

---

# AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation

---

**Qingyun Wu<sup>†</sup>, Gagan Bansal<sup>\*</sup>, Jieyu Zhang<sup>±</sup>, Yiran Wu<sup>†</sup>, Beibin Li<sup>\*</sup>**

**Erkang Zhu<sup>\*</sup>, Li Jiang<sup>\*</sup>, Xiaoyun Zhang<sup>\*</sup>, Shaokun Zhang<sup>†</sup>, Jiale Liu<sup>±</sup>**

**Ahmed Awadallah<sup>\*</sup>, Ryen W. White<sup>\*</sup>, Doug Burger<sup>\*</sup>, Chi Wang<sup>\*1</sup>**

<sup>\*</sup>Microsoft Research, <sup>†</sup>Pennsylvania State University

<sup>±</sup>University of Washington, <sup>±</sup>Xidian University

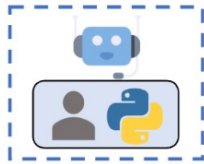
Presented by:  
John Tan Chong Min

Live Demo by:  
Soham Sarkar (Crit)

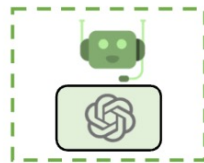
# Imagine this

Uses shell with  
human-in-the-loop

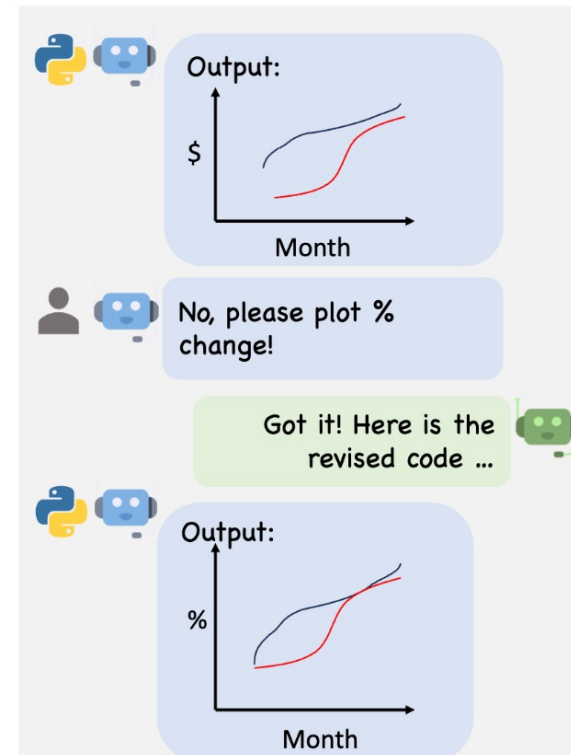
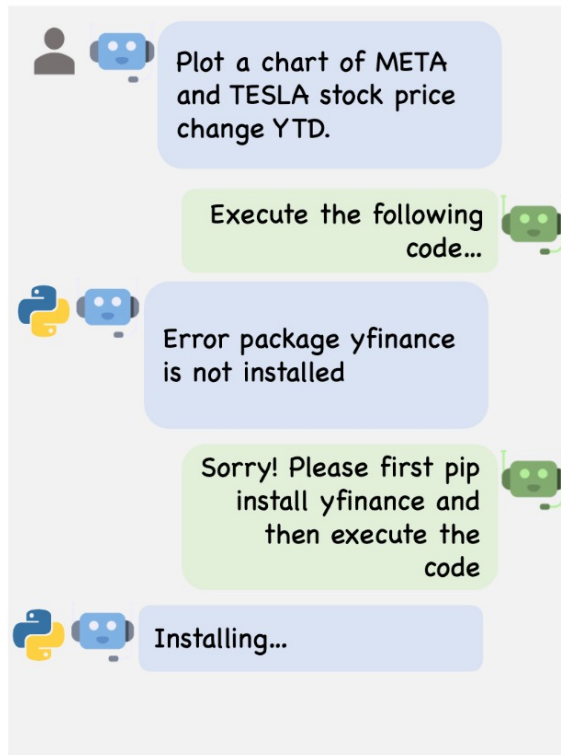
User Proxy Agent



