AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversations

Qingyun Wu, Gagan Bansal, Chi Wang et al.

COLM 2024

Fanglei Shu

Outline

- ***Background***
- ***Design***
- ***Evaluation***
- ***Conclusion***

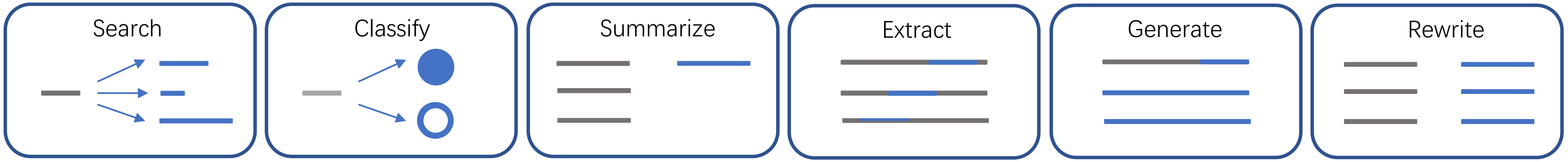
Outline

- **Background**
- Design
- Evaluation
- Conclusion

Background

Large Language Model

LLM Capability



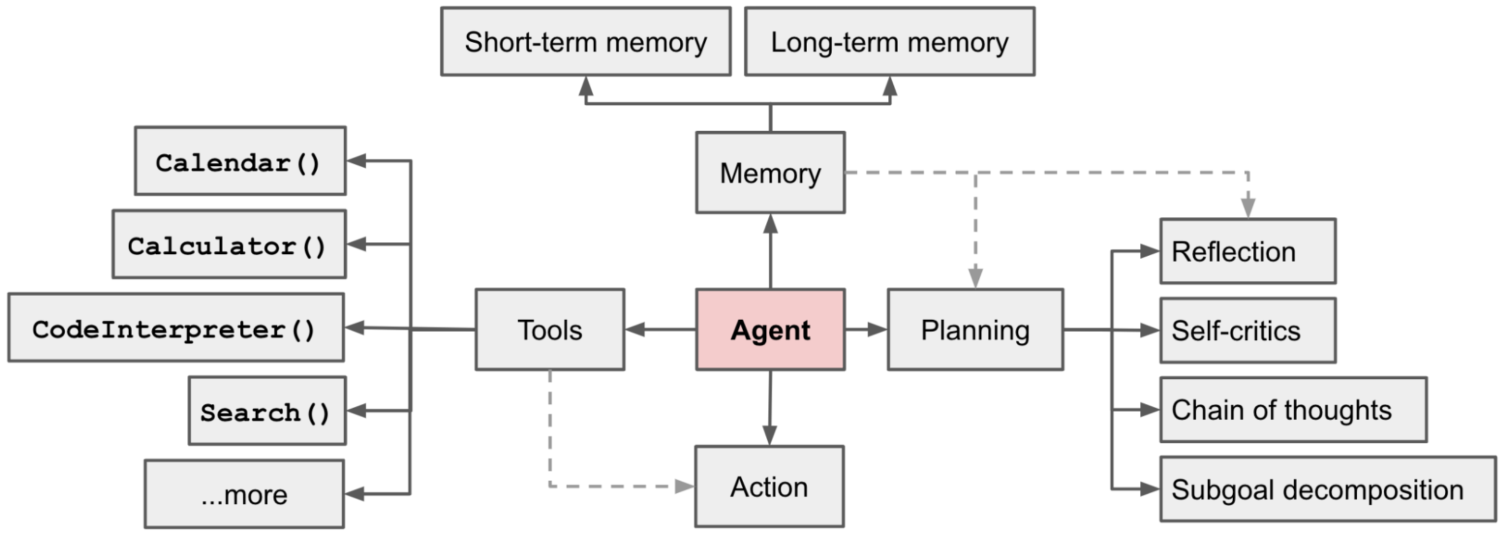
Strong semantic understanding and text generation abilities

Background

LLM Limitation

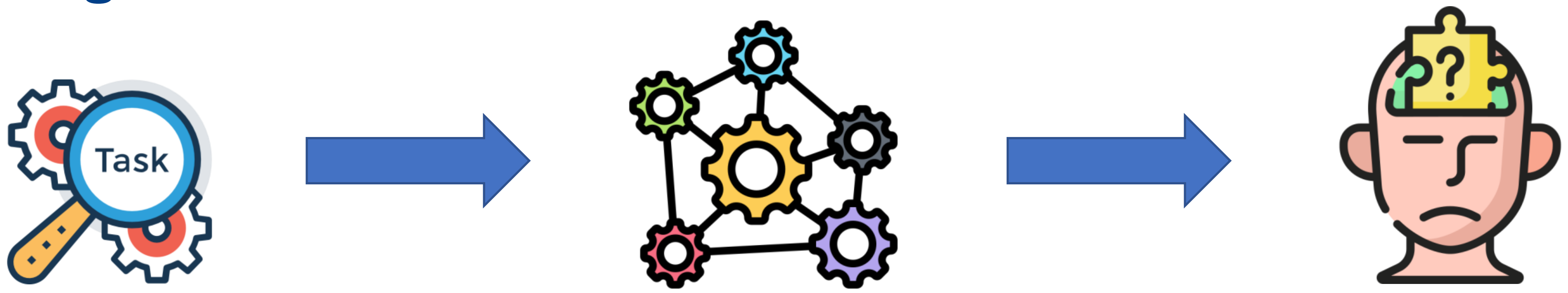
Hallucination, Limited logical reasoning ability, Lack of action ability

LLM Agent

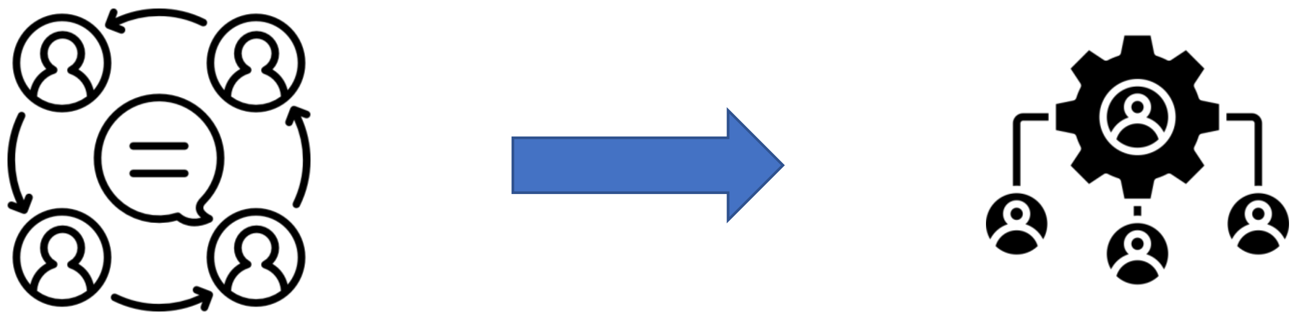


Background

Single-Agent



Multi-Agent



How can we facilitate the development of LLM applications using a multi-agent approach ?

Background

Related work

- **CAMEL [Neurips ’23]**
  - Let chat agents communicate with each other for task completion
  - Fixed workflow pattern and Support two or three agents
- **Agentverse [ICLR ’24]**
  - Dynamic orchestrate a collaborative group of expert agents
  - A sequence of pre-defined stages
- **Metagpt [ICLR ’24] & ChatDev [arXiv ’22]**
  - Software engineering tasks
  - Only support certain multi-agent structures

Specific scenarios or problem → Limited flexibility and generalizability

Background

AutoGen

- A multi-agent dialogue framework with universal abstraction and effective implementation, flexible to meet different application needs.

Challenge

- How to design individual agents in multi-agent collaboration?
- How can we develop a straightforward, unified interface that accommodates a wide range of agent conversation patterns?

