**Name:Garige.kalpana**

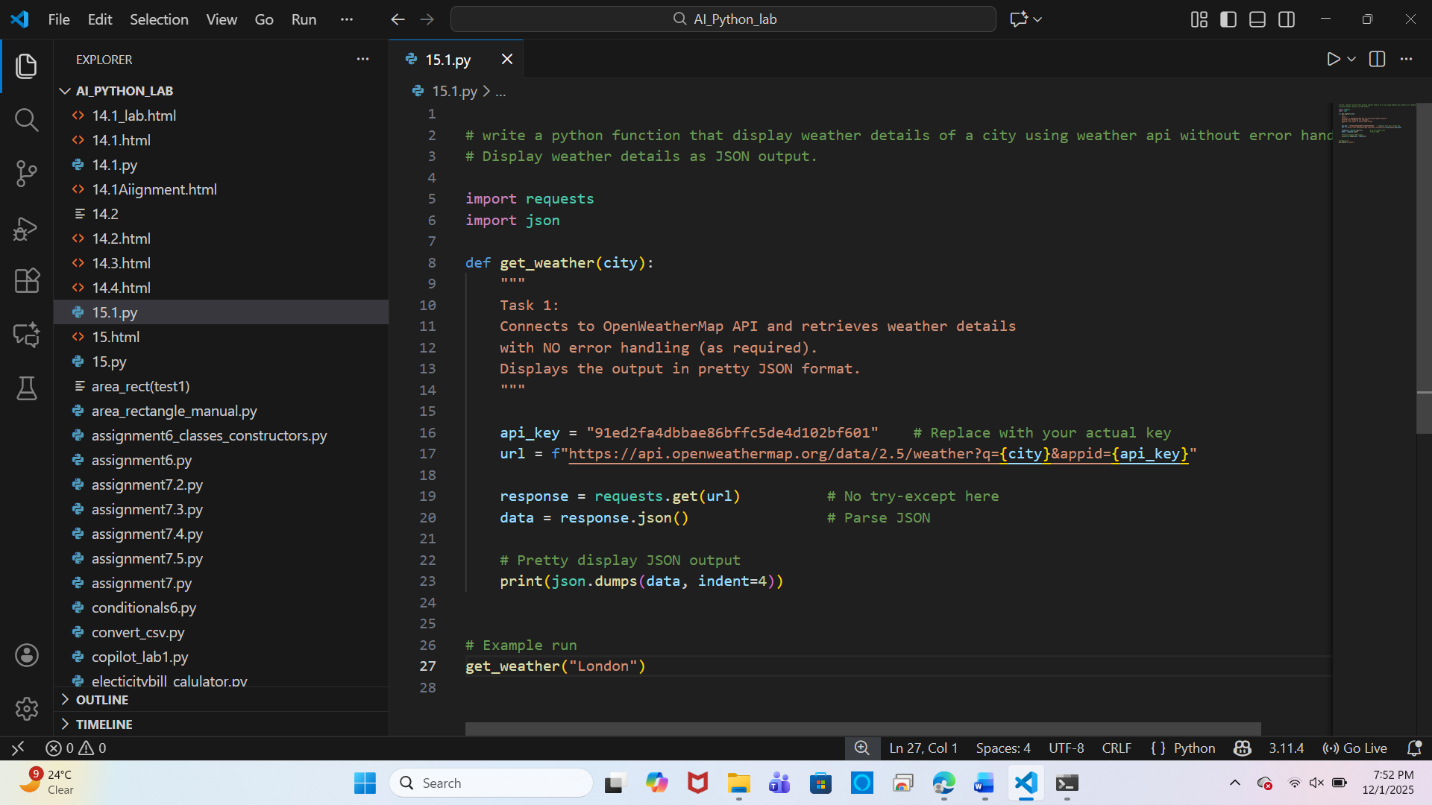
**Roll Number: 2503B05117**

**Lab 18 – API Integration: Connecting to External Services with Error Handling**

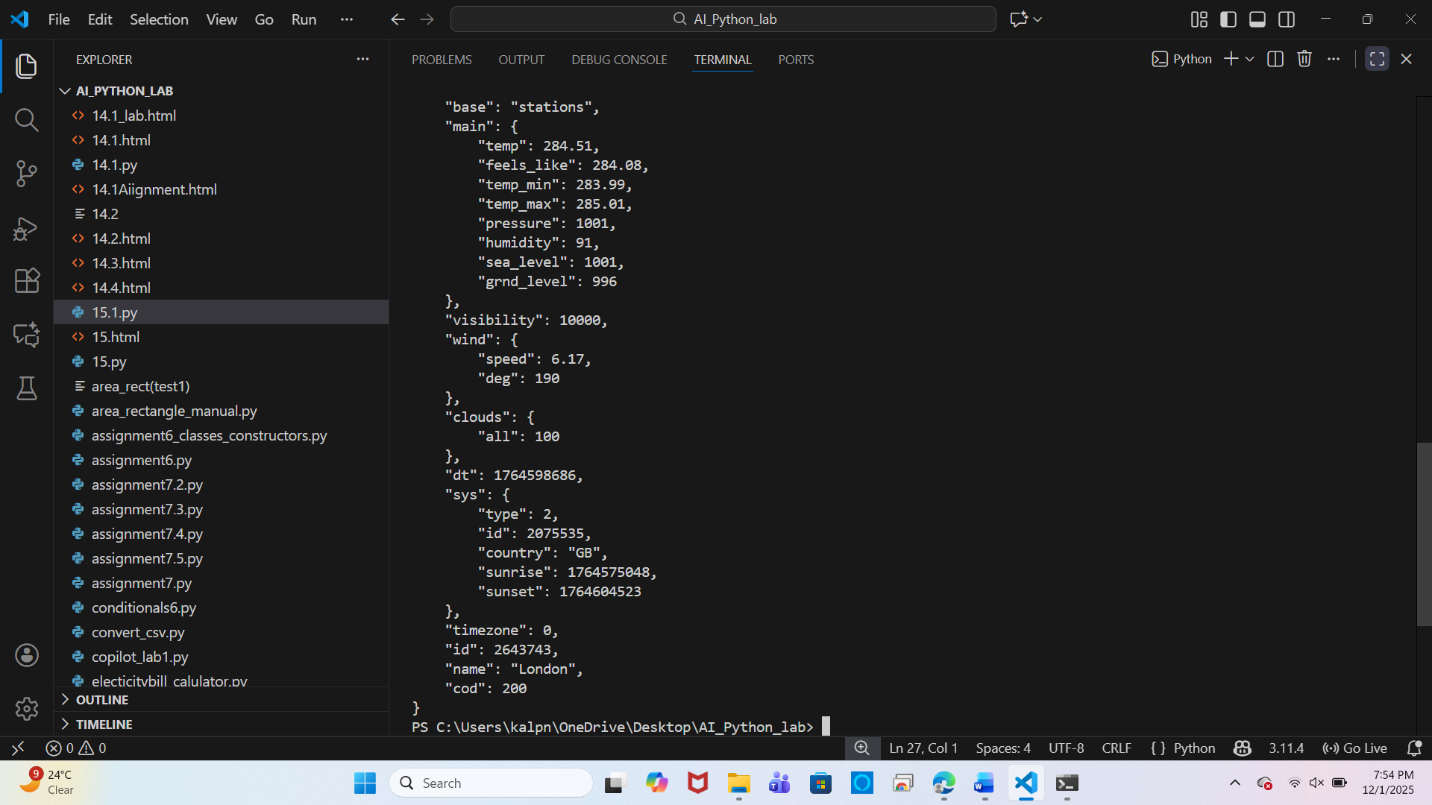
**Task 1: Connect to a Public API**

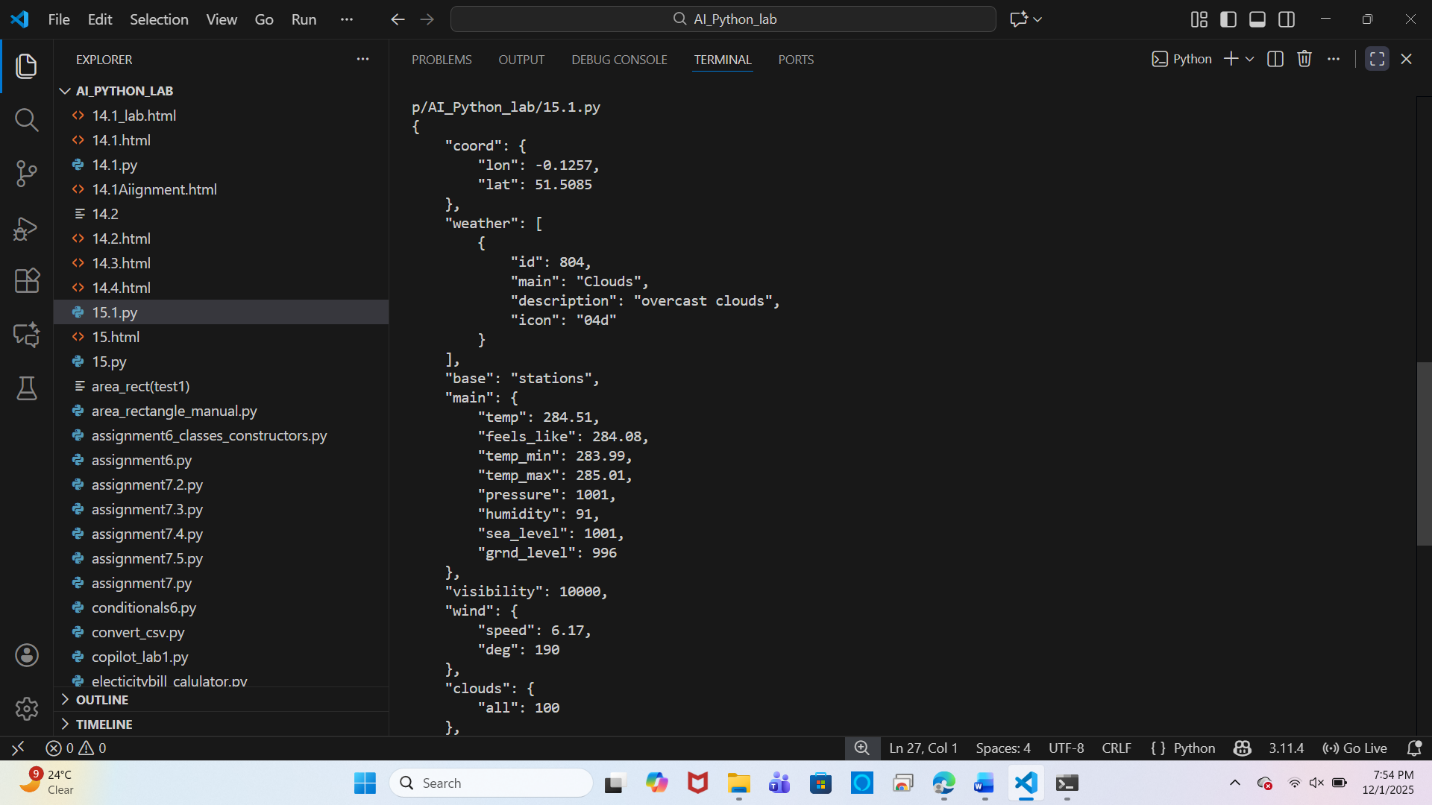
