AKSHAT KUMAR

Kumarakshaty2k2@gmail.com

Abstract

This project explores the development of an AI-driven assistant that provides weather information based on user queries. Leveraging the OpenAI API, the assistant is designed to handle user inputs related to weather conditions in various locations. The assistant integrates custom tools for weather retrieval, processes user queries, and responds using predefined weather data for testing purposes. The report outlines the process of creating the assistant, managing user interactions via threads, and handling tool outputs.

OpenAI API GET WEATHER FUNCTION.

Get weather function.

Contents

[**1.** **Business Objective** 2](#_Toc42152049)

[**2.** **Approach** 2](#_Toc42152050)

[**3.** **Detailed Explanation on Algorithms** 2](#_Toc42152051)

[**4.** **Results** 2](#_Toc42152052)

[**5.** **Conclusions** 2](#_Toc42152053)

[**6.** **References** 2](#_Toc42152054)

## **1. Business Objective:**

## **The primary objective of this project is to create an AI-driven assistant capable of providing weather information based on user queries. This assistant integrates with the OpenAI API to leverage its language model capabilities and custom tools for weather retrieval. The system aims to interact with users to provide accurate weather updates and handle various queries effectively**. 2. Approach:

## **The approach involves using the OpenAI API to create a custom assistant that can handle weather-related queries. The key steps are:**

## **API Setup: Installing the OpenAI package and configuring the environment.**

## **Assistant Creation: Defining the tools and creating a new assistant with specific instructions.**

## **User Interaction: Handling user inputs and creating a thread for interaction.**

## **Weather Retrieval: Using predefined weather data to simulate real-time responses.**

## **Response Handling: Managing and processing responses from the assistant, including tool outputs.**

## **Integration: Submitting the tool outputs and retrieving results to provide the final response to the user.**

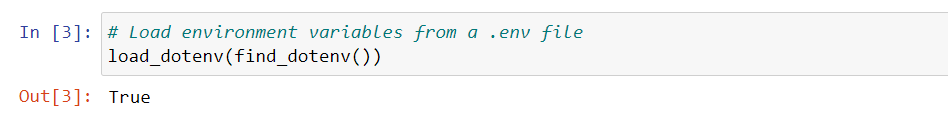
## 3. Detailed Explanation on Algorithms:

## **Code Breakdown:**

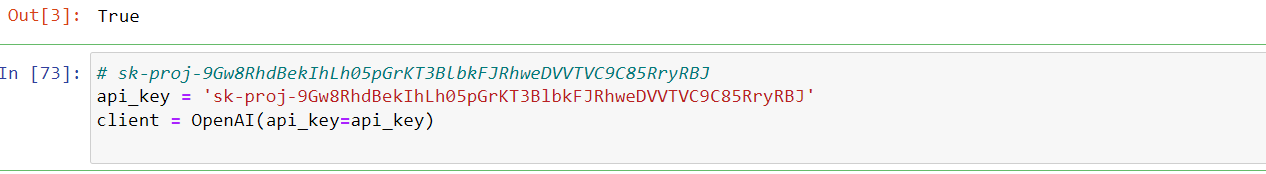
## **Install and Import Libraries :**

## 

* **Purpose: Install the latest OpenAI package and import required libraries.**
* **Explanation: openai is used to interact with the OpenAI API. dotenv helps in loading environment variables from a .env file.**