Design

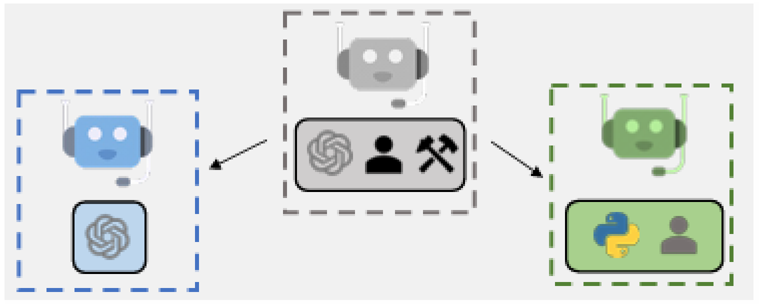
- Customizable and conversable agents
- Conversation programming

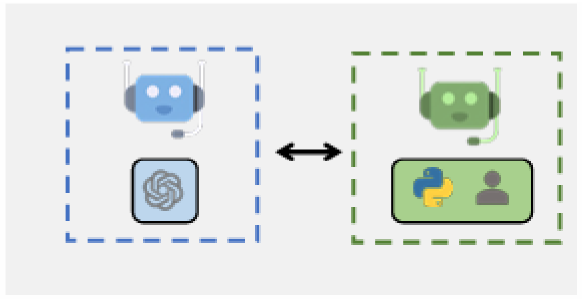
Outline

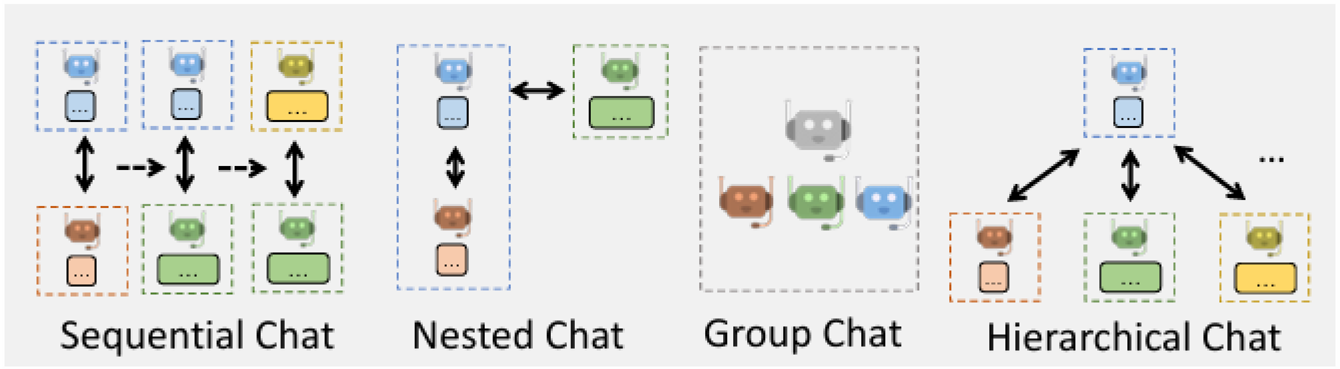
- Background
- **Design**
- Evaluation
- Conclusion

Design

AutoGen


Agent Customization

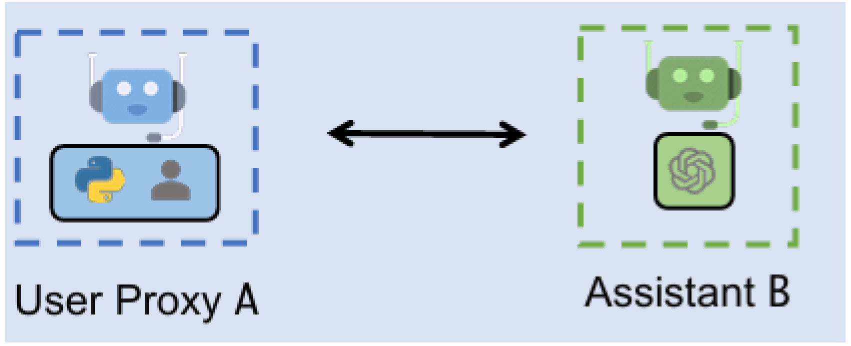

Multi-Agent Conversations

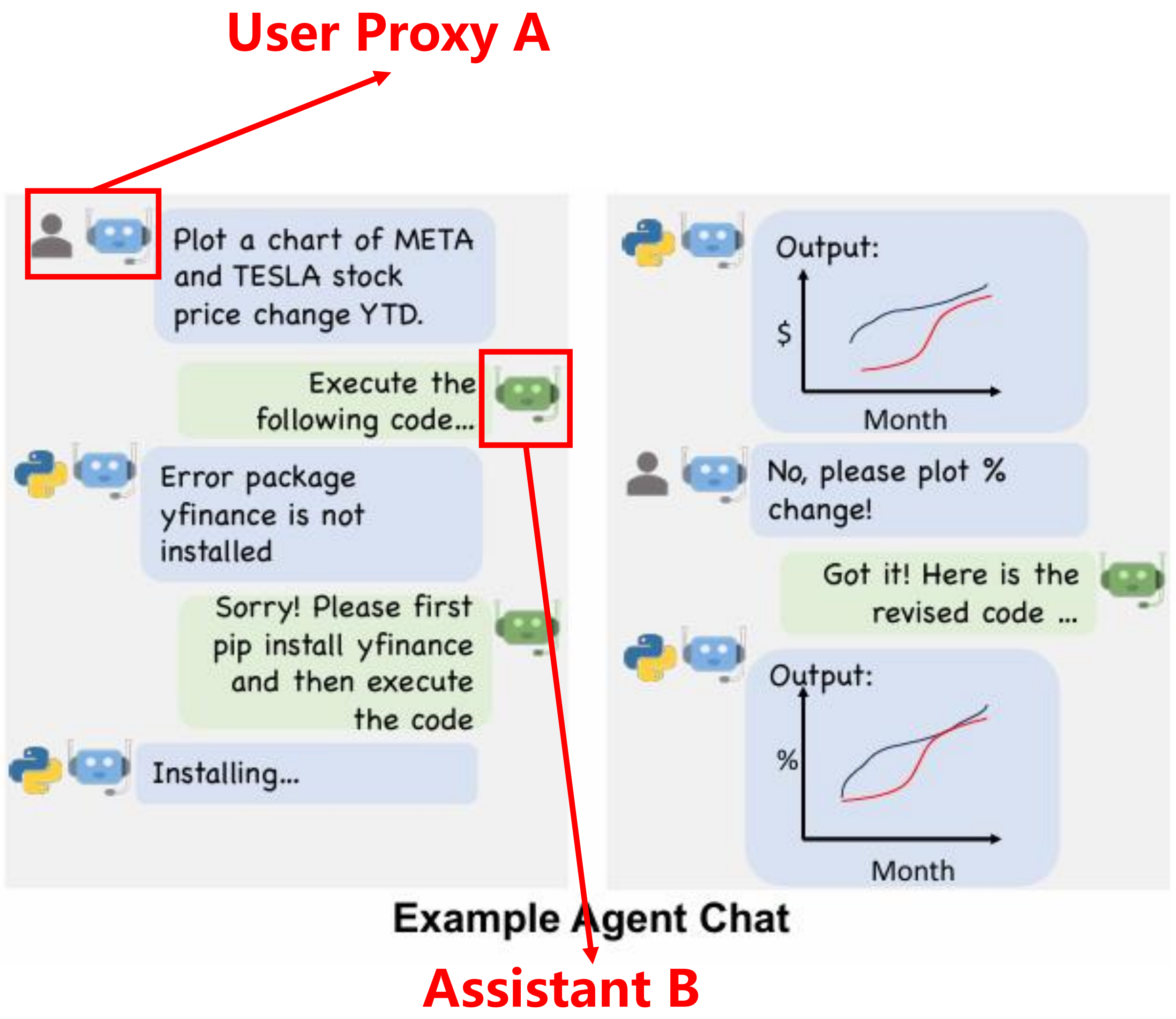


Flexible Conversation Patterns

Design

Overview





Plot a chart of META and TESLA stock price change YTD.

Execute the following code...

Error package yfinance is not installed

Sorry! Please first pip install yfinance and then execute the code

Installing...

Output:

No, please plot % change!