* **Instructions:**
  + **Use Python (or Node.js/JavaScript) to connect to a public API (e.g., OpenWeatherMap or JSONPlaceholder).**
  + **Send a simple GET request to retrieve data.**

**Display the response in a readable format (pretty JSON)**



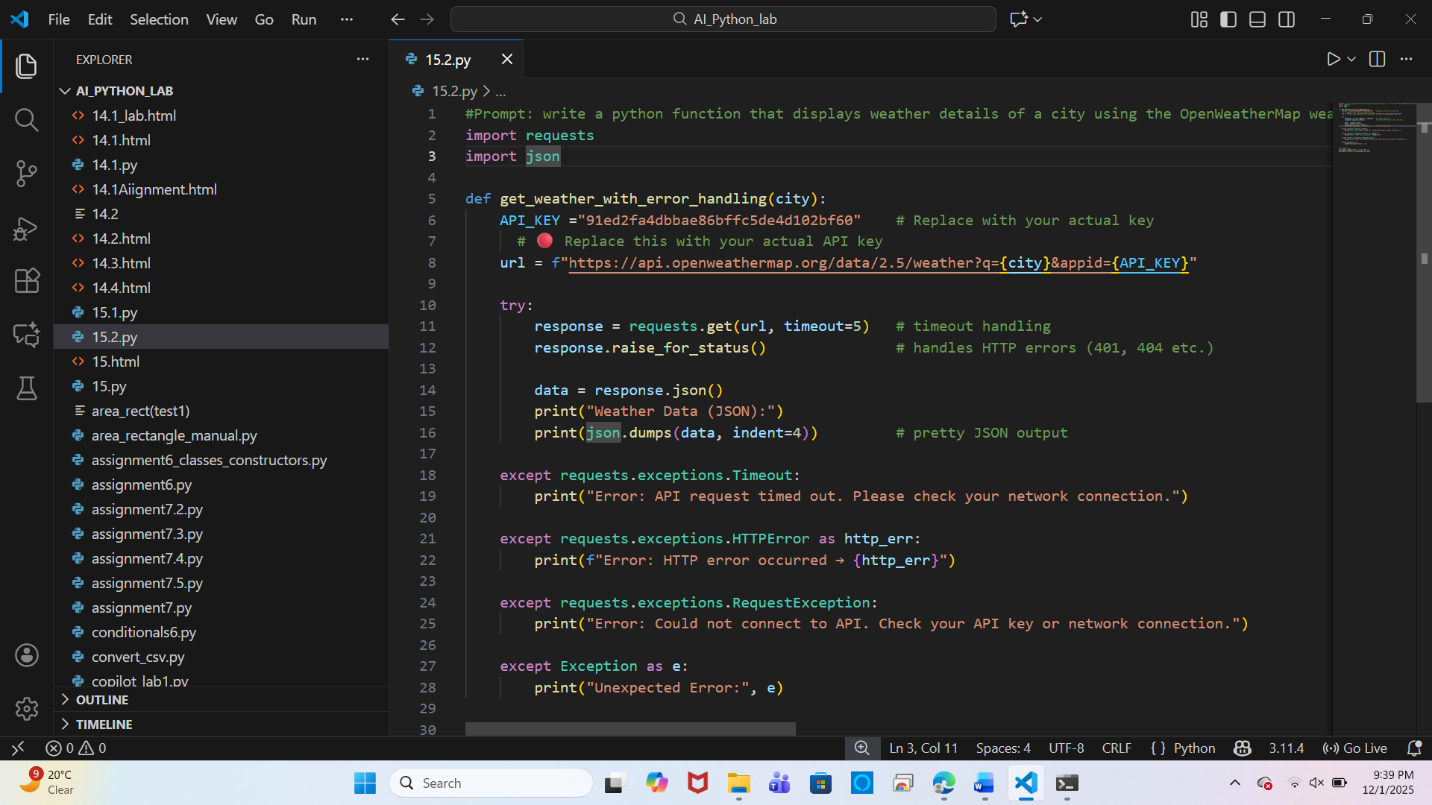
**Output**:

