```
import os
from swarms import Agent
from swarm_models import OpenAIChat
from swarms.prompts.finance_agent_sys_prompt import (
    FINANCIAL_AGENT_SYS_PROMPT,
)

# Get the OpenAI API key from the environment variable
api_key = os.getenv("OPENAI_API_KEY")

# Create an instance of the OpenAIChat class
model = OpenAIChat(
    api_key=api_key, model_name="gpt-4o-mini", temperature=0.1
)

# Initialize the agent
agent = Agent(
    agent_name="Financial-Analysis-Agent-General-11",
    system_prompt=FINANCIAL_AGENT_SYS_PROMPT,
    llm=model,
    max_loops=1,
    autosave=False,
    dashboard=False,
    verbose=True,
    dynamic_temperature_enabled=True,
    saved_state_path="finance_agent.json",
```

```
    user_name="swarms_corp",
    retry_attempts=3,
    context_length=200000,
    tool_system_prompt=None,
)

# # Convert the agent object to a dictionary
print(agent.to_dict())
print(agent.to_toml())
print(agent.model_dump_json())
print(agent.model_dump_yaml())

# Ingest documents into the agent's knowledge base
agent.ingest_docs("your_pdf_path.pdf")

# Receive a message from a user and process it
agent.receive_message(name="agent_name", message="message")

# Send a message from the agent to a user
agent.send_agent_message(agent_name="agent_name", message="message")

# Ingest multiple documents into the agent's knowledge base
agent.ingest_docs("your_pdf_path.pdf", "your_csv_path.csv")

# Run the agent with a filtered system prompt
agent.filtered_run(
```

```
    "How can I establish a ROTH IRA to buy stocks and get a tax break? What are the criteria?"
)

# Run the agent with multiple system prompts
agent.bulk_run(
    [
        "How can I establish a ROTH IRA to buy stocks and get a tax break? What are the criteria?",
        "Another system prompt",
    ]
)

# Add a memory to the agent
agent.add_memory("Add a memory to the agent")

# Check the number of available tokens for the agent
agent.check_available_tokens()

# Perform token checks for the agent
agent.tokens_checks()

# Print the dashboard of the agent
agent.print_dashboard()

# Fetch all the documents from the doc folders
agent.get_docs_from_doc_folders()
```

```
# Activate agent ops
agent.activate_agentops()
agent.check_end_session_agentops()

# Dump the model to a JSON file
agent.model_dump_json()
print(agent.to_toml())

# Print all of the output metadata of the agent
print(agent.agent_output.model_dump())
print(agent.agent_output.model_dump_json())
```