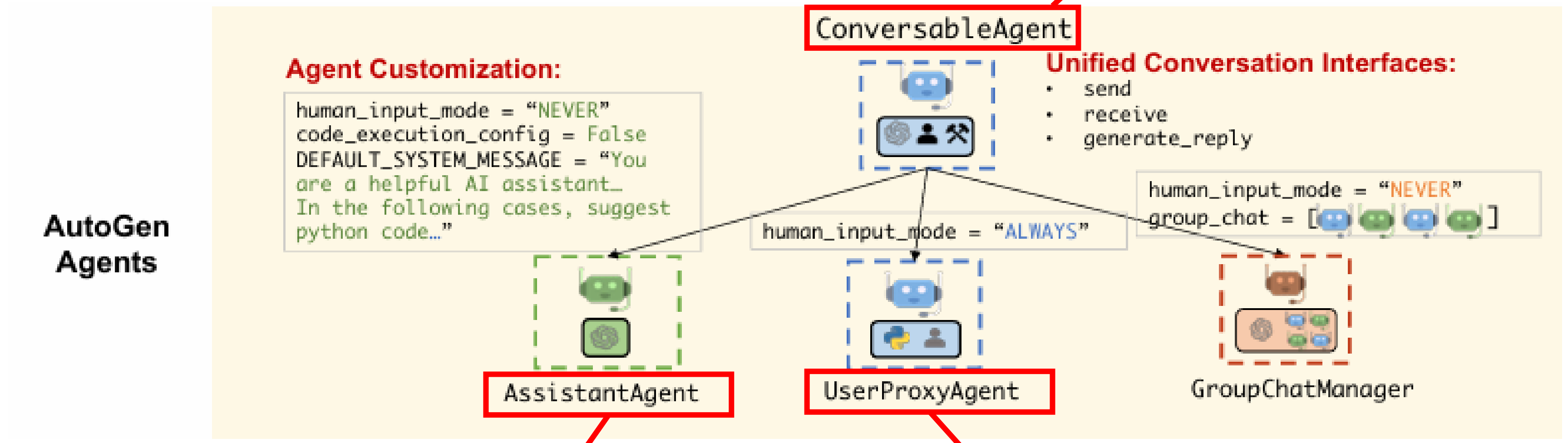
Got it! Here is the revised code ...

Output:

Example Agent Chat

Design

Conversable Agents



AutoGen Agents
ConversableAgent
Agent Customization:
human_input_mode = "NEVER"
code_execution_config = False
DEFAULT_SYSTEM_MESSAGE = "You are a helpful AI assistant… In the following cases, suggest python code…"
Unified Conversation Interfaces:
• send
• receive
• generate_reply
human_input_mode = "ALWAYS"
human_input_mode = "NEVER"
group_chat = [ ]
AssistantAgent
UserProxyAgent
GroupChatManager