Assistant Agent



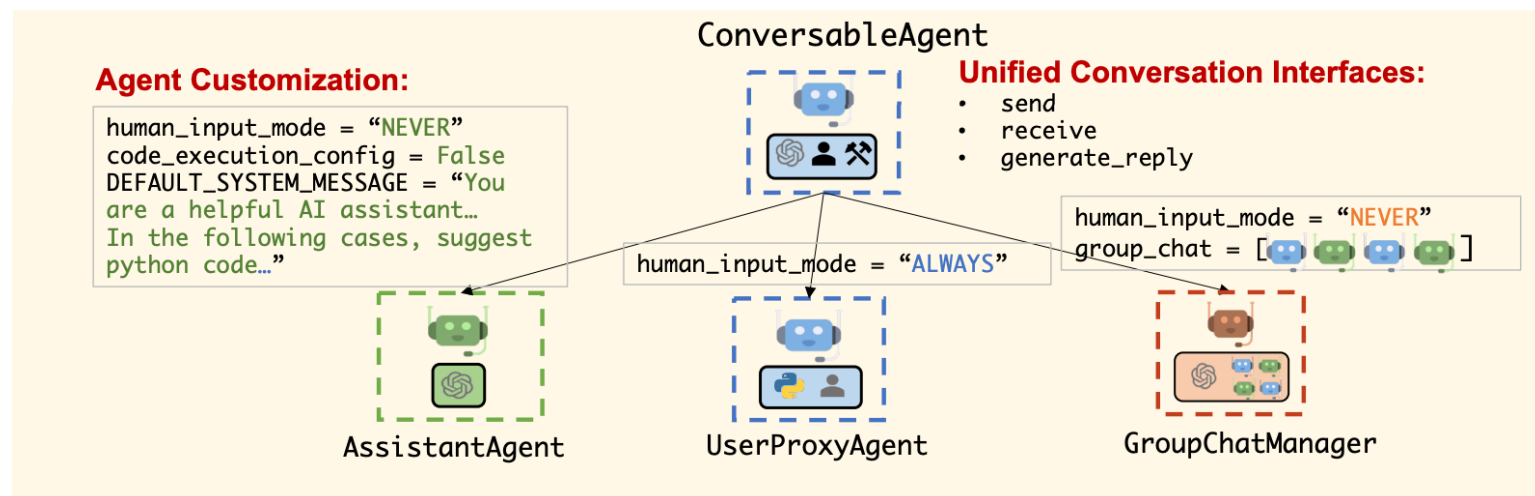
LLM configured to  
write python code



# Agents

- Agent has generally 3 functions: **send**, **receive**, **generate\_reply**
- You can also be an agent with `human_input_mode = "ALWAYS"`

## AutoGen Agents



# Flexible Customisable Functions

- Agent has template code which can be modified to suit your needs

## Developer Code

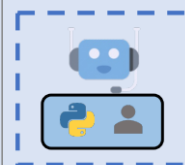
### 1.2 Register a Custom Reply Func:

```
# This func will be invoked in
generate_reply

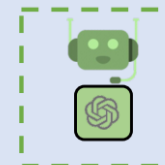
A.register_reply(B,
reply_func_A2B)
def reply_func_A2B(msg):
    output = input_from_human()

    if not output:
        if msg includes code:
            output = execute(msg)
    return output
```