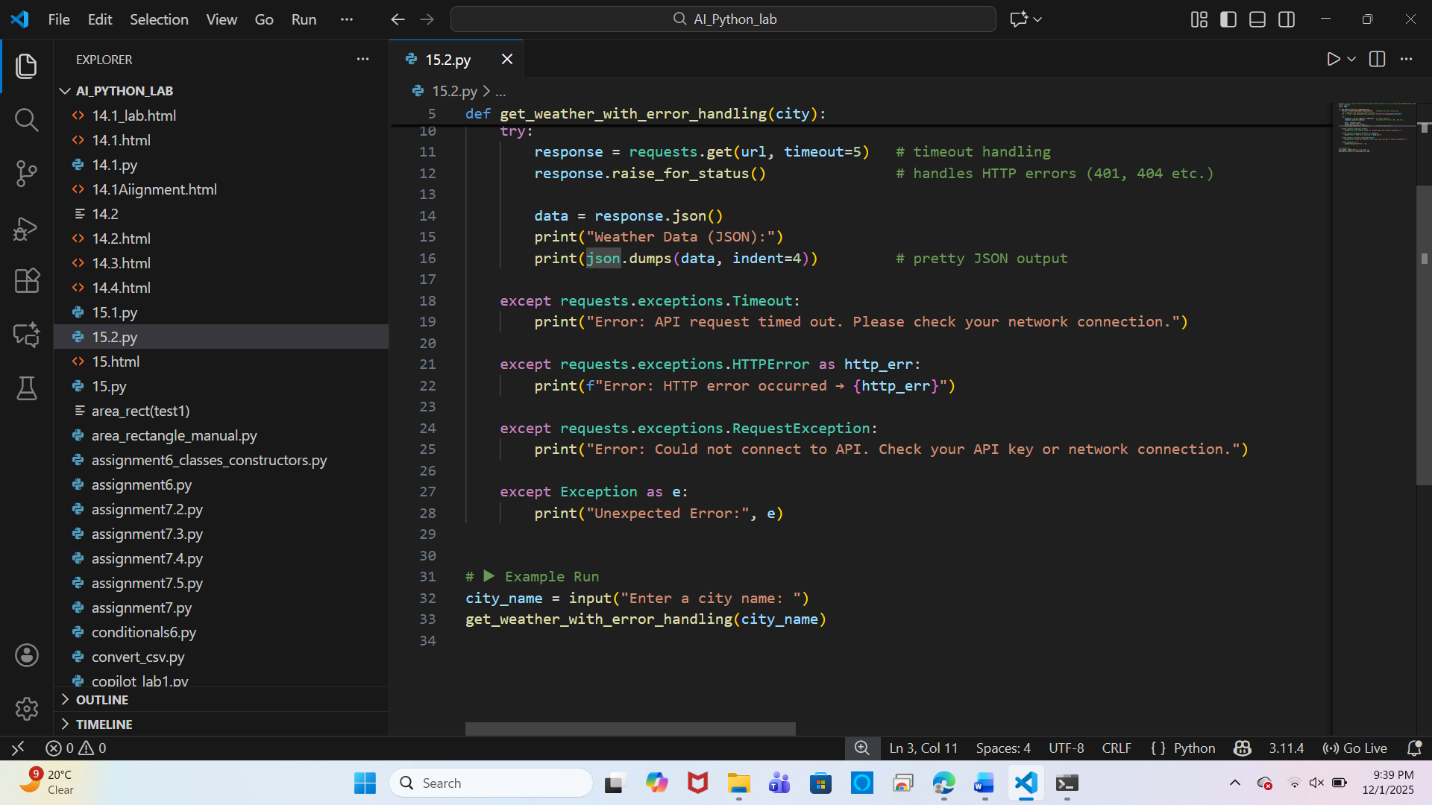




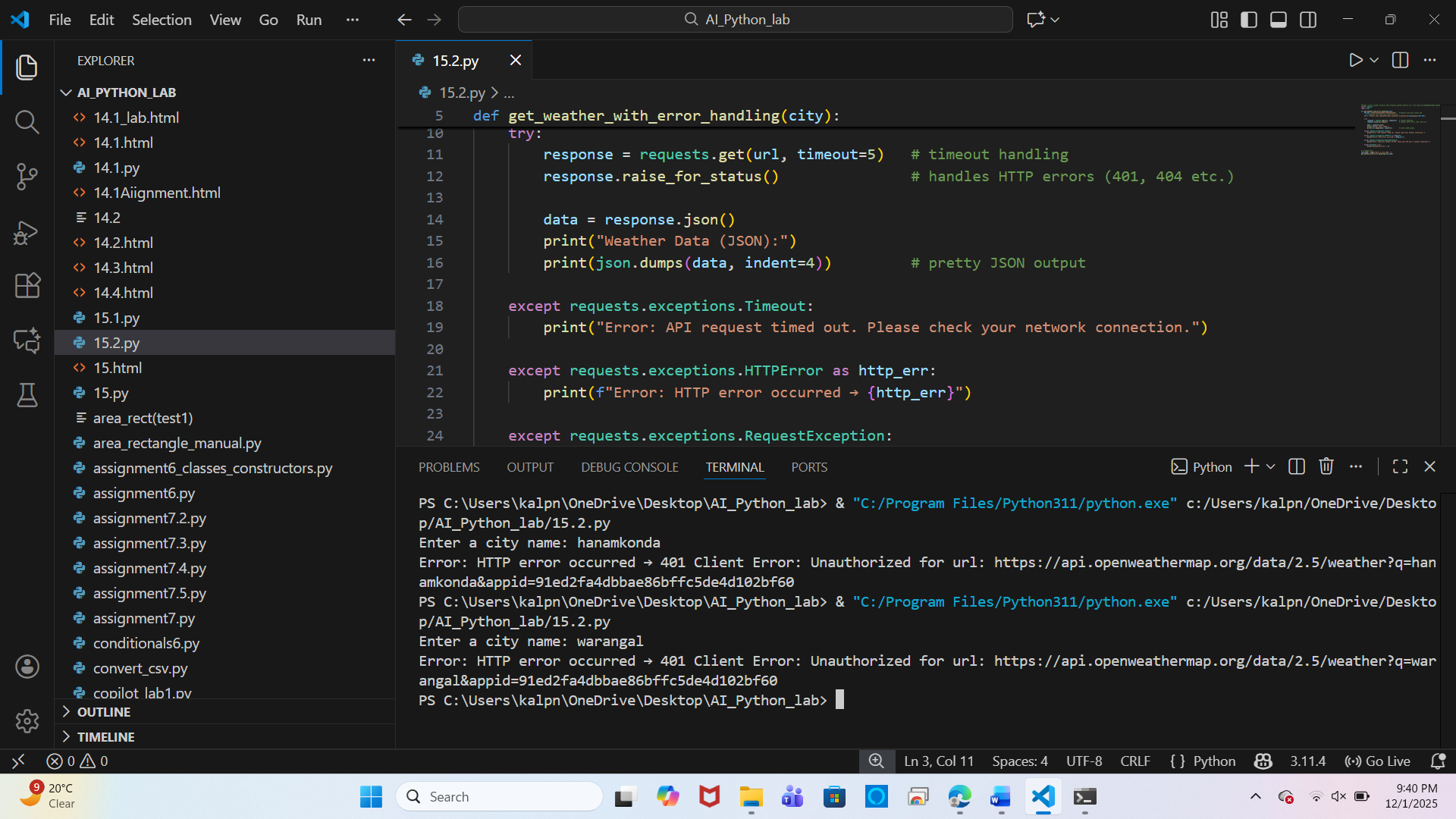
**Task 2: Add Error Handling for Invalid API Calls**

* **Instructions:**
  + Modify your code from Task 1 to handle errors.
  + Include try/except (Python) or try/catch (JavaScript) blocks.
  + Handle cases like:
    - Invalid URL
    - Network timeout
    - Wrong API key (if required)
  + Print user-friendly error messages.



