```
import os

import asyncio

from swarms import Agent

from swarm_models import OpenAIChat

import time

import psutil


from swarms.prompts.finance_agent_sys_prompt import (

    FINANCIAL_AGENT_SYS_PROMPT,

)

from dotenv import load_dotenv


load_dotenv()


# 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(

    openai_api_key=api_key, model_name="gpt-4o-mini", temperature=0.1

)


# Initialize the agent

agent = Agent(

    agent_name="Financial-Analysis-Agent",

    system_prompt=FINANCIAL_AGENT_SYS_PROMPT,
    llm=model,

    max_loops=1,

    autosave=True,

    dashboard=False,

    verbose=True,

    dynamic_temperature_enabled=True,

    saved_state_path="finance_agent.json",

    user_name="swarms_corp",

    retry_attempts=1,

    context_length=200000,

    return_step_meta=False,

    output_type="string",

    streaming_on=False,

)



# Function to measure time and memory usage

def measure_time_and_memory(func):

    def wrapper(*args, **kwargs):

        start_time = time.time()

        result = func(*args, **kwargs)

        end_time = time.time()

        memory_usage = psutil.Process().memory_info().rss / 1024**2

        print(f"Time taken: {end_time - start_time} seconds")

        print(f"Memory used: {memory_usage} MB")

        return result
    return wrapper



# Function to run the agent asynchronously
@measure_time_and_memory
async def run_agent_async():
    await asyncio.gather(
        agent.run(
            "How can I establish a ROTH IRA to buy stocks and get a tax break? What are the criteria"
        )
    )



# Function to run the agent on another thread
@measure_time_and_memory
def run_agent_thread():
    asyncio.run(run_agent_async())



# Run the agent asynchronously and on another thread to test the speed
asyncio.run(run_agent_async())
run_agent_thread()
```