### 1.1 Define Agents:



User Proxy A



Assistant B

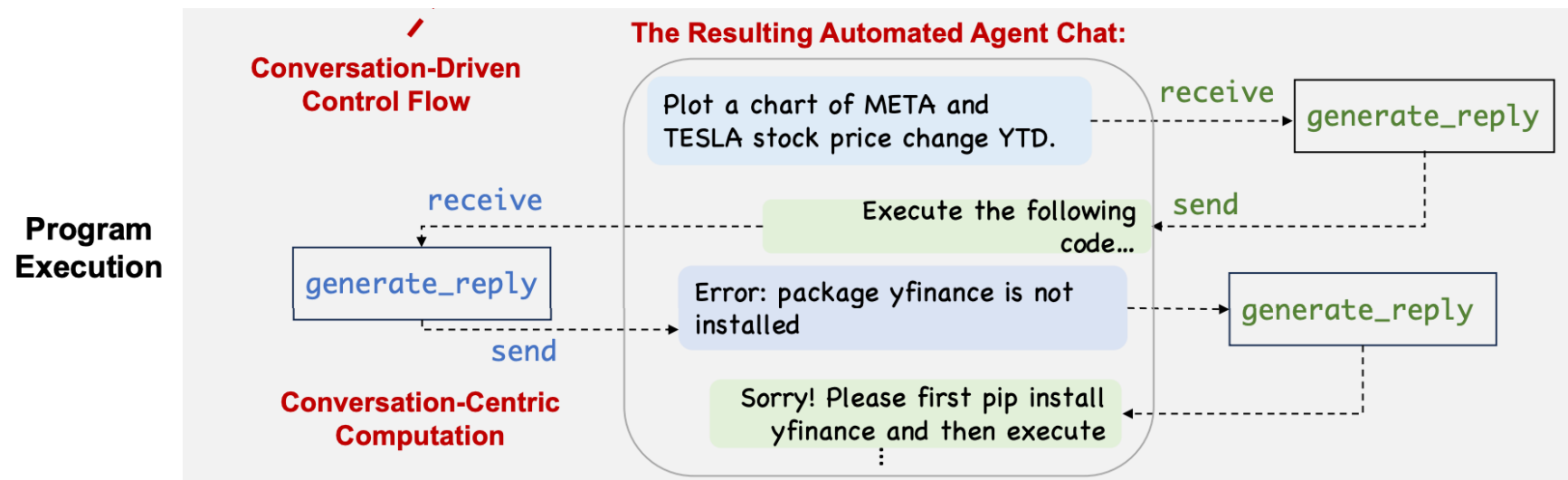
# Note: when no reply  
func is registered, a  
list of default reply  
functions will be used.

### 2 Initiate Conversations:

```
A.initiate_chat("Plot a chart of META and  
TESLA stock price change YTD.", B)
```

# Conversation-based Control Flow

- **receive:** Agent chooses what to receive
- **generate\_reply:** Agent generates reply based on what it receives
- **send:** Agent sends reply to specified agents