**2. Load Environment Variables:**

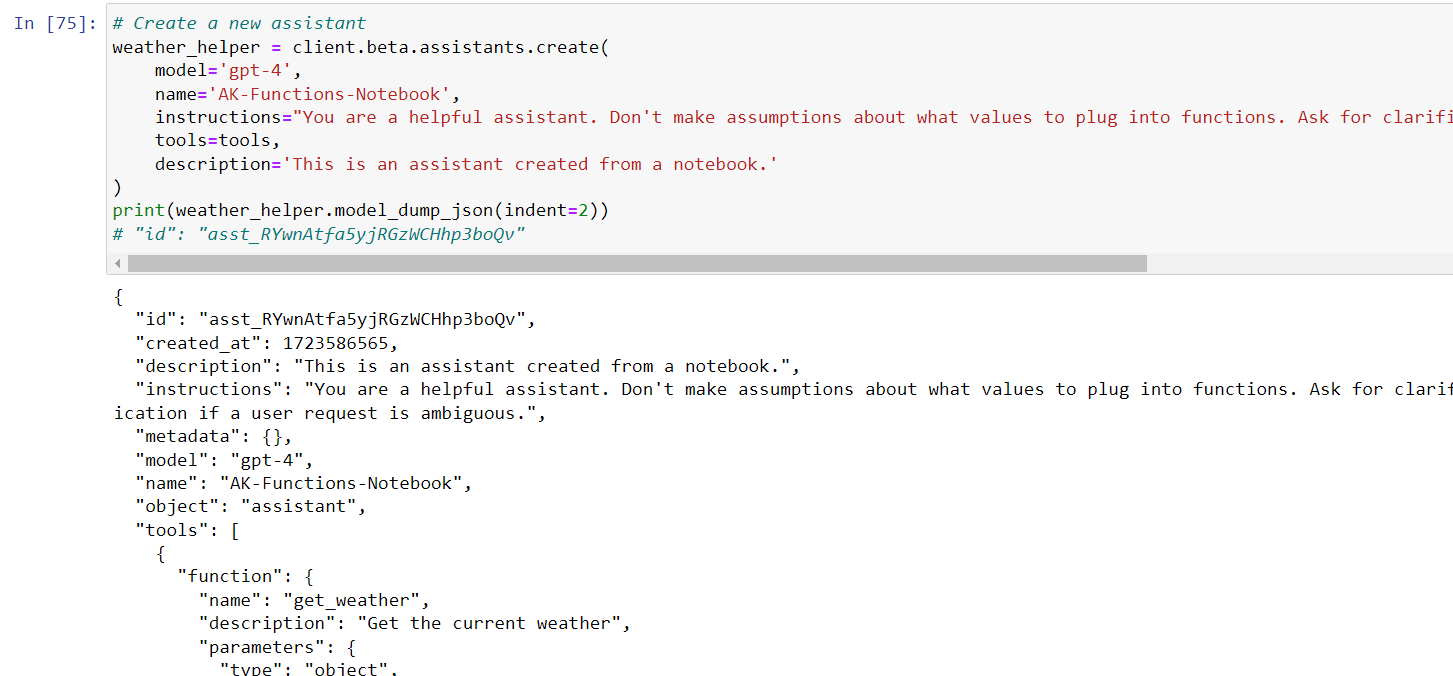
* **Purpose: Load environment variables from a .env file.**
* **Explanation: This allows for secure management of API keys and other sensitive information.**

**3. Initialize OpenAI Client:** 

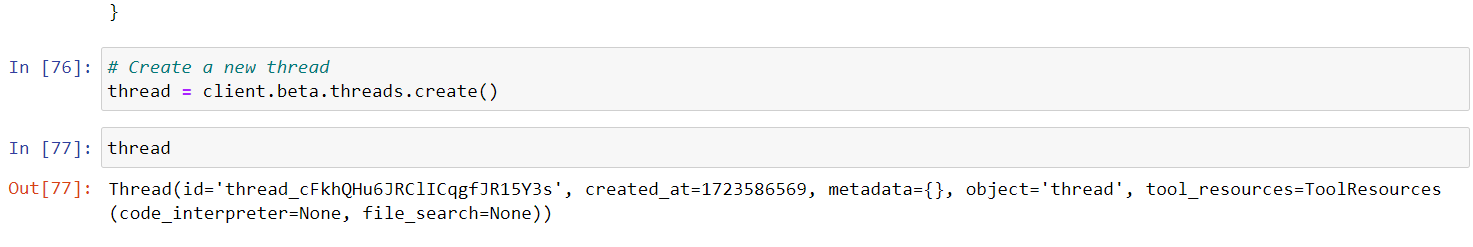
* **Purpose: Initialize the OpenAI client with the provided API key.**
* **Explanation: The OpenAI client is configured to interact with the API using the provided key.**

**4. Define Tools for the Assistant:**

* **Purpose: Define the tool that the assistant will use to retrieve weather information.**
* **Explanation: The tool is described with parameters for location and temperature unit.**