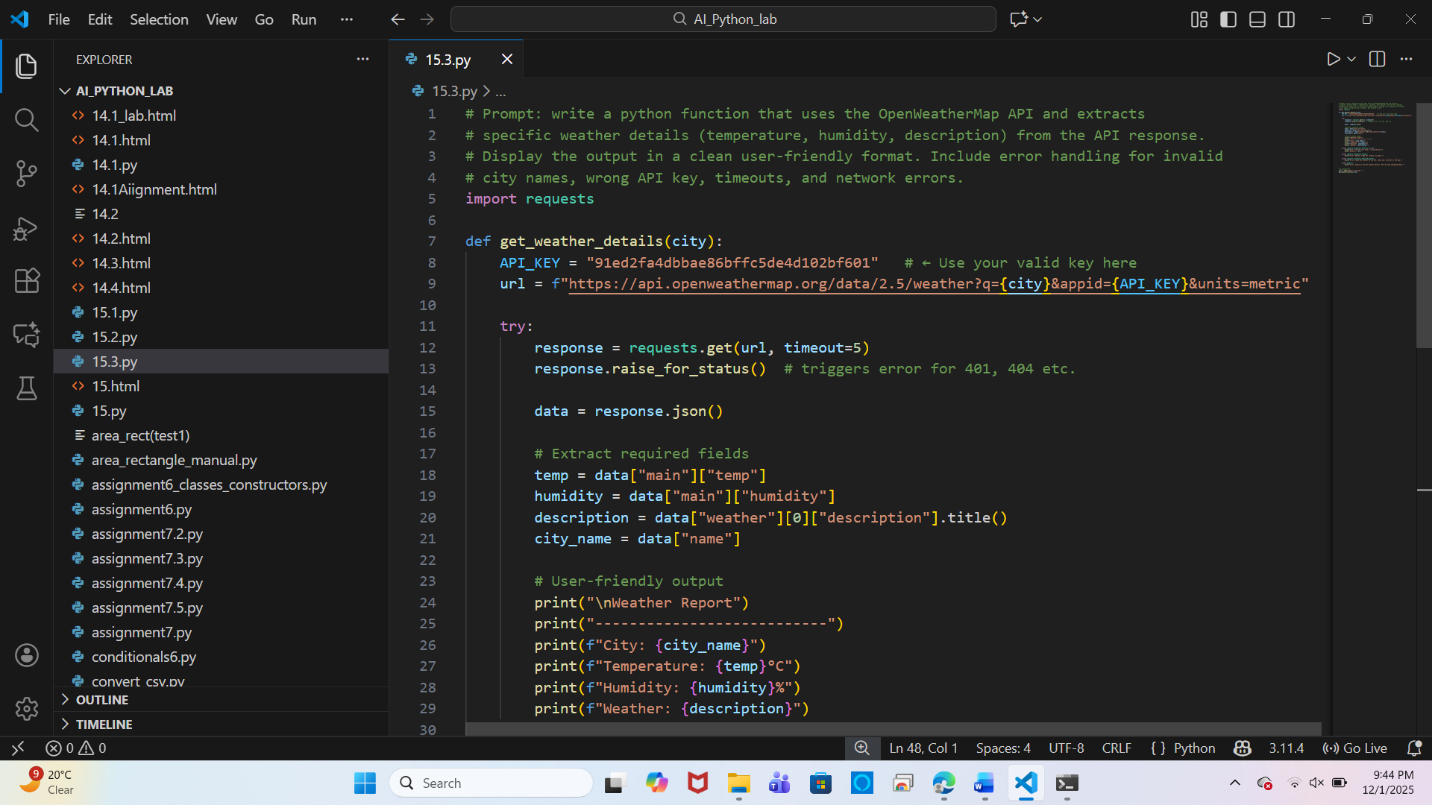


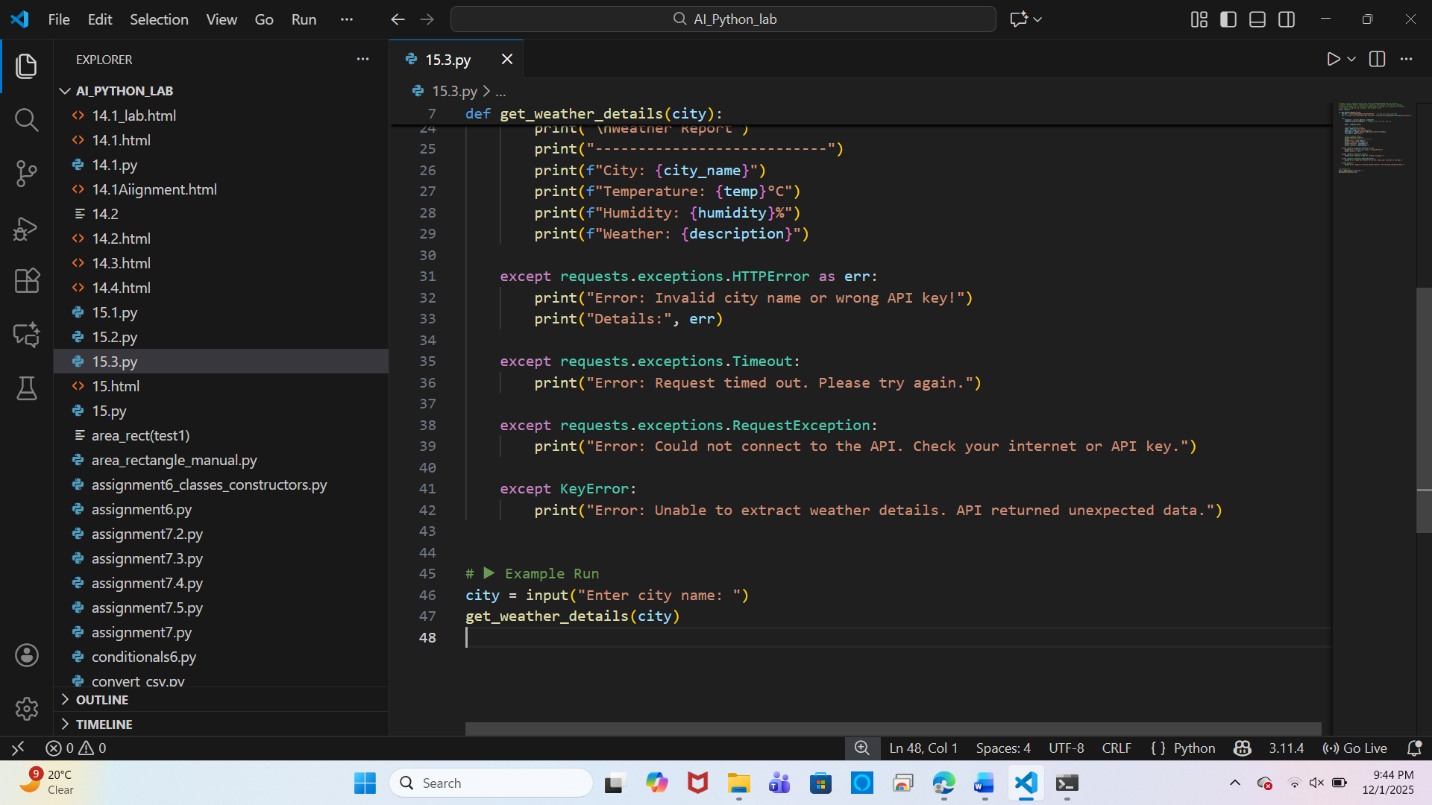
**Output**:



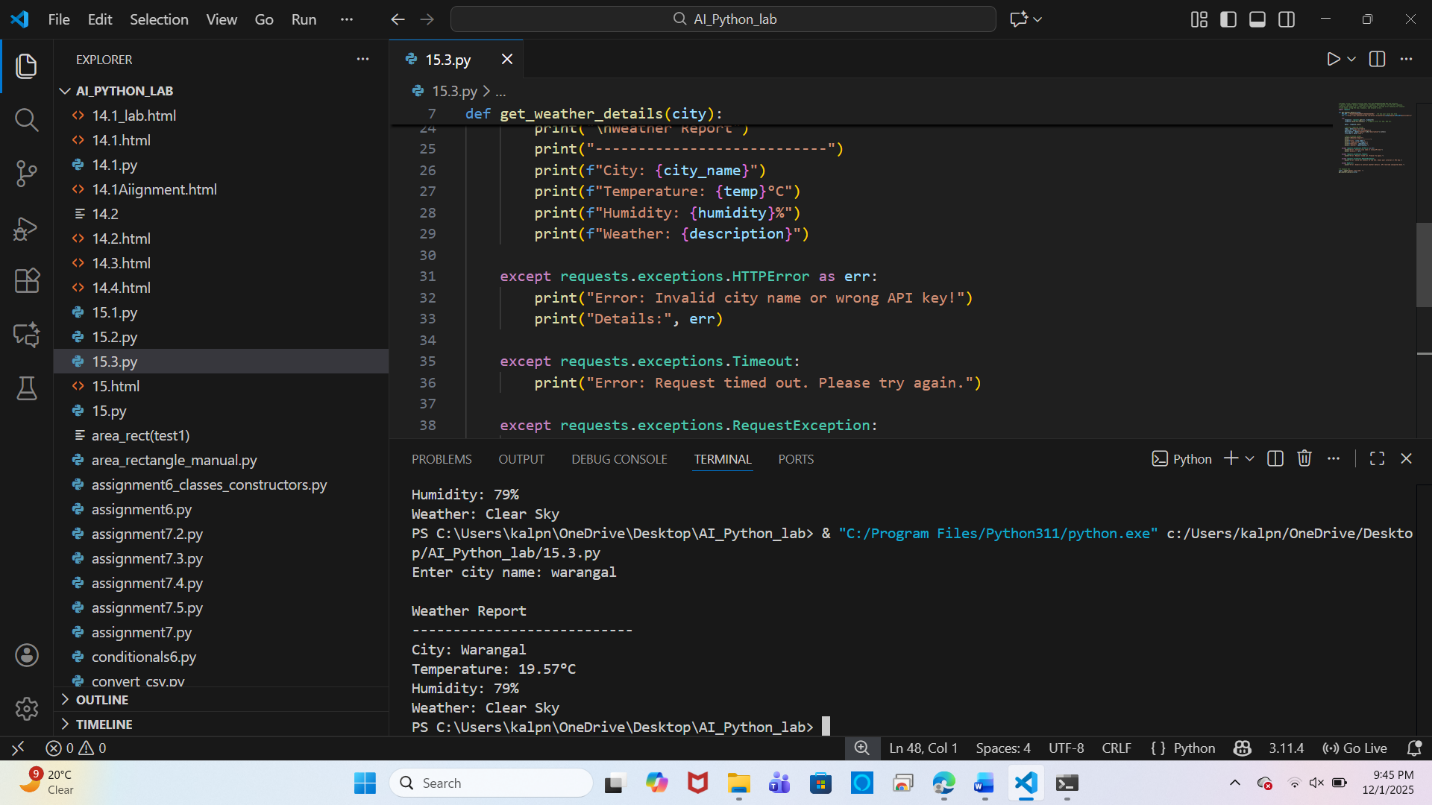
**Task 3: Extract and Display Specific Data**

* **Instructions:**
  1. From the API response (e.g., weather API), extract specific fields (temperature, humidity, description).
  2. Display them in a user-friendly format (not raw JSON).