# Versatile Agent Structure

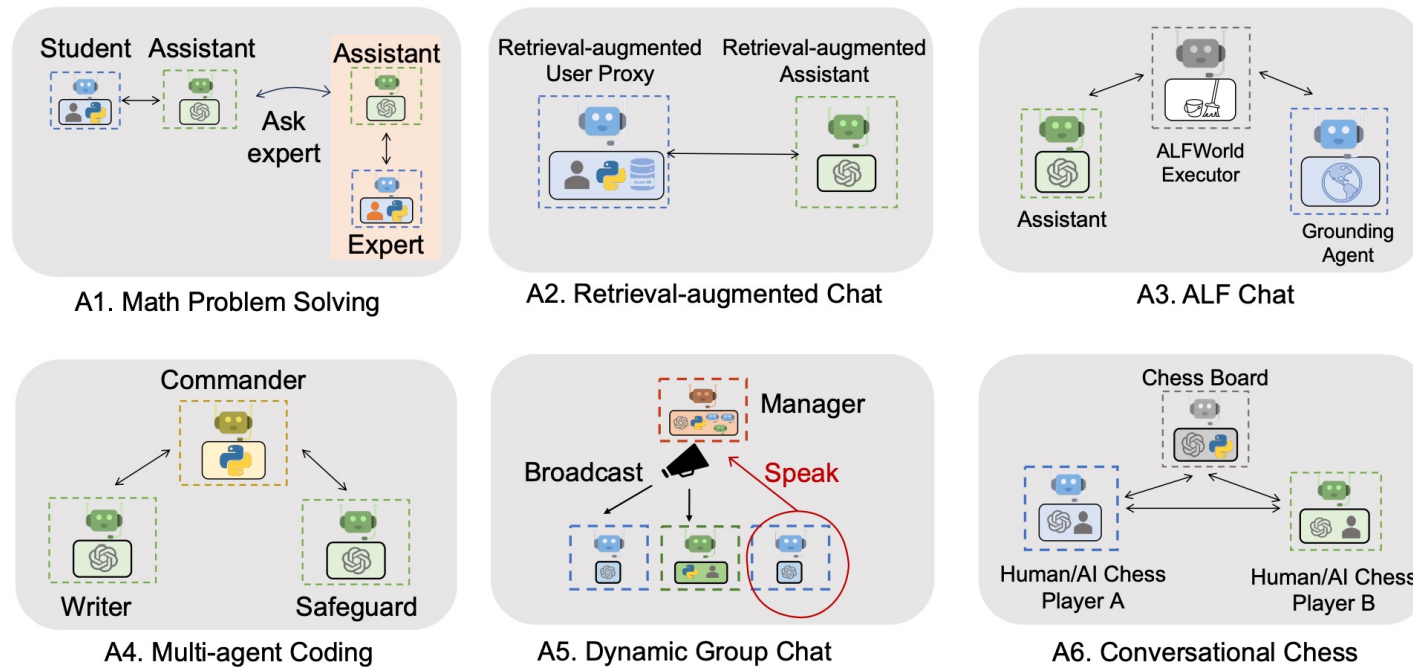


Figure 3: Six examples of diverse applications built using AutoGen. Their conversation patterns show AutoGen's flexibility and power.

# System Message

You are a helpful AI assistant. Solve tasks using your coding and language skills.

In the following cases, suggest python code (in a python coding block) or shell script (in a sh coding block) for the user to execute.

1. When you need to collect info, use the code to output the info you need, for example, browse or search the web, download/read a file, print the content of a webpage or a file, get the current date/time. After sufficient info is printed and the task is ready to be solved based on your language skill, you can solve the task by yourself.
2. When you need to perform some task with code, use the code to perform the task and output the result. Finish the task smartly.

Solve the task step by step if you need to. If a plan is not provided, explain your plan first. Be clear which step uses code, and which step uses your language skill.

When using code, you must indicate the script type in the code block. The user cannot provide any other feedback or perform any other action beyond executing the code you suggest. The user can't modify your code. So do not suggest incomplete code which requires users to modify. Don't use a code block if it's not intended to be executed by the user.

If you want the user to save the code in a file before executing it, put # filename: <filename> inside the code block as the first line. Don't include multiple code blocks in one response. Do not ask users to copy and paste the result. Instead, use 'print' function for the output when relevant. Check the execution result returned by the user.

If the result indicates there is an error, fix the error and output the code again. Suggest the full code instead of partial code or code changes. If the error can't be fixed or if the task is not solved even after the code is executed successfully, analyze the problem, revisit your assumption, collect additional info you need, and think of a different approach to try.

When you find an answer, verify the answer carefully. Include verifiable evidence in your response if possible.

Reply "TERMINATE" in the end when everything is done.

- Code as part of conversation enclosed in python coding block for regex extraction later
- Could perhaps be better formatted as JSON? (e.g. StrictJSON)

# Retrieval Augmented Generation on demand via Agentic Interaction

- "Update Context" is cue for RAG

## Task description

who carried the usa flag in opening ceremony?

### (a) W/O Interactive Retrieval

**Assistant:** You're a retrieve augmented chatbot. You answer user's questions based on your own knowledge and the context provided by the user. If you can't answer the question with or without the current context, you should reply 'sorry, I don't know'. You must give as short an answer as possible.

User's question is: who carried the usa flag in opening ceremony  
Context is: < P > On January 17 , 1899 , under ... The Astros beat the Yankees that night , 2 - 1 . < /P >

**UserProxy:** Sorry, I cannot find any information about who carried the USA flag in the opening ceremony. Sorry, I don't know.



### (b) W/ Interactive Retrieval

**Assistant:** You're a retrieve augmented chatbot. You answer user's questions based on your own knowledge and the context provided by the user. If you can't answer the question with or without the current context, you should reply exactly 'UPDATE CONTEXT'. You must give as short an answer as possible.

User's question is: who carried the usa flag in opening ceremony  
Context is: < P > On January 17 , 1899 , under ... The Astros beat the Yankees that night , 2 - 1 . < /P >

**UserProxy:** Sorry, I cannot find any information about who carried the USA flag in the opening ceremony. UPDATE CONTEXT.

**Assistant:** You're a retrieve augmented chatbot. You answer user's questions based on your own knowledge and the context provided by the user. If you can't answer the question with or without the current context, you should reply exactly 'UPDATE CONTEXT'. You must give as short an answer as possible.