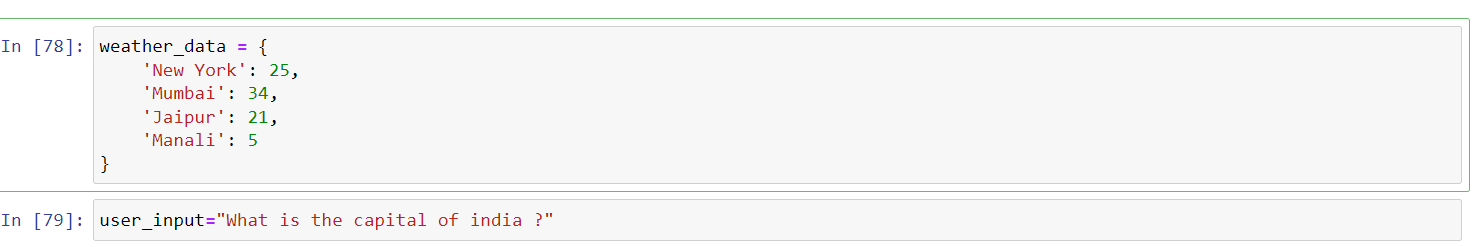
**5. Create a New Assistant:**

* **Purpose: Create and configure a new assistant with specific instructions and tools.**
* **Explanation: The assistant is created with the GPT-4 model and configured to handle weather-related queries.**

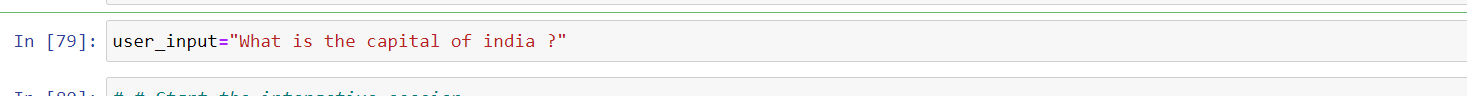
**6. Create a New Thread :** 

* **Purpose: Initialize a new thread for user interaction.**
* **Explanation: Threads are used to manage conversations with the assistant.**

1. **Define Predefined Weather Data:**

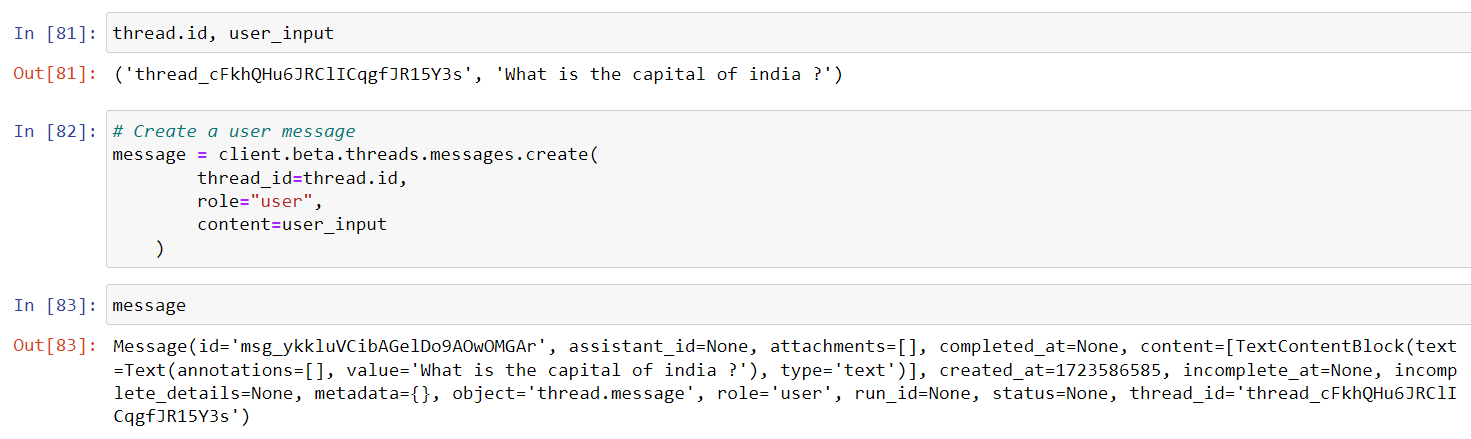


* **Purpose**: Define sample weather data for testing.
* **Explanation**: This data is used to simulate weather responses.

**8**. **Handle User Input:**

* **Purpose: Define a sample user query.**
* **Explanation: This input is used to simulate user interaction with the assistant.**

