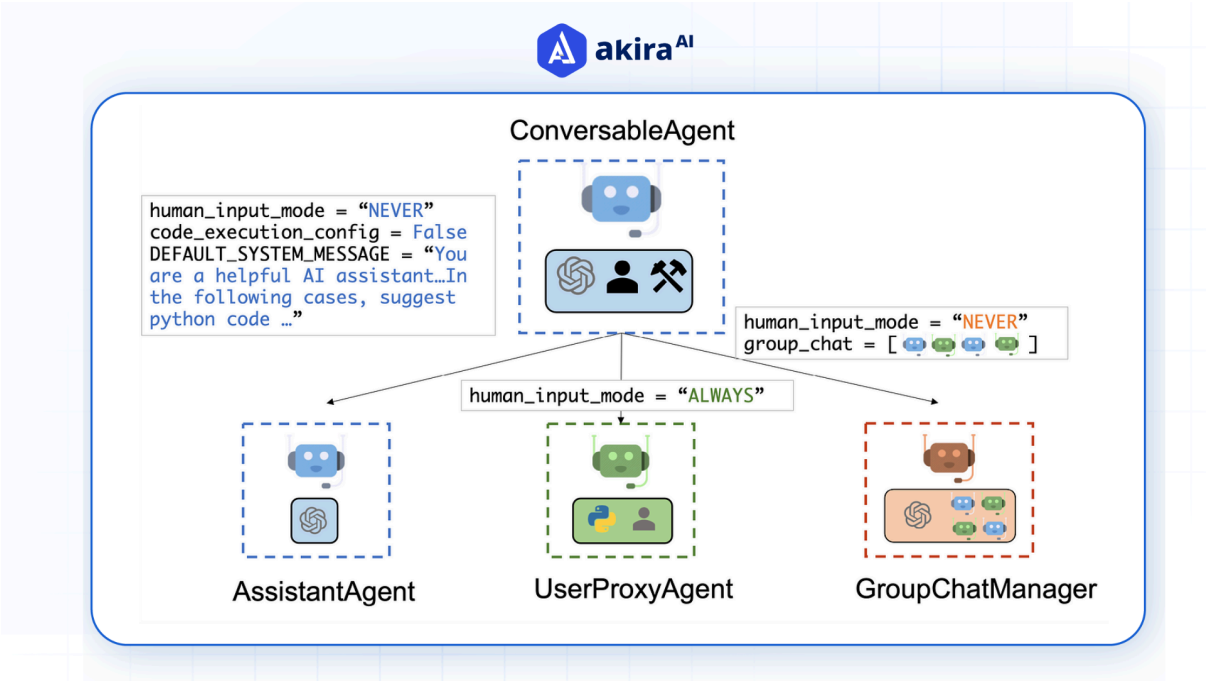
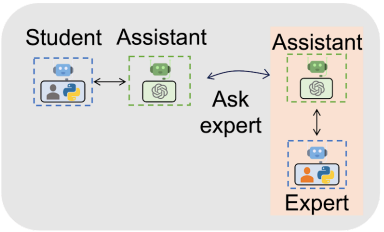
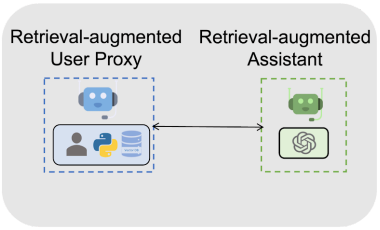
What is AutoGen??

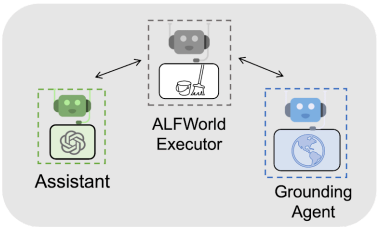




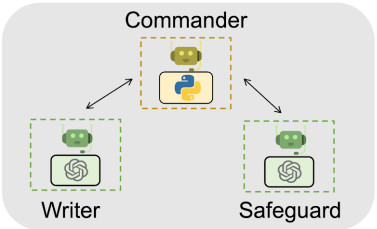
A1. Math Problem Solving



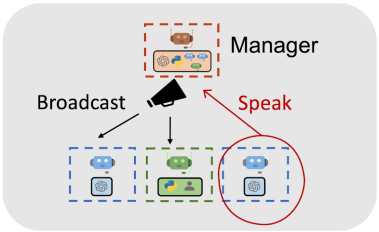
A2. Retrieval-augmented Chat



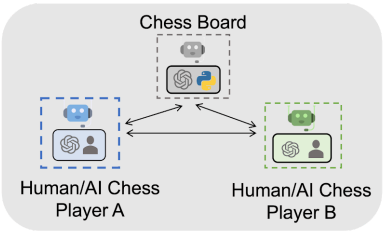
A3. Decision Making



A4. Multi-agent Coding



A5. Dynamic Group Chat



A6. Conversational Chess

Definition & Background

AutoGen is an open-source programming framework developed by Microsoft Research (in collaboration with academia) that enables the creation, orchestration and deployment of multi-agent AI and conversational systems. (Microsoft)