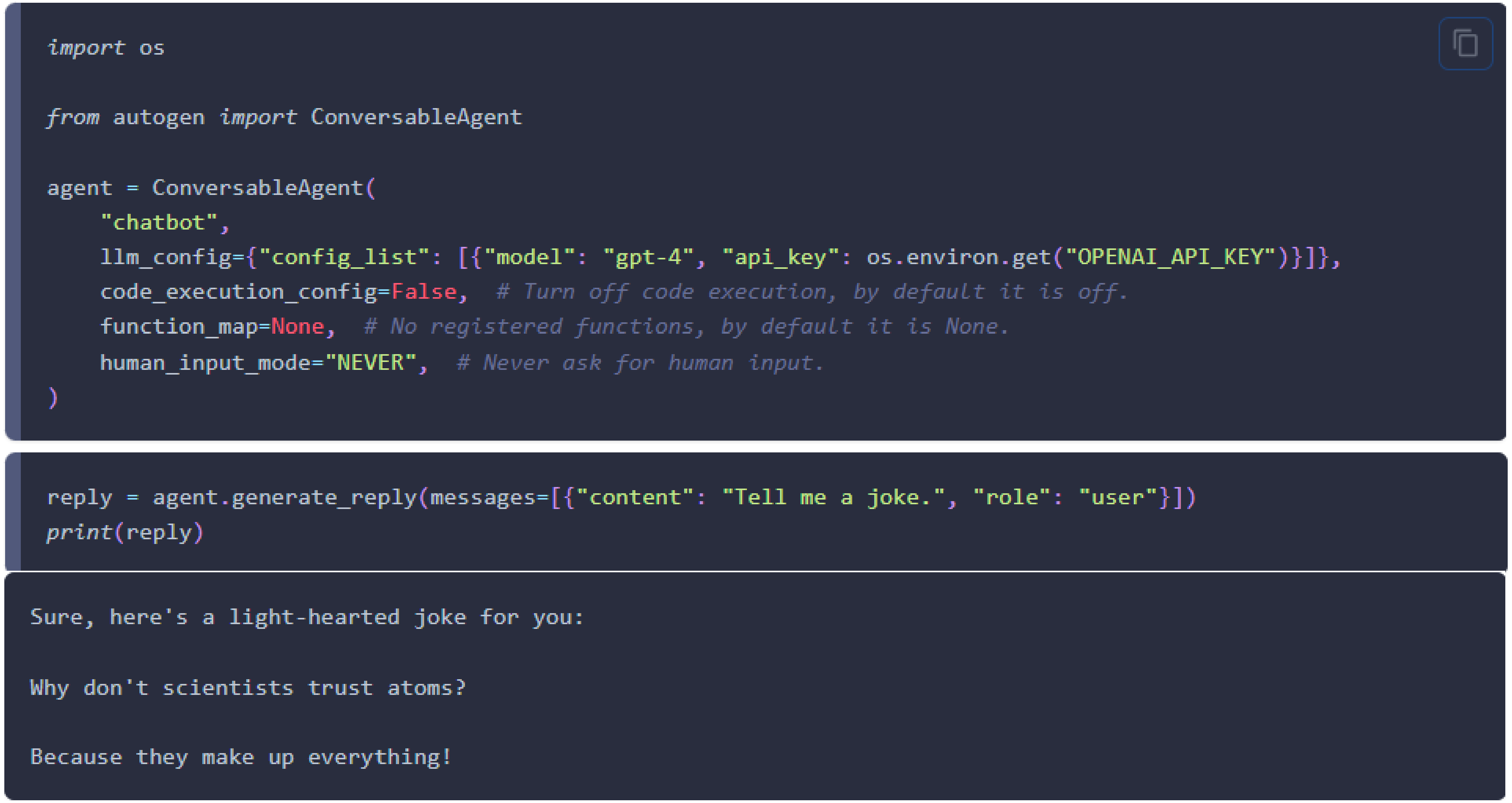
Basic entity

Powered by LLM

Powered by User and Tools

Design

Conversable Agents



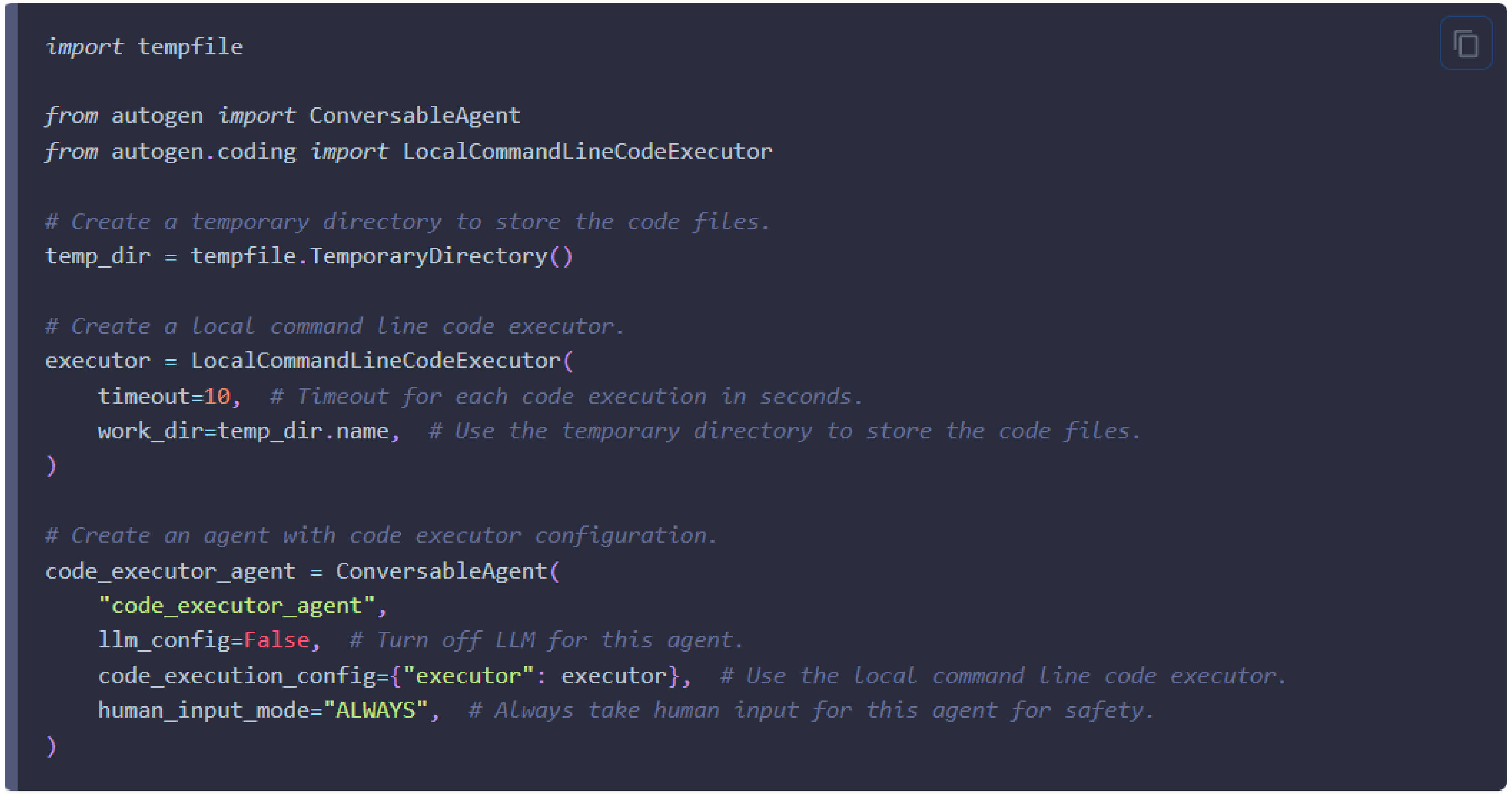
```python
import os

from autogen import ConversableAgent

agent = ConversableAgent(
    "chatbot",
    llm_config={"config_list": [{"model": "gpt-4", "api_key": os.environ.get("OPENAI_API_KEY")}]},
    code_execution_config=False,  # Turn off code execution, by default it is off.
    function_map=None,  # No registered functions, by default it is None.
    human_input_mode="NEVER",  # Never ask for human input.
)
```

```python
reply = agent.generate_reply(messages=[{"content": "Tell me a joke.", "role": "user"}])
print(reply)
```

```
Sure, here's a light-hearted joke for you:

Why don't scientists trust atoms?

Because they make up everything!
```

Design

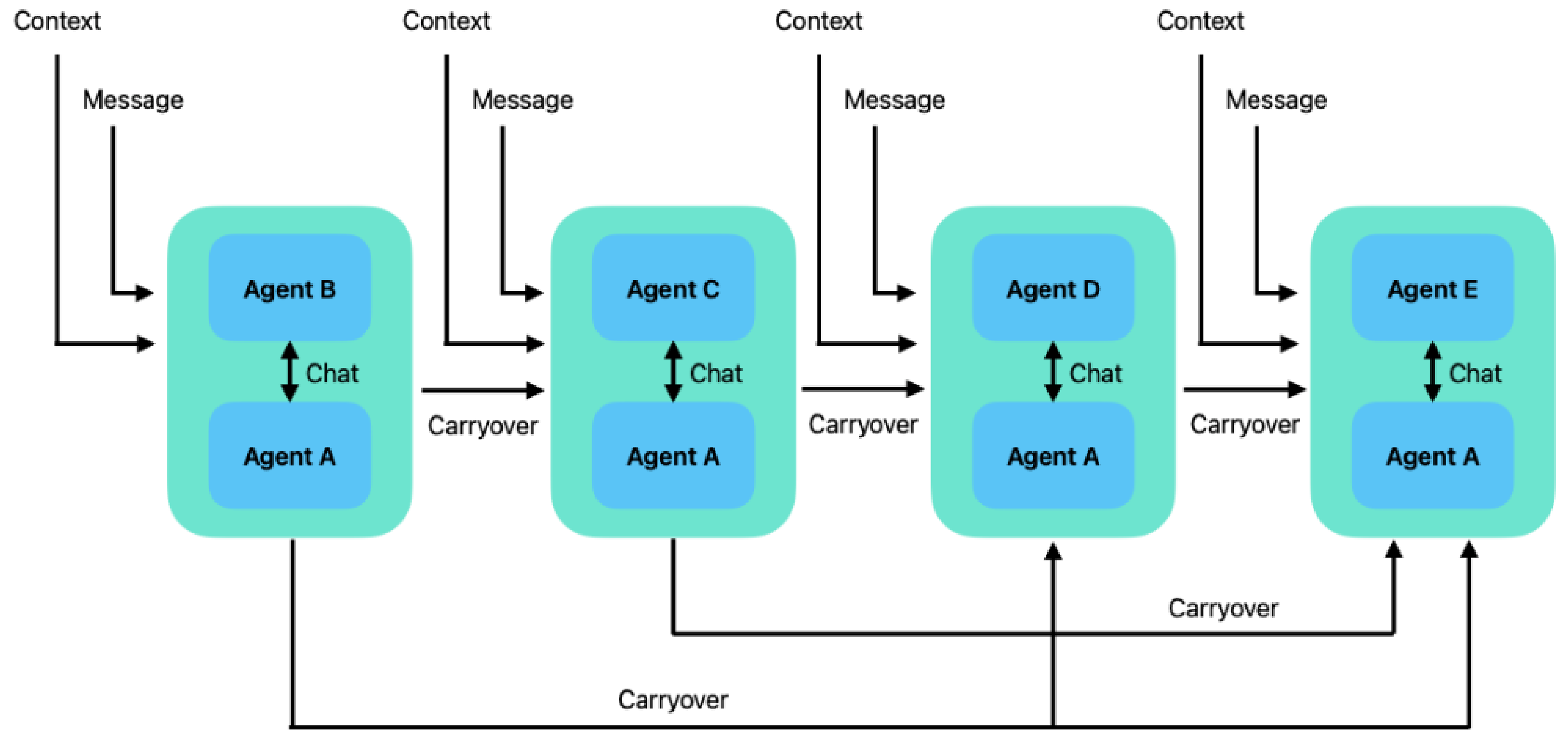
Conversable Agents



```python
import tempfile

from autogen import ConversableAgent
from autogen.coding import LocalCommandLineCodeExecutor

# Create a temporary directory to store the code files.
temp_dir = tempfile.TemporaryDirectory()

# Create a local command line code executor.
executor = LocalCommandLineCodeExecutor(
    timeout=10,  # Timeout for each code execution in seconds.
    work_dir=temp_dir.name,  # Use the temporary directory to store the code files.
)

# Create an agent with code executor configuration.
code_executor_agent = ConversableAgent(
    "code_executor_agent",
    llm_config=False,  # Turn off LLM for this agent.
    code_execution_config={"executor": executor},  # Use the local command line code executor.
    human_input_mode="ALWAYS",  # Always take human input for this agent for safety.
)
```

Design

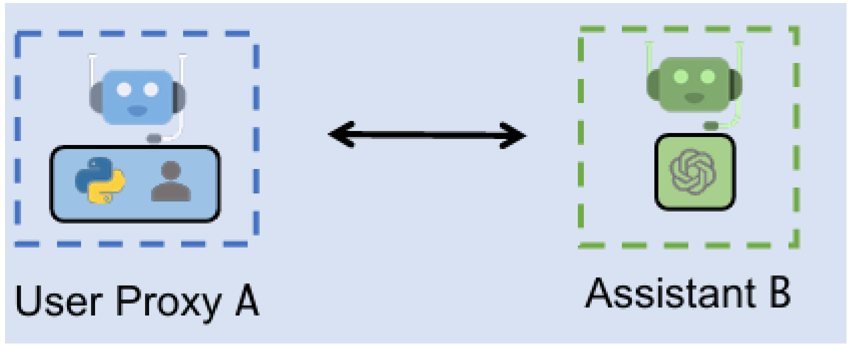
Conversation Programming



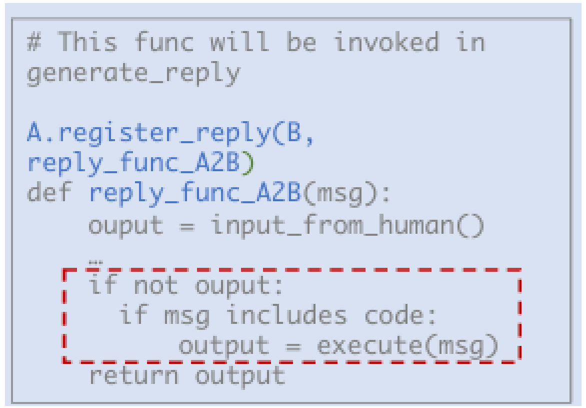
Design

Conversation Programming

功能	描述
send/receive	发送或接收消息。
generate_reply	根据接收到的消息采取行动并生成响应。
register_reply	注册自定义回复函数，以便在聊天流程中使用。