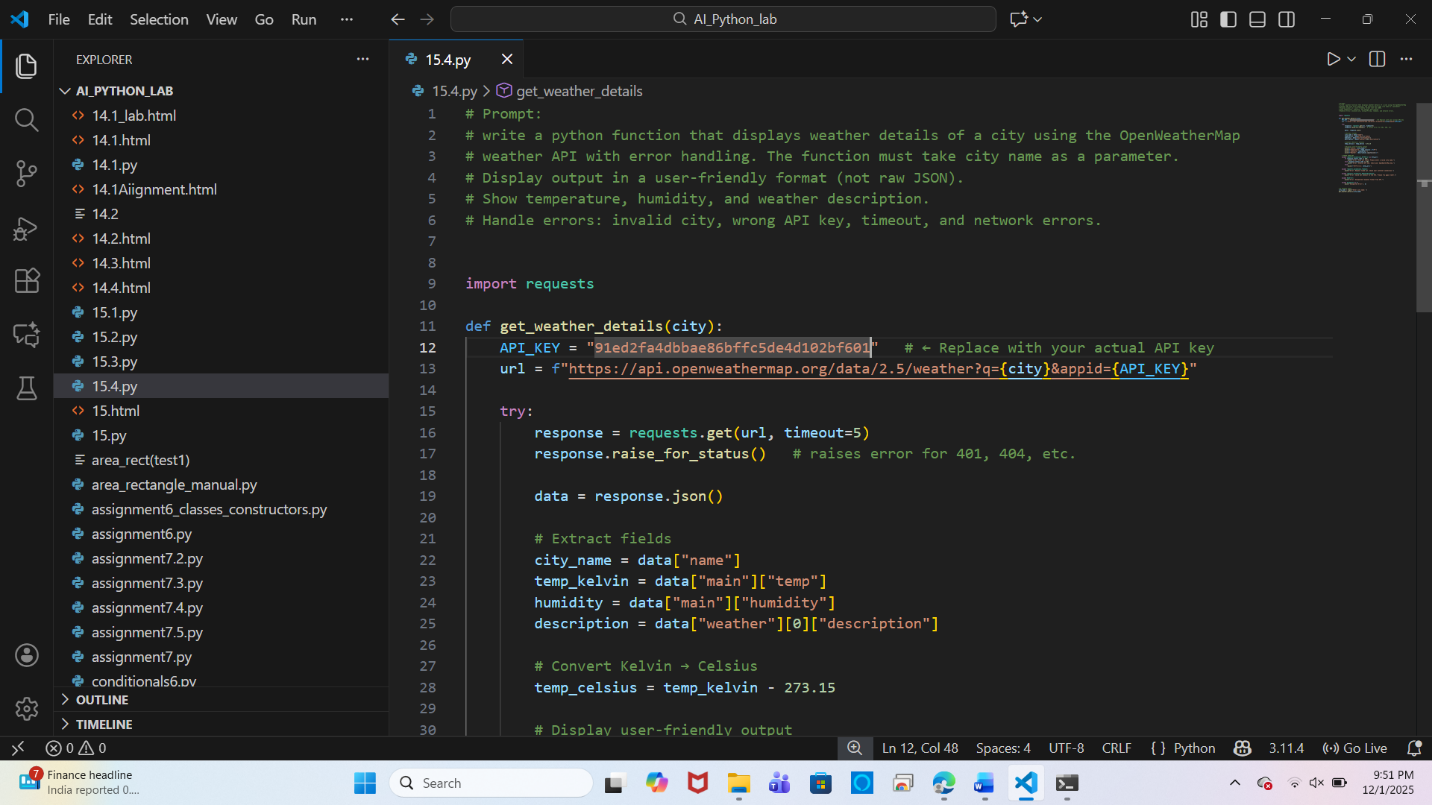


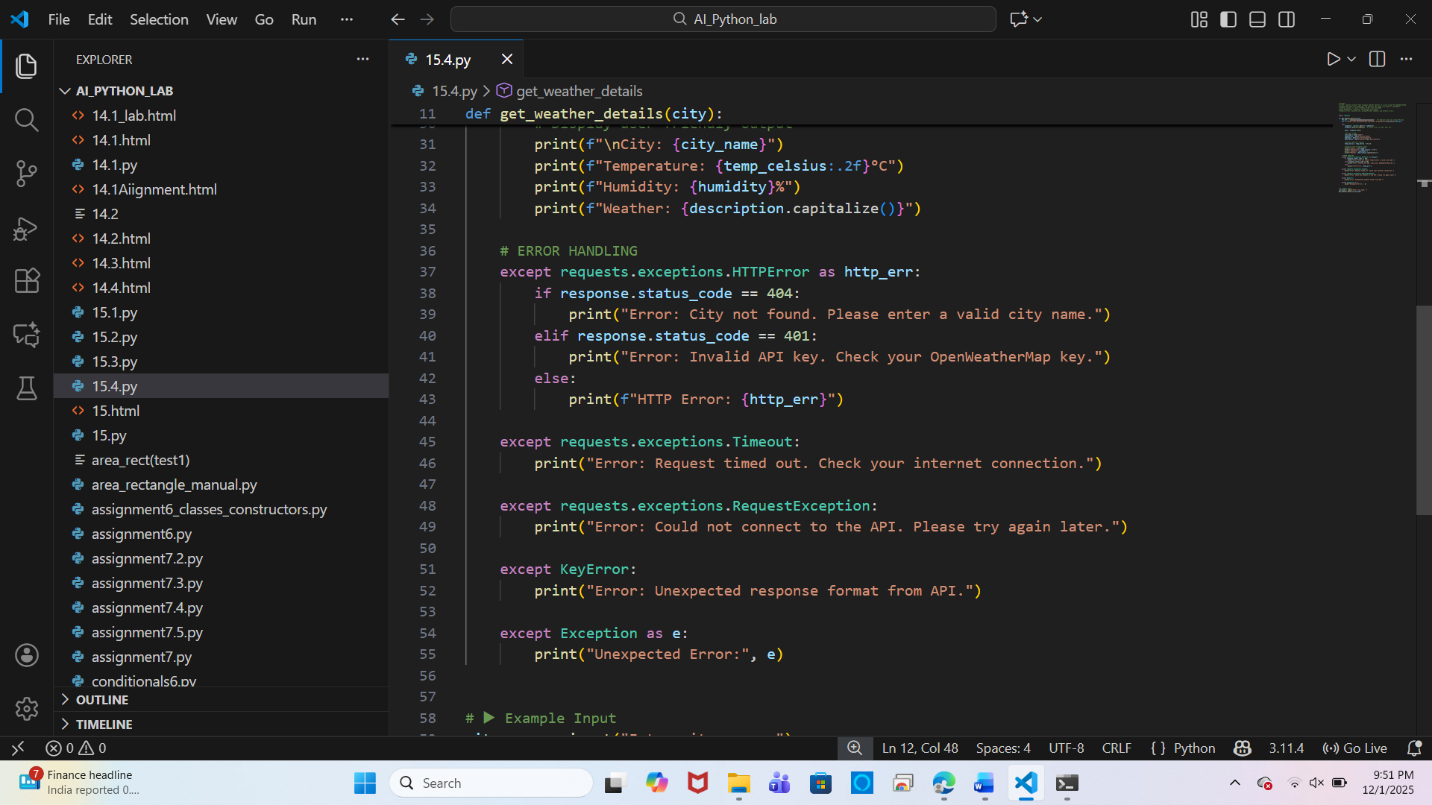
**Output**:

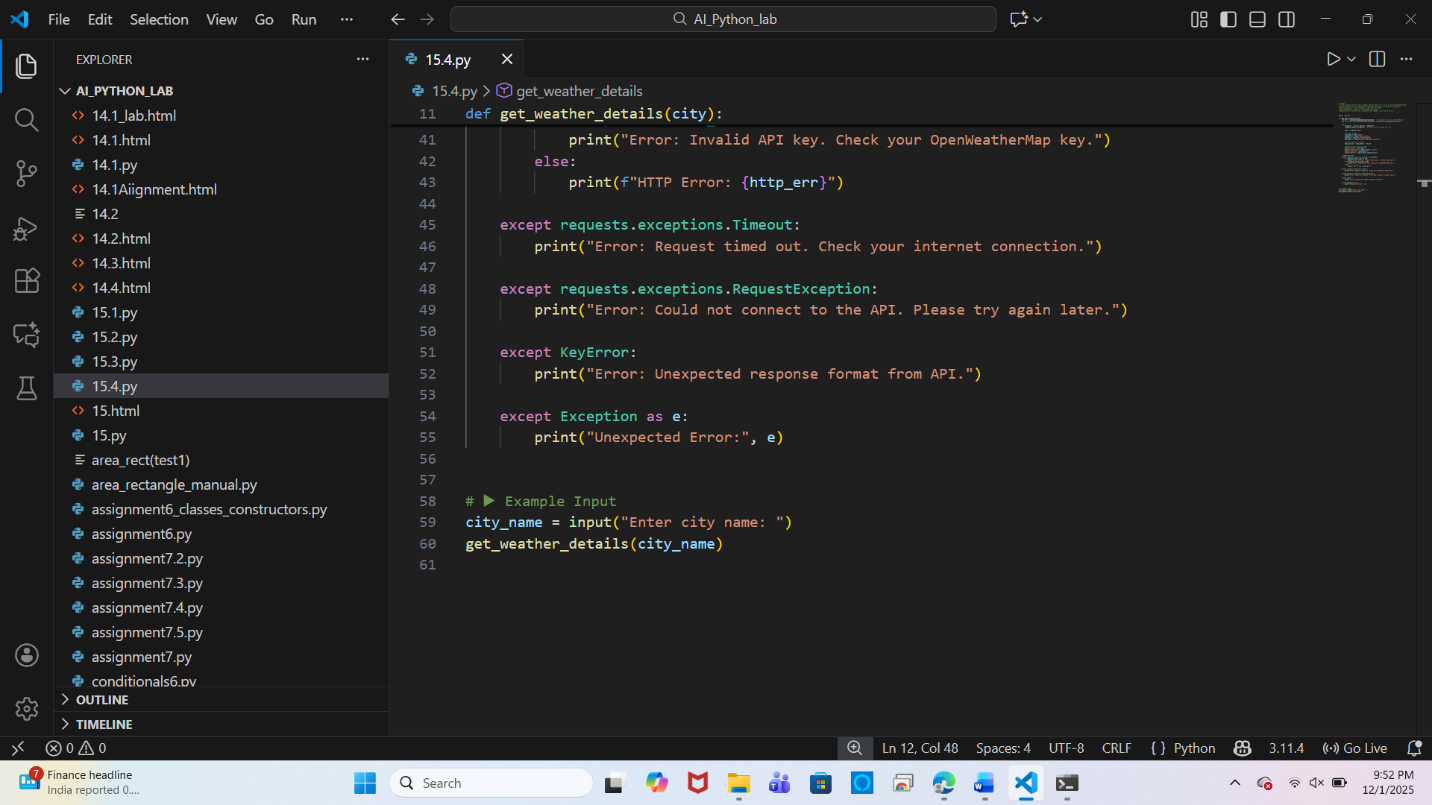


**Task 4: Build a Function with Parameters**

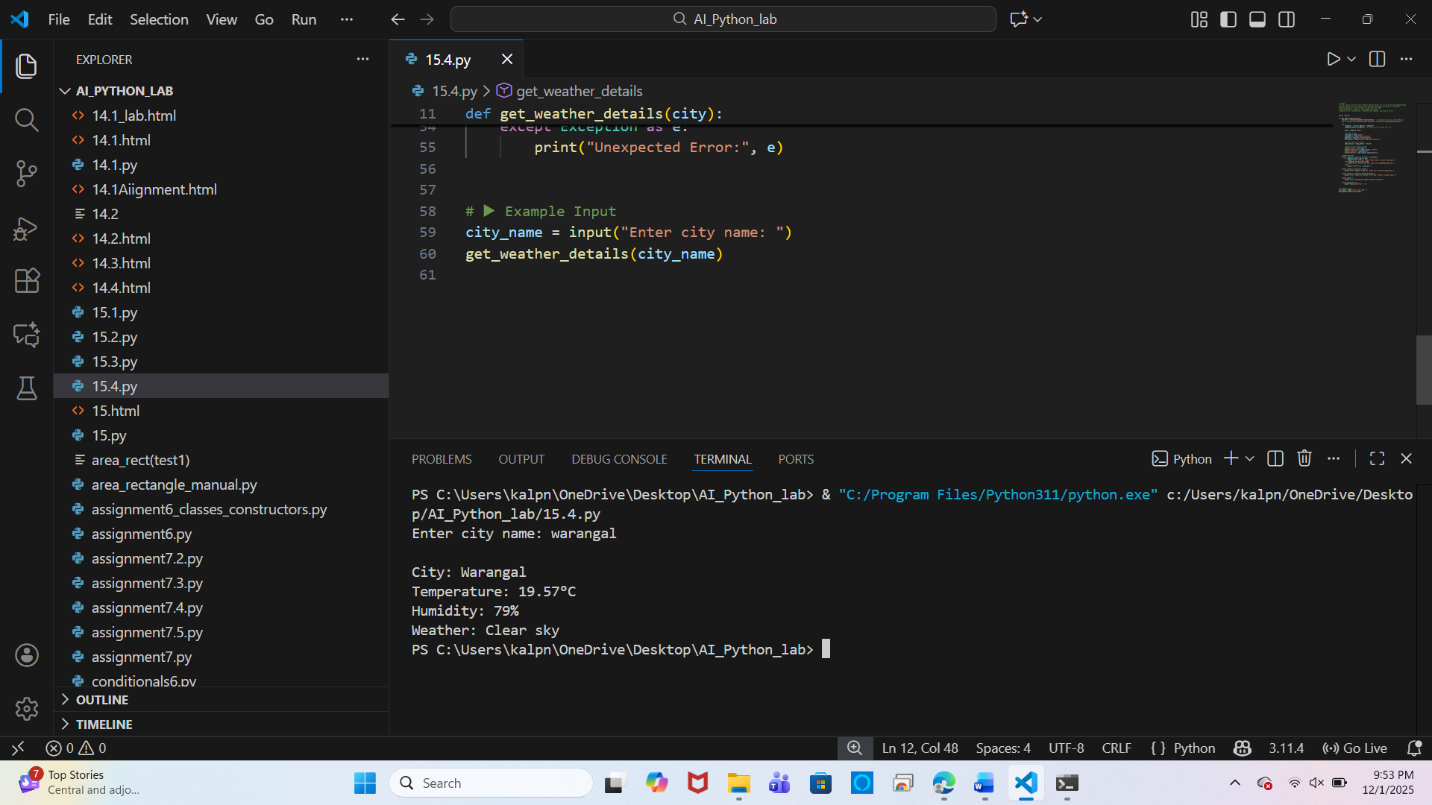
* **Instructions:**
  + Write a function that accepts a parameter (e.g., city name for weather API).
  + The function should call the API dynamically based on user input.
  + Include error handling if the city is invalid.