At its core, AutoGen allows you to build applications in which multiple “agents” (software components, often backed by large-language-models (LLMs) and other tools) **communicate with each other**—through asynchronous message-passing, conversation patterns, tool usage, human-in-the-loop or fully autonomous workflows. (Microsoft GitHub)

Key Components & Architecture

Here’s a breakdown of what AutoGen consists of and how it works:

- **Agents**: Each agent can represent a role (assistant, user proxy, tool, code executor, critic, etc). Agents can send and receive messages, make decisions, optionally execute code or tools. (arXiv)

- **Conversation / Message infrastructure**: Agents talk to each other via messages, supporting asynchronous workflows. This enables coordination, delegation, step-by-step reasoning. (Microsoft)

- **Tools & Execution**: Agents can be configured to call tools, execute code, retrieve information or perform actions (e.g., via local executors, Docker, web browsing). (GitHub)

- **Topology & Workflow Patterns**: The framework supports different patterns of agent interactions — simple two-agent chats, group chats, round-robin, nested workflows, etc. This provides flexibility. (DataCamp)

- **Extensions & Modular Design**: AutoGen has core APIs, agent-chat APIs and extensions (for model clients, code execution, different languages). This modularity allows customizing or scaling up. (Microsoft GitHub)

- **Observability / Distributed / Multi-language**: Later versions emphasise features like tracing, observability (OpenTelemetry), distributed agents across boundaries, and support for Python and .NET etc. (Microsoft)

What makes it different / why use it?

- Compared to simpler frameworks (just chat with a single LLM) AutoGen brings **agentic structure** — i.e., multiple cooperating components each with distinct roles, enabling more complex orchestration. (arXiv)

- It addresses some of the limitations of monolithic LLM use-cases (such as scaling, tool usage, reasoning by decomposition) by structuring conversations and workflows. (Microsoft GitHub)

- Offers abstraction layers: high-level APIs for rapid prototyping (AgentChat), a GUI (Studio) for low-code/no-code and core runtime for deep customisation. (DataCamp)

Use-Cases of AutoGen

Here are several use-cases, showing how AutoGen can be applied in real scenarios.

1. Code Generation, Execution & Debugging

AutoGen is frequently used to build systems where:

- An “assistant” agent writes code (in Python, shell, etc) based on a task description.
- A “code executor” agent runs that code, captures outputs or errors.
- A “critic” or “reviewer” agent checks the results, suggests fixes.
- A “user proxy” agent (human or human-represented) initiates tasks and reviews results.

Why it’s useful: developers can prototype complex pipelines (e.g., data analysis, automation scripts, dev-ops tasks) by structuring interactions rather than writing monolithic scripts. Also supports tool integration (e.g., database queries, web scraping).

2. Multi-Agent Conversation & Task Orchestration

In many applications you need more than one role: for example:

- One agent plans a strategy, another agent executes steps, another checks quality, another handles exceptions or human feedback.
- Agents may communicate in turn, pass tasks, decompose big problems into subtasks.

Why useful: works well for complex domains like operations research, decision making, business workflows, customer-service pipelines, where tasks are composite and need coordinated steps.

3. Human-in-the-Loop / Assisted Automation

AutoGen allows combining human agents (or proxies) with automated agents. For example:

- A human approves actions or provides feedback to the automated agents.
- Agents generate recommendations, humans review, then agents execute.

Why useful: In domains where full automation is risky (legal, medical, financial), having human oversight while automating parts of the workflow saves time while retaining control.

4. Tools & External Systems Integration

Because agents can use tools, execute code, perform web browsing, query databases, etc., AutoGen is applicable for workflows that interact with external systems. For example:

- An agent queries a database, another processes results, another generates a report.
- An agent uses a web scraping tool, passes results to another agent for summarisation.
- Agents coordinate to monitor dashboards, trigger alerts, and execute remedial code.

Why useful: Many real-world AI systems are not just “chat with model” but “model + tool + action”. AutoGen bridges that gap.

5. Domain Specific Systems: Education, Research, Business Intelligence

Examples:

- Education: Build an agent network that guides a student (one agent asks questions, another evaluates answers, another recommends resources).
- Research assistant: An agent proposes experiments, another executes code, another writes reports, another critiques.
- Business intelligence: Agents parse incoming data feeds, another translates to insights, another builds dashboards, another alerts stakeholders.