1.Define Agents



```
# This func will be invoked in
generate_reply

A.register_reply(B,
reply_func_A2B)
def reply_func_A2B(msg):
    ouput = input_from_human()
    ...
    if not ouput:
      if msg includes code:
          output = execute(msg)
    return output
```

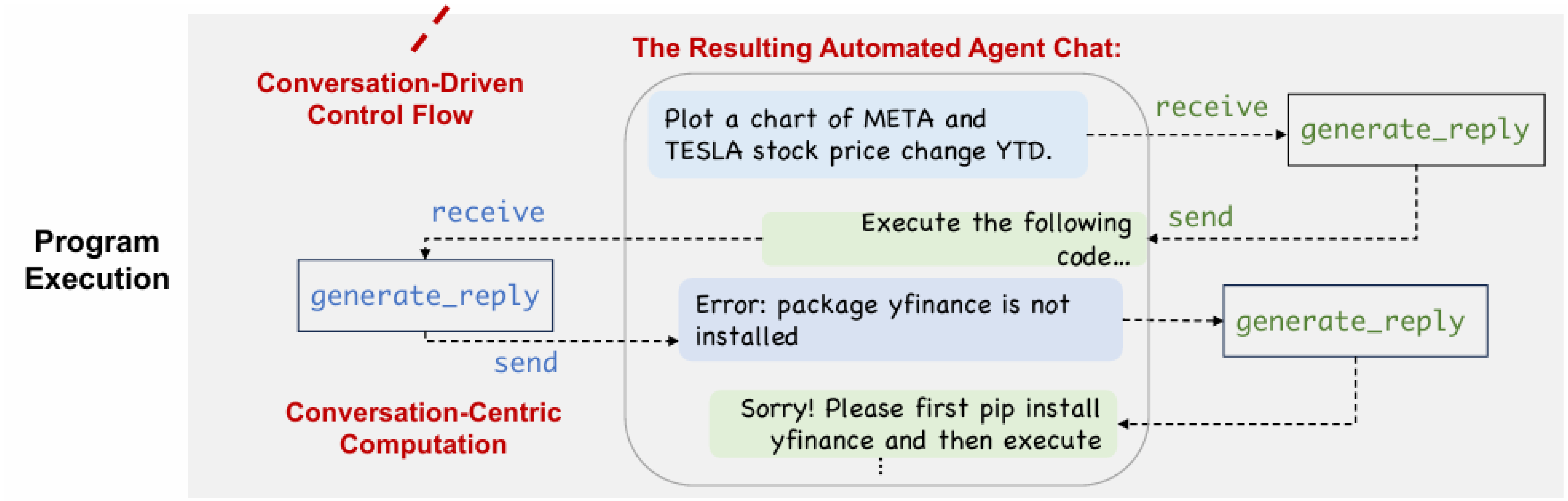
2. Register a Custom Reply Func

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

3. Initiate Conversations

Design

Conversation Programming



Conversation-Driven Control Flow
The Resulting Automated Agent Chat:
Program Execution
receive
generate_reply
send
Conversation-Centric Computation
receive
generate_reply
send
generate_reply

Plot a chart of META and TESLA stock price change YTD.

Execute the following code...

Error: package yfinance is not installed

Sorry! Please first pip install yfinance and then execute

⋮

Termination:

- Initiate_chat(max_turns=3)
- max_consecutive_auto_reply=1
- is_termination_msg=lambda msg: "good bye" in msg["content"].lower()

Design

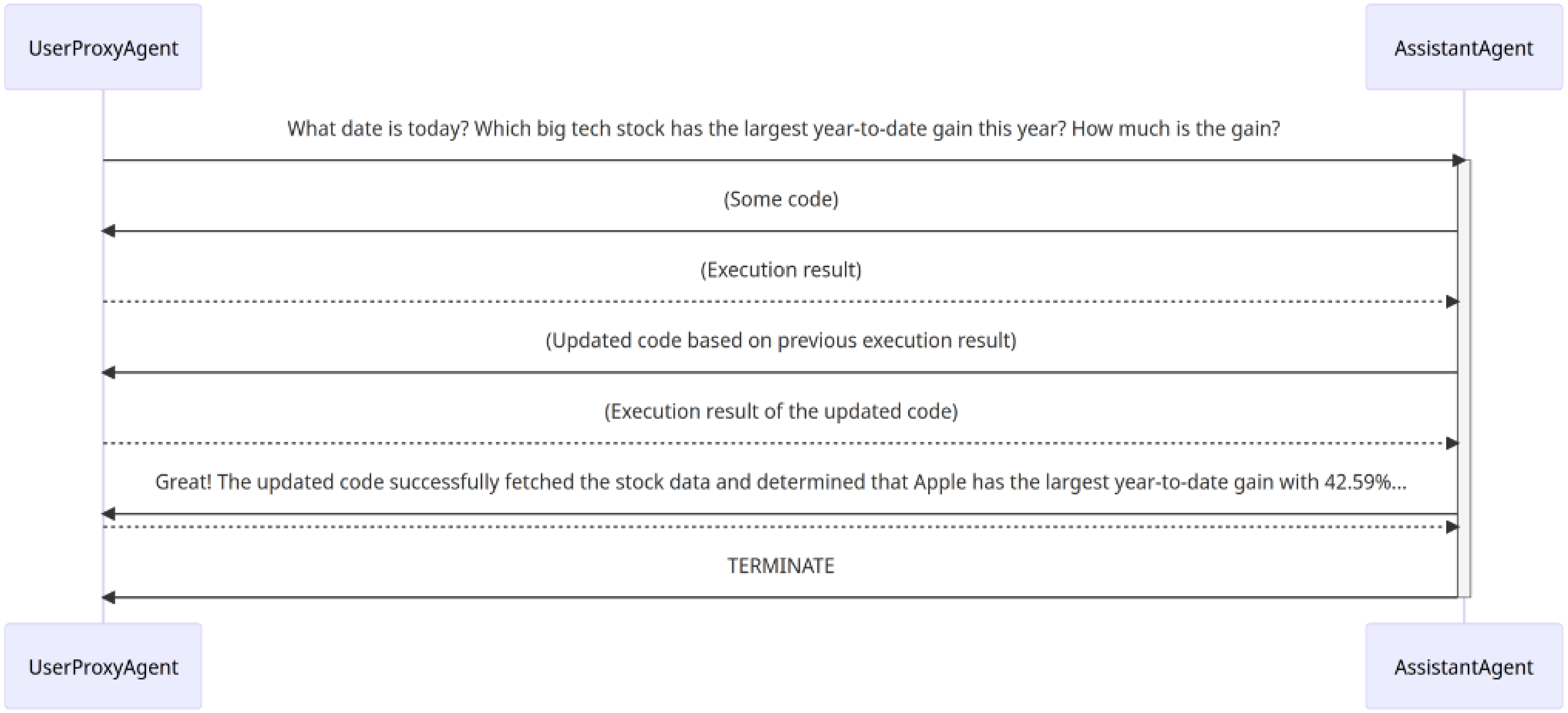
Conversation Programming

```python
import os
from autogen import AssistantAgent, UserProxyAgent
from autogen.coding import DockerCommandLineCodeExecutor

config_list = [{"model": "gpt-4", "api_key": os.environ["OPENAI_API_KEY"]}]

# create an AssistantAgent instance named "assistant" with the LLM configuration.
assistant = AssistantAgent(name="assistant", llm_config={"config_list": config_list})

# create a UserProxyAgent instance named "user_proxy" with code execution on docker.
code_executor = DockerCommandLineCodeExecutor()
user_proxy = UserProxyAgent(name="user_proxy", code_execution_config={"executor": code_executor})
```

```python
# the assistant receives a message from the user, which contains the task description
user_proxy.initiate_chat(
    assistant,
    message="""What date is today? Which big tech stock has the largest year-to-date gain this year? How
much is the gain?""",
)
```