User's question is: who carried the usa flag in opening ceremony  
Context is: < T able > ... Erin Hamlin ... < /T d >

**UserProxy:** Erin Hamlin carried the USA flag in the opening ceremony.





# Grounding in Environment Reminder can be an agent

- What is typically a system message can also be an agent
  - Efficient?
- Conversation flow is akin to reflection and can improve performance

## Task description

You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 2, a desk 1, a safe 1, a drawer 2, a drawer 1, a shelf 3, a shelf 2, and a shelf 1.  
**Your task is to:** look at bowl under the desklamp.

### (a) Two agents

```
Assistant(to Executor): THOUGHT [First, I need to find a bowl. A
bowl is more likely to appear in desk (1-2), drawer (1-2), shelf
(1-3), bed (1). Then I need to find and use a desk lamp.] ACTION
[go to desk 1]
(Assistant finds the desk lamp at desk 1, then goes to desk 2.)
Executor(to Assistant): On the desk 2, you see a bowl 1, and a
cd 3...
Assistant(to Executor): THOUGHT [Now I find a bowl (1). I need
to use the desk lamp to look at the bowl. I'll go to the desk lamp
now.] ACTION [go to desk 1]
Executor(to Assistant): On the desk 1, you see a desk lamp 1, a
pencil 2...
Assistant: ACTION [use desk lamp 1]
Executor(to Assistant): You turn on the desk lamp 1.
Assistant(to Executor): THOUGHT [Now that the desk lamp is turned
on, I can use it to look at the bowl (1) that I found earlier.]
ACTION [look at bowl 1 under the desk lamp 1]
Executor(to Assistant): You turn on the desk lamp 1.
(Assistant falls into an infinite loop turning on desk lamp.)
Executor(to Assistant): Task failed. Reply TERMINATE.
Assistant(to Executor): TERMINATE
```



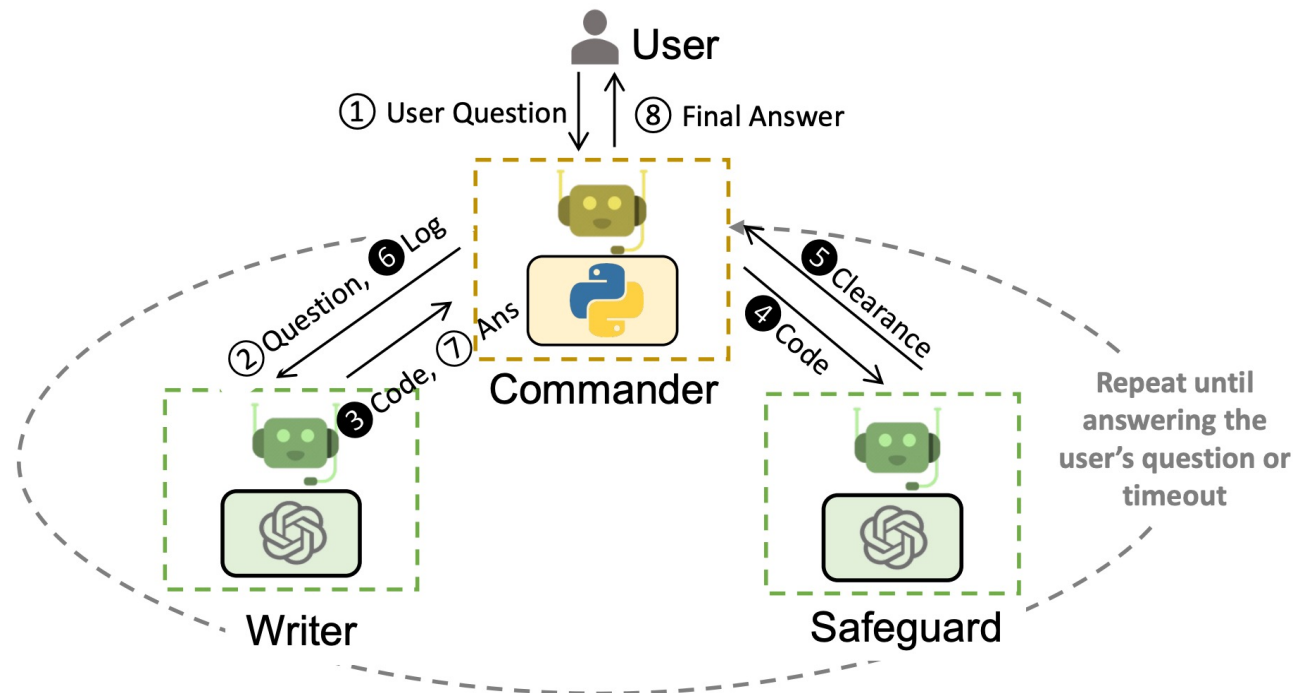
### (b) Three agents

```
Assistant(to Executor): THOUGHT [First, I need to find a bowl. A
bowl is more likely to appear in desk (1-2), drawer (1-2), shelf
(1-3), bed (1). Then I need to find and use a desk lamp.] ACTION
[go to desk 1]
(Assistant finds the desk lamp at desk 1, then goes to desk 2.)
(Assistant falls into the same loop as (a))
GroundingAgent(to Executor): You must find and take the object
before you can examine it. You must go to where the target
object is before you can use it.
Executor(to Assistant): You turn on the desk lamp 1. Hint: You must
find and take the object before you can examine it.
Assistant(to Executor): THOUGHT [I need to find and take the
bowl (1) before I can examine it. Let me find the bowl first.]
ACTION [go to desk 2]
Executor(to Assistant): On the desk 2, you see a bowl 1, a cd
3...
Assistant(to Executor): ACTION [take bowl 1 from desk 2]
Executor(to Assistant): You pick up bowl 1 from desk 2.
Assistant(to Executor): ACTION [go to desk 1]
Assistant(to Executor): ACTION [use desk lamp 1]
Executor(to Assistant): Task success. Reply TERMINATE.
Assistant(to Executor): TERMINATE
```



