```
import asyncio
import os

from dotenv import load_dotenv
from loguru import logger
from swarm_models import OpenAIChat
from tickr_agent.main import TickrAgent

from swarms.structs.swarming_architectures import (
    circular_swarm,
    linear_swarm,
    mesh_swarm,
    pyramid_swarm,
    star_swarm,
)

# Load environment variables (API keys)
load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")

# Initialize the OpenAI model
model = OpenAIChat(
    openai_api_key=api_key, model_name="gpt-4", temperature=0.1
)

# Custom Financial Agent System Prompts
```

```
STOCK_ANALYSIS_PROMPT = """
You are an expert financial analyst. Your task is to analyze stock market data for a company
and provide insights on whether to buy, hold, or sell. Analyze trends, financial ratios, and market conditions.
"""

NEWS_SUMMARIZATION_PROMPT = """
You are a financial news expert. Summarize the latest news related to a company and provide insights on
how it could impact its stock price. Be concise and focus on the key takeaways.
"""

RATIO_CALCULATION_PROMPT = """
You are a financial ratio analyst. Your task is to calculate key financial ratios for a company
based on the available data, such as P/E ratio, debt-to-equity ratio, and return on equity.
Explain what each ratio means for investors.
"""

# Example Usage
# Define stock tickers
stocks = ["AAPL", "TSLA"]


# Initialize Financial Analysis Agents
stock_analysis_agent = TickrAgent(
    agent_name="Stock-Analysis-Agent",
```

```
    system_prompt=STOCK_ANALYSIS_PROMPT,
    stocks=stocks,
)

news_summarization_agent = TickrAgent(
    agent_name="News-Summarization-Agent",
    system_prompt=NEWS_SUMMARIZATION_PROMPT,
    stocks=stocks,
)

ratio_calculation_agent = TickrAgent(
    agent_name="Ratio-Calculation-Agent",
    system_prompt=RATIO_CALCULATION_PROMPT,
    stocks=stocks,
)
# Create a list of agents for swarming
agents = [
    stock_analysis_agent,
    news_summarization_agent,
    ratio_calculation_agent,
]

# Define financial analysis tasks
tasks = [
    "Analyze the stock performance of Apple (AAPL) in the last 6 months.",
    "Summarize the latest financial news on Tesla (TSLA).",
```

```
    "Calculate the P/E ratio and debt-to-equity ratio for Amazon (AMZN).",
]


# ------------------------------# Showcase Circular Swarm
# ------------------------------
logger.info("Starting Circular Swarm for financial analysis.")
circular_result = circular_swarm(agents, tasks)
logger.info(f"Circular Swarm Result:\n{circular_result}\n")



# ------------------------------
# Showcase Linear Swarm
# ------------------------------
logger.info("Starting Linear Swarm for financial analysis.")
linear_result = linear_swarm(agents, tasks)
logger.info(f"Linear Swarm Result:\n{linear_result}\n")



# ------------------------------
# Showcase Star Swarm
# ------------------------------
logger.info("Starting Star Swarm for financial analysis.")
star_result = star_swarm(agents, tasks)
logger.info(f"Star Swarm Result:\n{star_result}\n")
```

```
# ------------------------------
# Showcase Mesh Swarm
# ------------------------------
logger.info("Starting Mesh Swarm for financial analysis.")
mesh_result = mesh_swarm(agents, tasks)
logger.info(f"Mesh Swarm Result:\n{mesh_result}\n")


# ------------------------------
# Showcase Pyramid Swarm
# ------------------------------
logger.info("Starting Pyramid Swarm for financial analysis.")
pyramid_result = pyramid_swarm(agents, tasks)
logger.info(f"Pyramid Swarm Result:\n{pyramid_result}\n")


# ------------------------------
# Example: One-to-One Communication between Agents
# ------------------------------
logger.info(
    "Starting One-to-One communication between Stock and News agents."
)
one_to_one_result = stock_analysis_agent.run(
    "Analyze Apple stock performance, and then send the result to the News Summarization Agent"
)
news_summary_result = news_summarization_agent.run(one_to_one_result)
```

```
logger.info(
    f"One-to-One Communication Result:\n{news_summary_result}\n"
)



# ------------------------------
# Example: Broadcasting to all agents
# ------------------------------
async def broadcast_task():
    logger.info("Broadcasting task to all agents.")
    task = "Summarize the overall stock market performance today."
    await asyncio.gather(*[agent.run(task) for agent in agents])



asyncio.run(broadcast_task())



# ------------------------------
# Deep Comments & Explanations
# ------------------------------

"""
Explanation of Key Components:

1. **Agents**:
   - We created three specialized agents for financial analysis: Stock Analysis, News Summarization,
```

and Ratio Calculation.

- Each agent is provided with a custom system prompt that defines their unique task in analyzing stock data.

2. **Swarm Examples**:

- **Circular Swarm**: Agents take turns processing tasks in a circular manner.

- **Linear Swarm**: Tasks are processed sequentially by each agent.

- **Star Swarm**: The first agent (Stock Analysis) processes all tasks before distributing them to other agents.

- **Mesh Swarm**: Agents work on random tasks from the task queue.

- **Pyramid Swarm**: Agents are arranged in a pyramid structure, processing tasks layer by layer.

3. **One-to-One Communication**:

- This showcases how one agent can pass its result to another agent for further processing, useful for complex workflows where agents depend on each other.

4. **Broadcasting**:

- The broadcasting function demonstrates how a single task can be sent to all agents simultaneously. This can be useful for situations like summarizing daily stock market performance across multiple agents.

5. **Logging with Loguru**:

- We use `loguru` for detailed logging throughout the swarms. This helps to track the flow of information and responses from each agent.

"""