Design

Conversation Programming



UserProxyAgent
AssistantAgent
What date is today? Which big tech stock has the largest year-to-date gain this year? How much is the gain?
(Some code)
(Execution result)
(Updated code based on previous execution result)
(Execution result of the updated code)
Great! The updated code successfully fetched the stock data and determined that Apple has the largest year-to-date gain with 42.59%...
TERMINATE
UserProxyAgent
AssistantAgent

Design

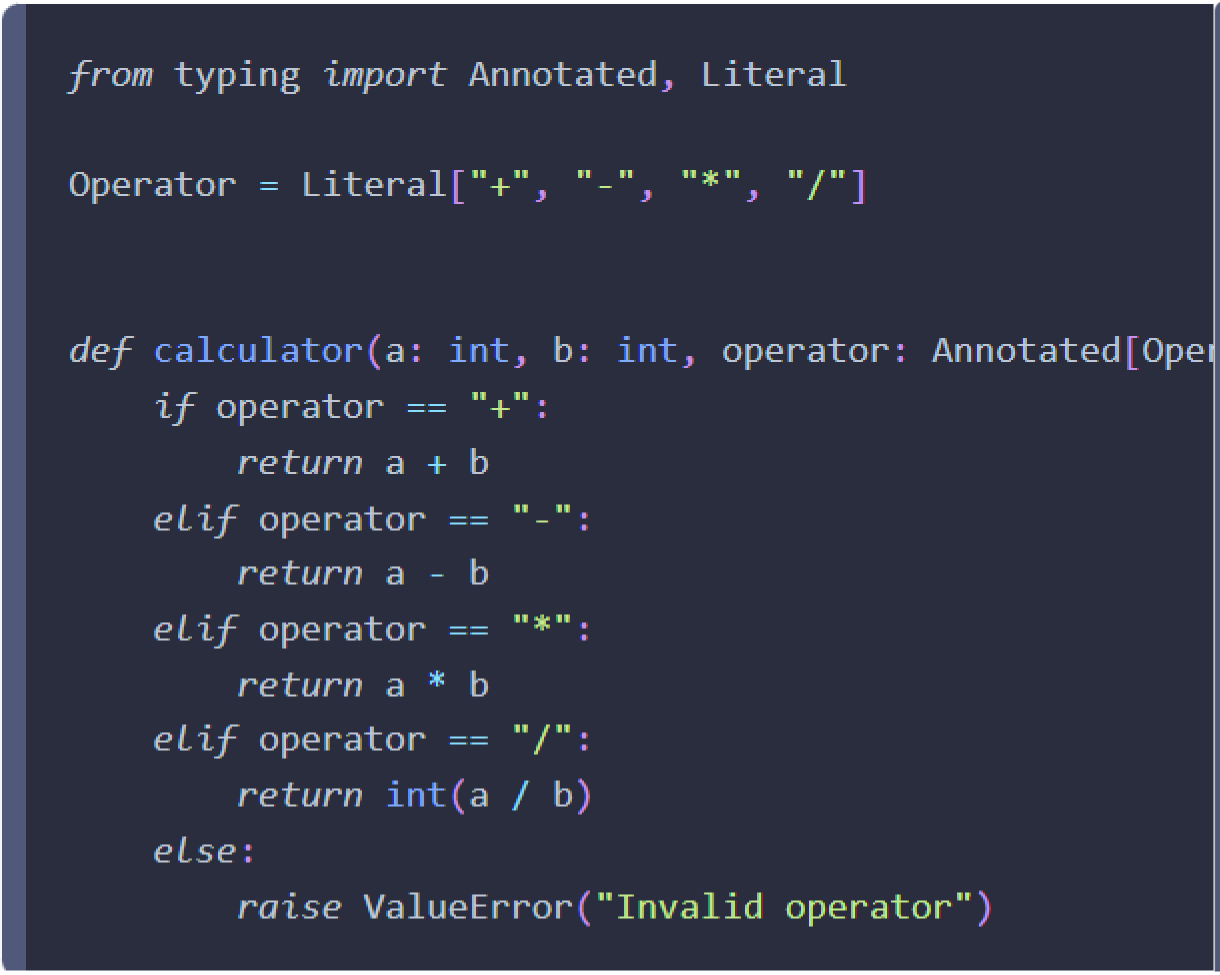
Conversation Programming

Why AutoGen can accommodate a wide range of agent conversation patterns?

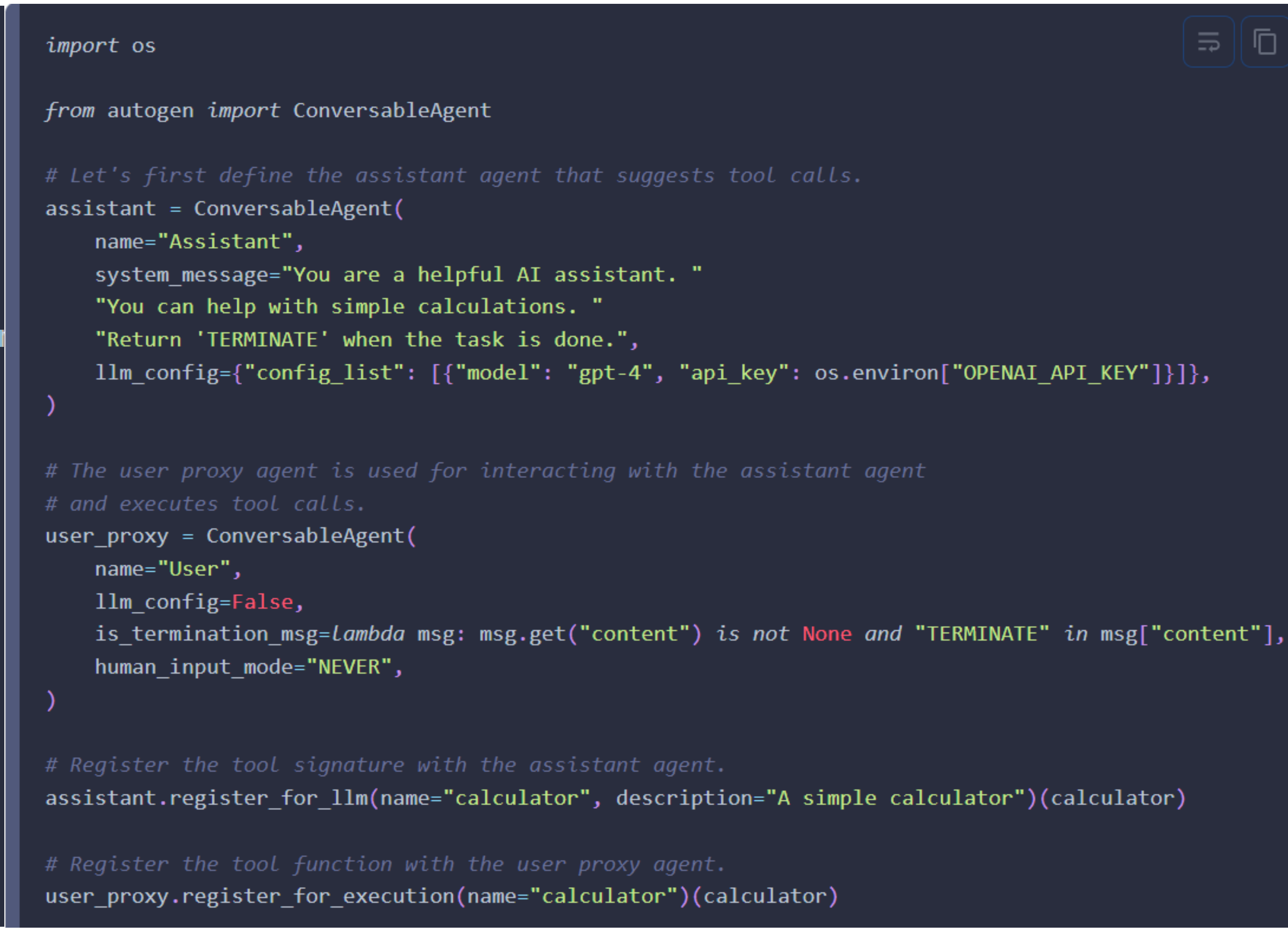
- Custom reply functions and triggers
  - Hold the current conversation & invoking conversations with other agents
- LLM-driven function calls
  - LLM decides whether or not to call a particular function
  - Calling functions to send messages to other agents