# Red-teaming and Blue-teaming as a System of agents!

- One agent writes code, one agent safeguards against code
- Repeat until Commander is satisfied



# Advanced Topics

Dynamic Flow!

# Turn Order

- In a conversation, important to have only one person speaking at one time
- How do we decide whose turn to speak, and whether to skip the agents when not required?

# Solution: Manager aka Flow Regulator

- Can either be turn-based (cycle through agents deterministically), or task-based (manager selects agent to respond based on current context)

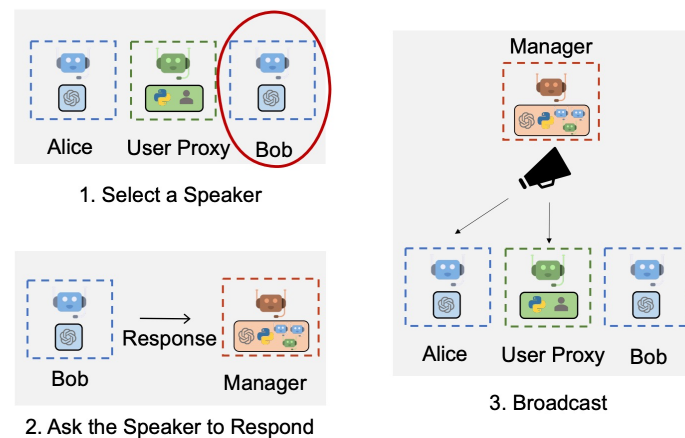


Figure 12: A5: Dynamic Group Chat: Overview of how AutoGen enables dynamic group chats to solve tasks. The Manager agent, which is an instance of the `GroupChatManager` class, performs the following three steps—select a single speaker (in this case Bob), ask the speaker to respond, and broadcast the selected speaker's message to all other agents

# More Agents = Better?

- Generally, more agents can help improve performance (to a limit)
- **Task-based speaker selection performs poorer** – LLMs may not know how to select based on tasks well enough
- More agents = more context = more history
  - LLMs fail at long context

Table 5: Number of successes on the 12 tasks (higher the better).

Model	Two Agent	Group Chat	Group Chat with a task-based speaker selection policy
GPT-3.5-turbo	8	<b>9</b>	7
GPT-4	9	<b>11</b>	8

Table 6: Average # LLM calls and number of termination failures on the 12 tasks (lower the better).