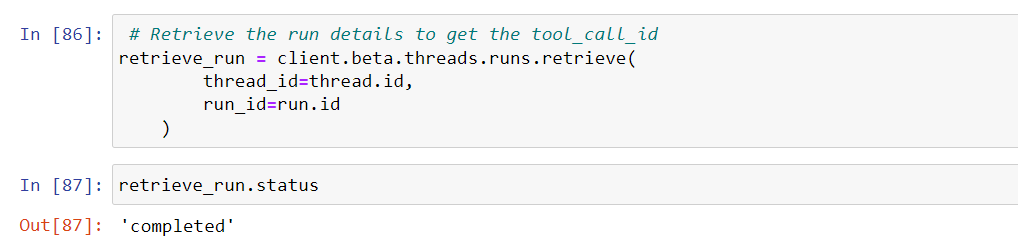
**9.Create User Message:**



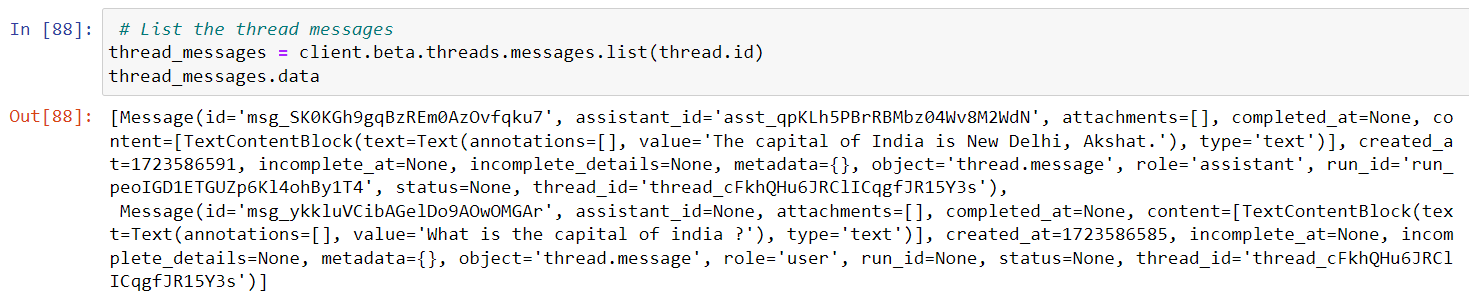
* **Purpose: Create a message in the thread with the user's query.**
* **Explanation: This message simulates user input within the conversation thread.**

**10.Create a Run for the Assistant:**

* **Purpose: Initiate a run for the assistant to process the user query.**
* **Explanation: A run is a single execution instance where the assistant processes the user input.**

**11. Retrieve Run Details :**

* **Purpose: Retrieve details about the run to get information on tool calls.**
* **Explanation: This helps in understanding the assistant’s processing status and tool usage.**

**12. List Thread Messages :  
  
**

* **Purpose: List all messages in the thread.**
* **Explanation: This helps in extracting both user and assistant messages for further processing.**

**13. Process Messages :**

  
  
  
 **Purpose: Separate and display user and assistant messages.**

* **Explanation: This helps in verifying the assistant’s response and user interaction.**

**14. Submit Tool Outputs:**  


**Purpose: Process and submit the output of the tool.**

## Explanation: This includes generating the weather output based on predefined data and submitting it as a response. **4. Result :** **The assistant successfully processes user queries related to weather information. It handles predefined weather data and returns responses based on the input location. The integration of tool outputs and message handling demonstrates effective interaction with the OpenAI API.**

**5. Conclusions:**

The implemented code effectively sets up an AI-driven assistant for weather queries. It showcases how to create an assistant, manage conversation threads, and process tool outputs. The approach allows for scalable interaction with users, though it primarily relies on predefined weather data for demonstration.  
  
6.Theory on the way :  
  
**1. Threads: A thread is a sequence of instructions that can be executed independently within a program. In the context of programming, threads allow for parallel execution of tasks, enabling more efficient use of system resources by performing multiple operations concurrently.**

**2. Reply: In the context of communication or systems, a reply refers to the response generated or provided after receiving a query or input. In programming, this can refer to the output of a function or system based on the input data provided.**

**3. Linear Function Calling: Linear function calling refers to executing functions sequentially, where each function completes its execution before the next one begins. There is no parallelism or concurrency involved, and the program follows a straight line of execution from one function to the next.**

**4. Function: A function is a reusable block of code designed to perform a specific task. It takes input parameters (optional), processes them, and returns an output . Functions allow for code modularity and reusability, helping to structure larger programs efficiently**.

**7. References:**<https://platform.openai.com/docs/overview>  
<https://platform.openai.com/docs/models>  
<https://platform.openai.com/docs/guides/function-calling>  
<https://platform.openai.com/docs/guides/structured-outputs>  
<https://platform.openai.com/api-keys>  
<https://gist.github.com/win3zz/0a1c70589fcbea64dba4588b93095855>  
<https://github.com/PawanOsman/ChatGPT>  
<https://github.com/PawanOsman/ChatGPT>  
<https://github.com/openai/openai-python>