Design

Conversable Agents



```python
from typing import Annotated, Literal

Operator = Literal["+", "-", "*", "/"]


def calculator(a: int, b: int, operator: Annotated[Ope
    if operator == "+":
        return a + b
    elif operator == "-":
        return a - b
    elif operator == "*":
        return a * b
    elif operator == "/":
        return int(a / b)
    else:
        raise ValueError("Invalid operator")
```



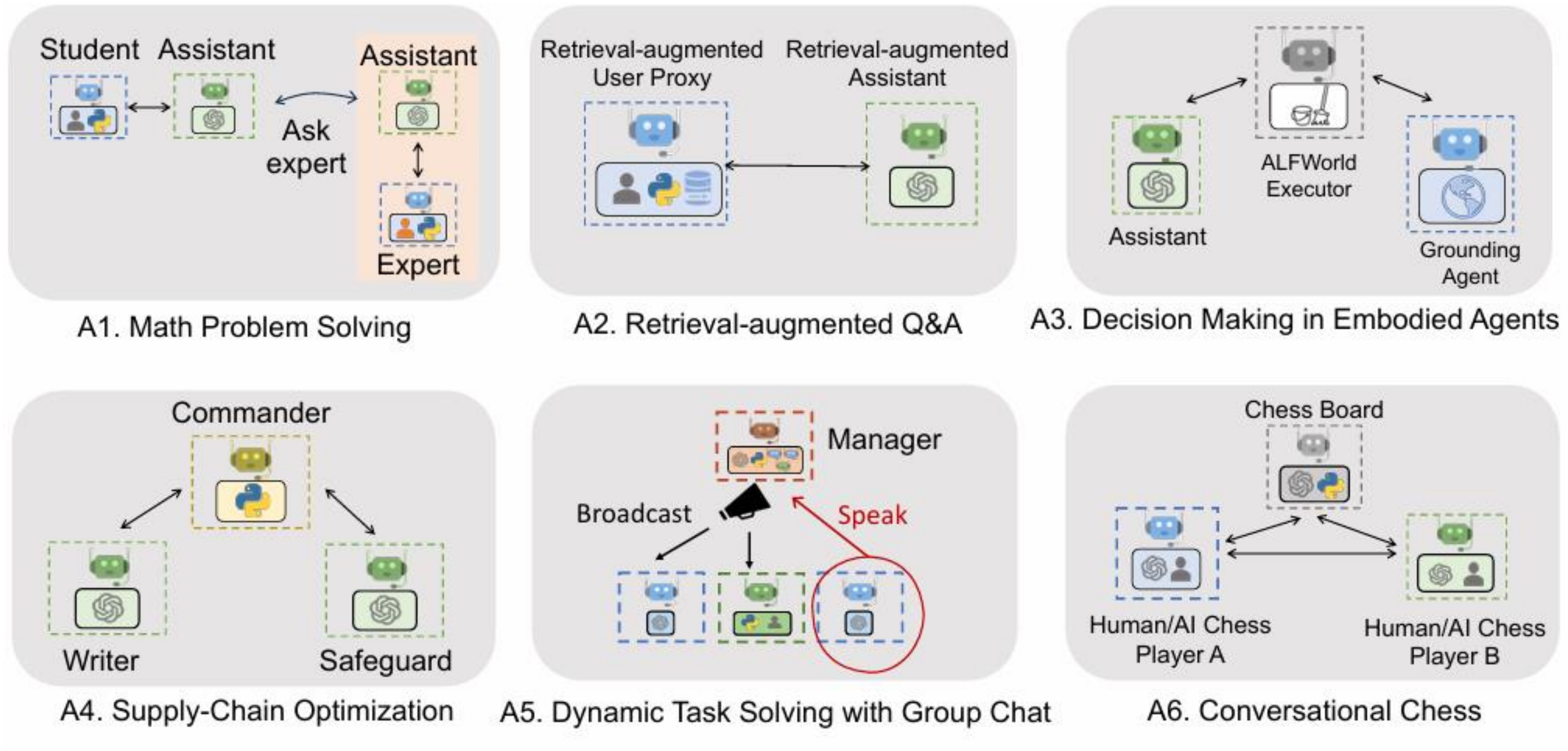
```python
import os

from autogen import ConversableAgent

# Let's first define the assistant agent that suggests tool calls.
assistant = ConversableAgent(
    name="Assistant",
    system_message="You are a helpful AI assistant. "
    "You can help with simple calculations. "
    "Return 'TERMINATE' when the task is done.",
    llm_config={"config_list": [{"model": "gpt-4", "api_key": os.environ["OPENAI_API_KEY"]}]},
)

# The user proxy agent is used for interacting with the assistant agent
# and executes tool calls.
user_proxy = ConversableAgent(
    name="User",
    llm_config=False,
    is_termination_msg=lambda msg: msg.get("content") is not None and "TERMINATE" in msg["content"],
    human_input_mode="NEVER",
)

# Register the tool signature with the assistant agent.
assistant.register_for_llm(name="calculator", description="A simple calculator")(calculator)

# Register the tool function with the user proxy agent.
user_proxy.register_for_execution(name="calculator")(calculator)
```

Outline

- Background
- Design
- **Evaluation**
- Conclusion

Evaluation

Evaluation



Evaluation

Evaluation: Math Problem Solving

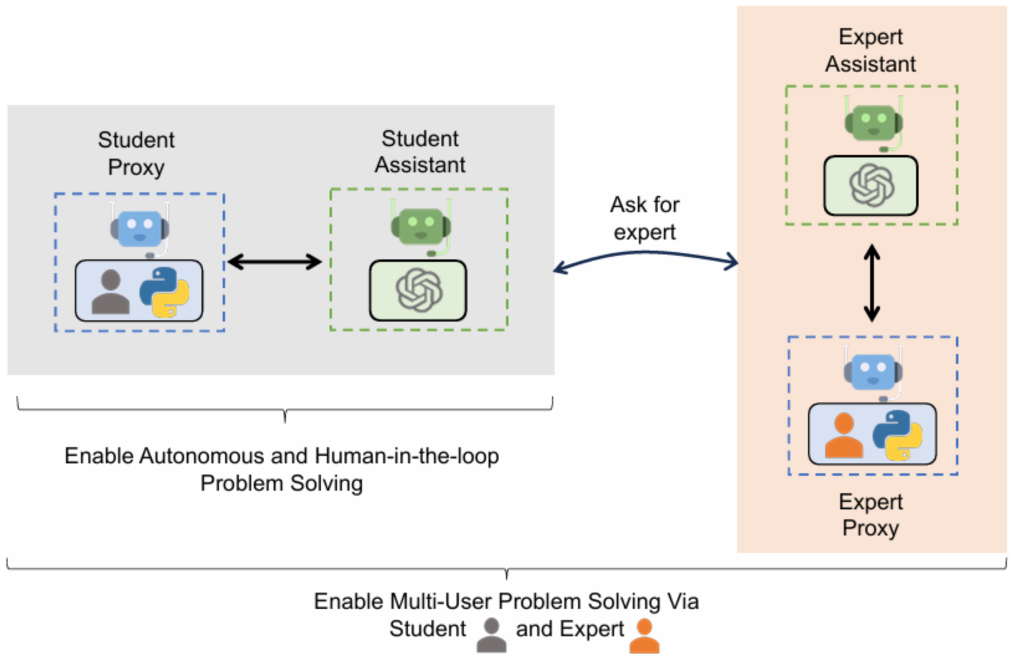
Base model: GPT-4

Baseline:

- ChatGPT+ Plugin
- ChatGPT+Code Interpreter
- LangChain ReAct+Python
- Multi-Agent Debate