Model	Two Agent	Group Chat	Group Chat with a task-based speaker selection policy
GPT-3.5-turbo	9.9, 9	5.3, 0	4, 0
GPT-4	6.8, 3	4.5, 0	4, 0

# Self-correcting code with a suite of agents

## (b) group chat

**Admin:**

Welcome to the group chat! Work together to resolve my task. I'll reply [TERMINATE] when conversation ended.

**Admin:**

Critic, if code running fail, you ask engineer to rewrite code.

**Admin:**

engineer, you write python code step by step to resolve my task.

**Admin:**

executor, you run python code from engineer and report bug

**Admin:**

Write a script to download all the pdfs from arxiv in last three days and save them under /arxiv folder.

**Engineer:**

Part 1:

```
```sh
pip install arxiv
pip install requests
... (rest of code)
```
```

**Executor:**

exitcode: 1 (execution failed)

Code output:

Collecting arxiv

// pip install log ...

Traceback (most recent call last):

File "", line 15, in <module>

results = arxiv.query(query=query, max\_results=100)

^^^^^^^^^^

AttributeError: module 'arxiv' has no attribute 'query'

**Critic:**

It seems like there is an issue with the 'arxiv' module, as it does not have the 'query' attribute. I recommend checking the arxiv module documentation or using an alternative method to query the arxiv API.

**Engineer:**

I apologize for the confusion. It seems I made a mistake in using the 'arxiv' module. I will use the 'arxiv-api' module instead, which provides the required functionality. Please find the updated code below.

```
```sh
```

(updated code)

**Executor:**

... (pip log)

Traceback (most recent call last):

File "", line 3, in <module>

import arxiv\_api

ModuleNotFoundError: No module named 'arxiv\_api'

**Critic:**

I apologize for the confusion. It seems that there is no package named 'arxiv-api' available. I made a mistake in my previous response. Let's revert to the 'arxiv' package and modify the code accordingly.

... (modified code)

**Executor:**

exitcode: 0 (execution)

**Admin:**

[Terminate]

# Pseudo Multi-Agent Conversation



- In addition to conversation with predefined flow, AutoGen also supports **dynamic conversation flows with multiple agents** (indirectly – there is still a predefined flow to keep, but you can branch separate processes)
- AutoGen provides two general ways to achieve this:
  - **1) Customized generate reply function:** within the customized generate reply function, one agent can hold the current conversation while invoking conversations with other agents depending on the content of the current message and context.
  - **2) Function calls:** In this approach, LLM decides whether or not to call a particular function depending on the conversation status. By messaging additional agents in the called functions, the LLM can drive dynamic multi-agent conversation




# Recap ChatDev Presentation:


## Is this really necessary?


Designing: Programming Language


: CEO (Chief Executive Officer) : CTO (Chief Technology Officer)


---

: To satisfy the new user's demand and make a desktop application realizable, you should propose a concrete programming language.

: Python with the Pygame library is a suitable choice for implementing a basic Gomoku game.

: Great! Python is a versatile and widely-used programming language that will be suitable for implementing a basic Gomoku game. Its simplicity and readability make it a popular choice for beginners and experienced developers alike.

: Let's get started with the development of the basic Gomoku game in Python using the Pygame library!

: <END> Python.

ChatDev

⚡ Default (GPT-3.5)

C

I want to create a Gomoku game.

Output the following json without explanation

```
{'programming language': 'programming language to create it',  
'modules': 'corresponding modules to import'}
```



```
{  
  "programming language": "Python",  
  "modules": "pygame"  
}
```

Mine

My Critique on “Communicative Agents for Software Development. Qian et al. 2023.”

# My Takeaways from AutoGen

- More agents may not be good – **context length should be short** for it to work too
- **Modular agents to do different tasks** is a good idea
- Conversation-style is **not necessary** if we already have structured approaches

Live Demo (by Crit)

# Questions to Ponder

- Is the conversation interface the most efficient at getting things done?
- Should the number of agents be flexibly designed to be 2 or more depending on the situation, or just limited to 2 as per the CAMEL framework used in ChatDev?
  - Communicative Agents for “Mind” Exploration of Large Language Model Society (CAMEL) (Li et al, 2023)
- How to carry over learnings of one agent to another instance?