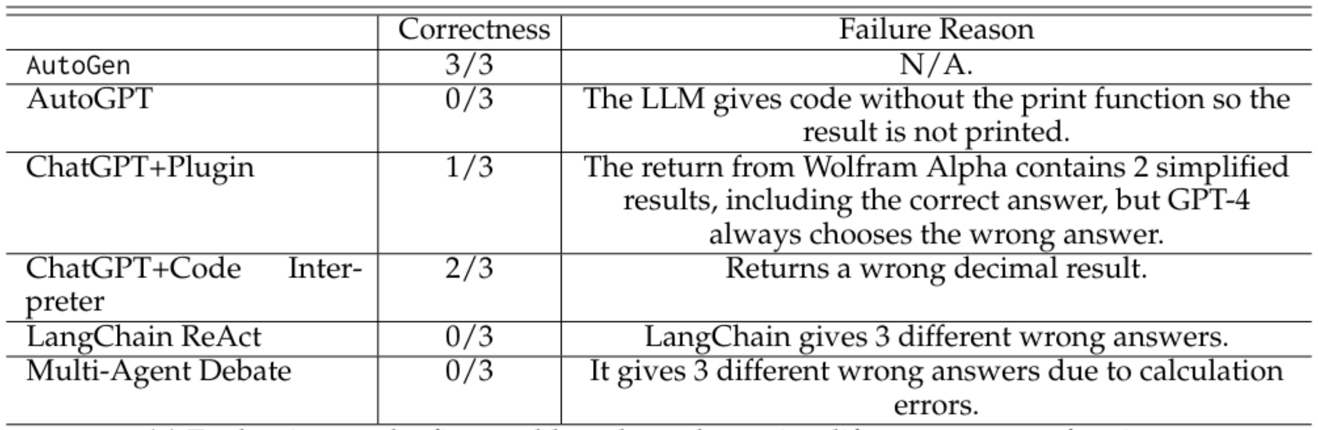
Scenario

- Autonomous Problem Solving
- Human-in-the-loop Problem Solving
- Multi-User Problem Solving


Student Proxy
Student Assistant
Ask for expert
Expert Assistant
Expert Proxy
Enable Autonomous and Human-in-the-loop Problem Solving
Enable Multi-User Problem Solving Via Student and Expert

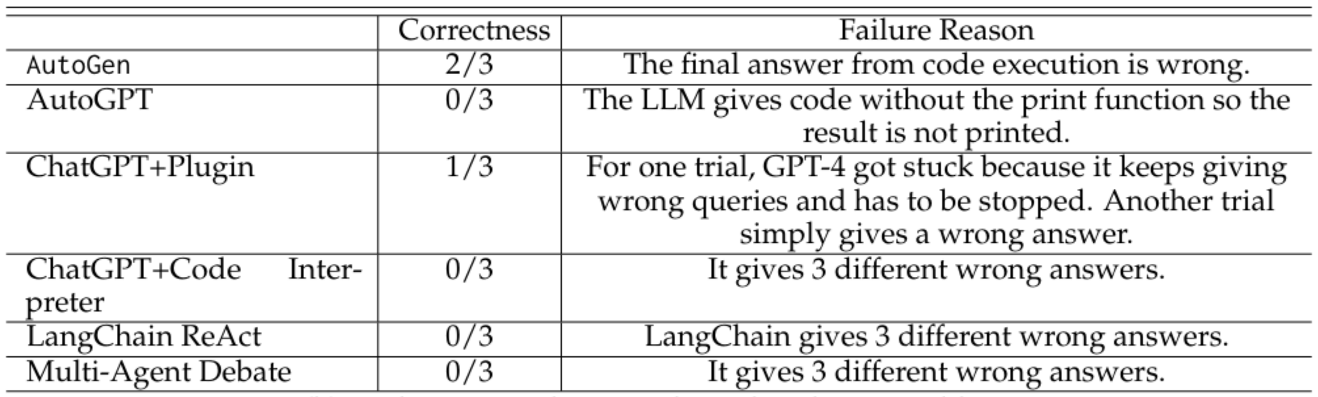
Evaluation

Autonomous Problem Solving



	Correctness	Failure Reason
`AutoGen`	3/3	N/A.
AutoGPT	0/3	The LLM gives code without the print function so the result is not printed.
ChatGPT+Plugin	1/3	The return from Wolfram Alpha contains 2 simplified results, including the correct answer, but GPT-4 always chooses the wrong answer.
ChatGPT+Code Interpreter	2/3	Returns a wrong decimal result.
LangChain ReAct	0/3	LangChain gives 3 different wrong answers.
Multi-Agent Debate	0/3	It gives 3 different wrong answers due to calculation errors.

(a) Evaluation on the first problem that asks to simplify a square root fraction.

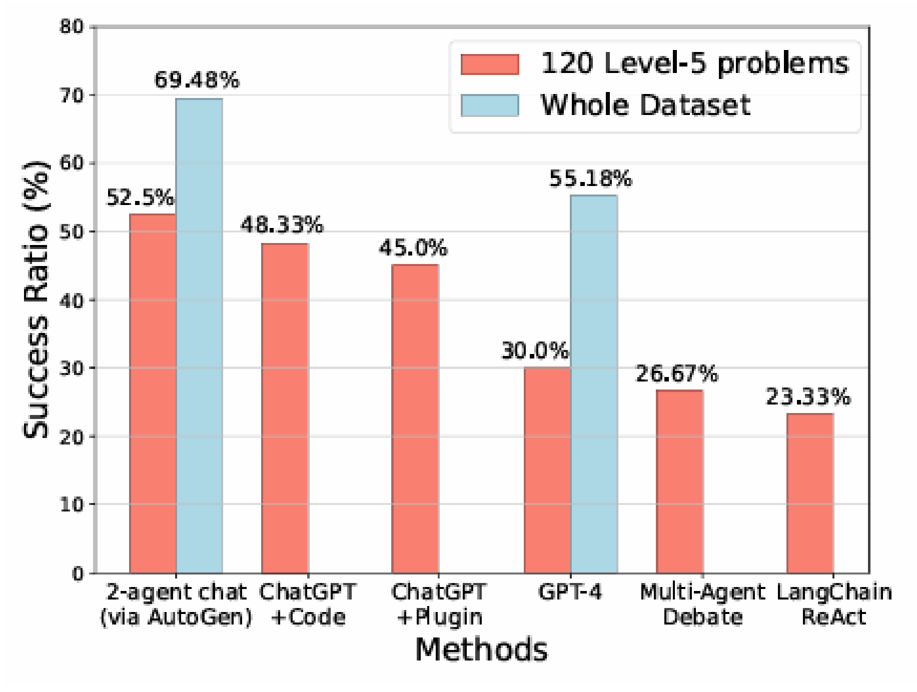


	Correctness	Failure Reason
`AutoGen`	2/3	The final answer from code execution is wrong.
AutoGPT	0/3	The LLM gives code without the print function so the result is not printed.
ChatGPT+Plugin	1/3	For one trial, GPT-4 got stuck because it keeps giving wrong queries and has to be stopped. Another trial simply gives a wrong answer.
ChatGPT+Code Interpreter	0/3	It gives 3 different wrong answers.
LangChain ReAct	0/3	LangChain gives 3 different wrong answers.
Multi-Agent Debate	0/3	It gives 3 different wrong answers.

(b) Evaluation on the second number theory problem.

Evaluation

Autonomous Problem Solving



(a) A1: Performance on MATH (w/ GPT-4).

Evaluation

Human-in-the-loop Problem Solving

$$\begin{cases} 3x-6y+2z+5 = 0 \\ 4x-12y+3z-3 = 0 \end{cases}$$

Find a plane bisects the angle between the two plane, which contains $(-5,-1,-5)$

系统名称	问题解决情况	失败原因
AutoGen	3次成功	-
ChatGPT+Code Interpreter	2次成功，1次失败	失败时未遵循人类提示
ChatGPT+Plugin	2次成功，1次失败	失败时产生了几乎正确的解决方案，但最终答案有符号错误
AutoGPT	3次失败	一次试验中推导出错误的距离方程；另外两次试验中因代码执行错误导致最终答案错误

Outline

- Background
- Design
- Evaluation
- **Conclusion**

Conclusion

An open-source library——AutoGen: paradigms of conversable agents and conversation programming.

- Conversable Agents
  - Define the Single Agent
- Conversation Programming
  - Define the cooperation between Agents

東南大學
SOUTHEAST UNIVERSITY

恳请各位老师与同学批评指正！

汇报人：東方磊

2024.11.10