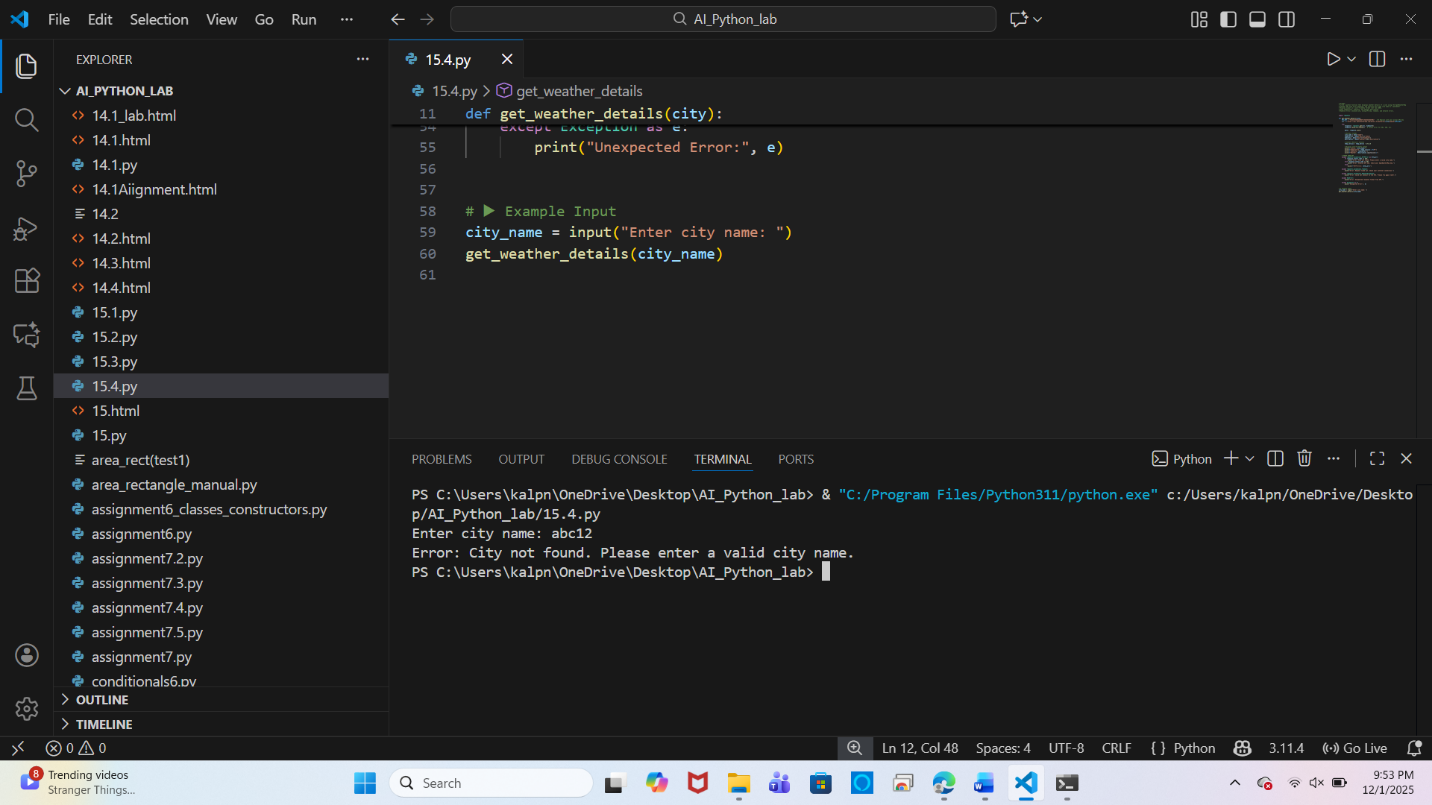






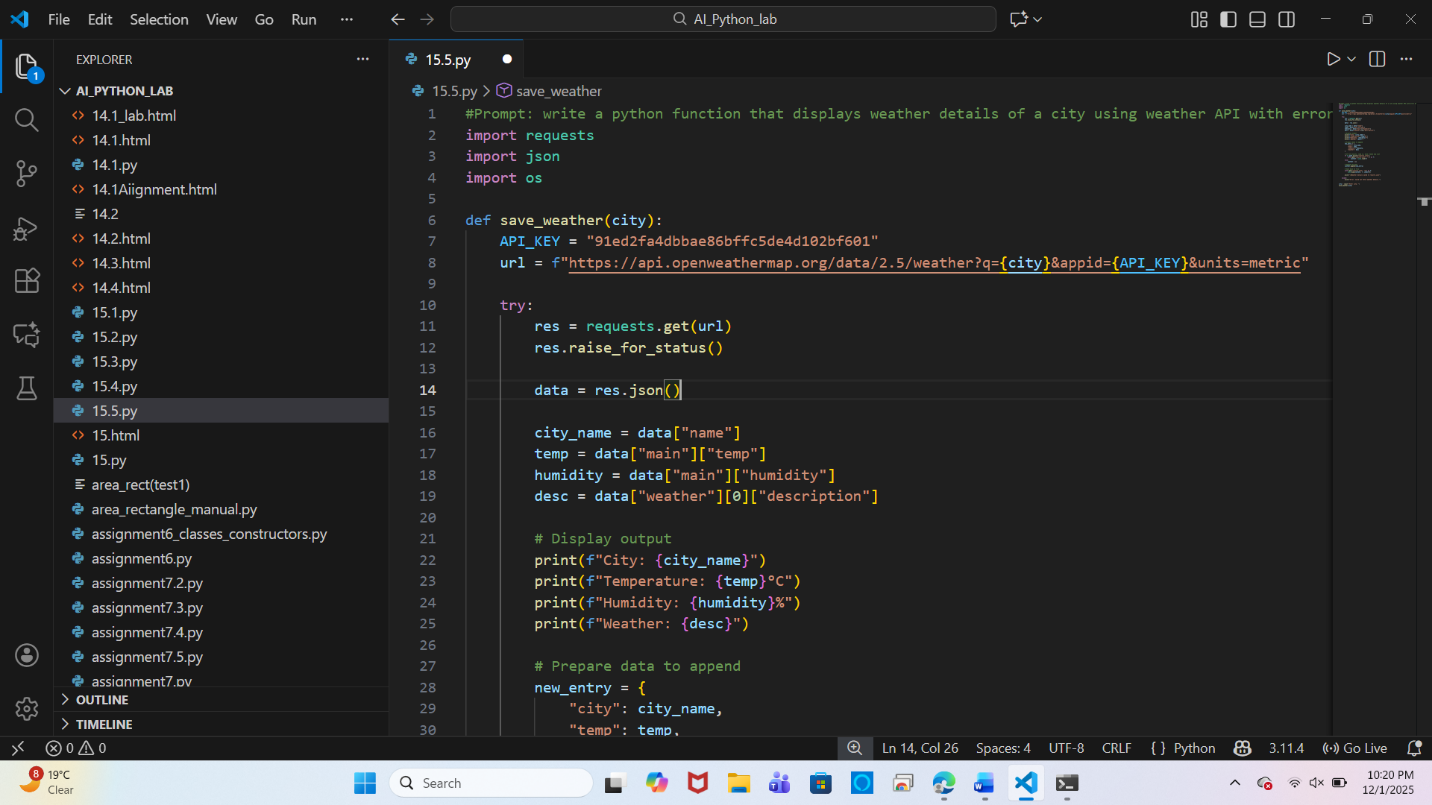
**Output:**

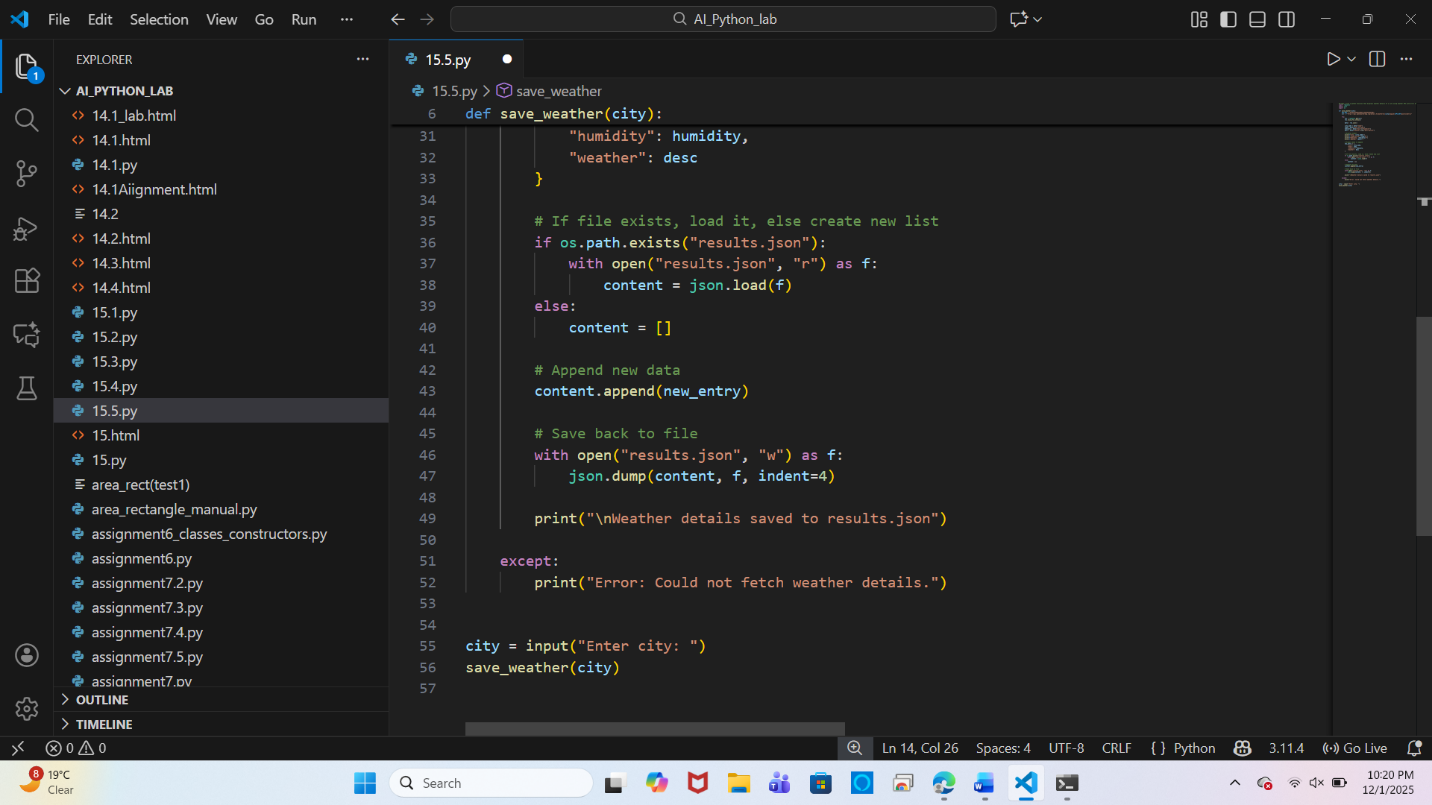
****

****

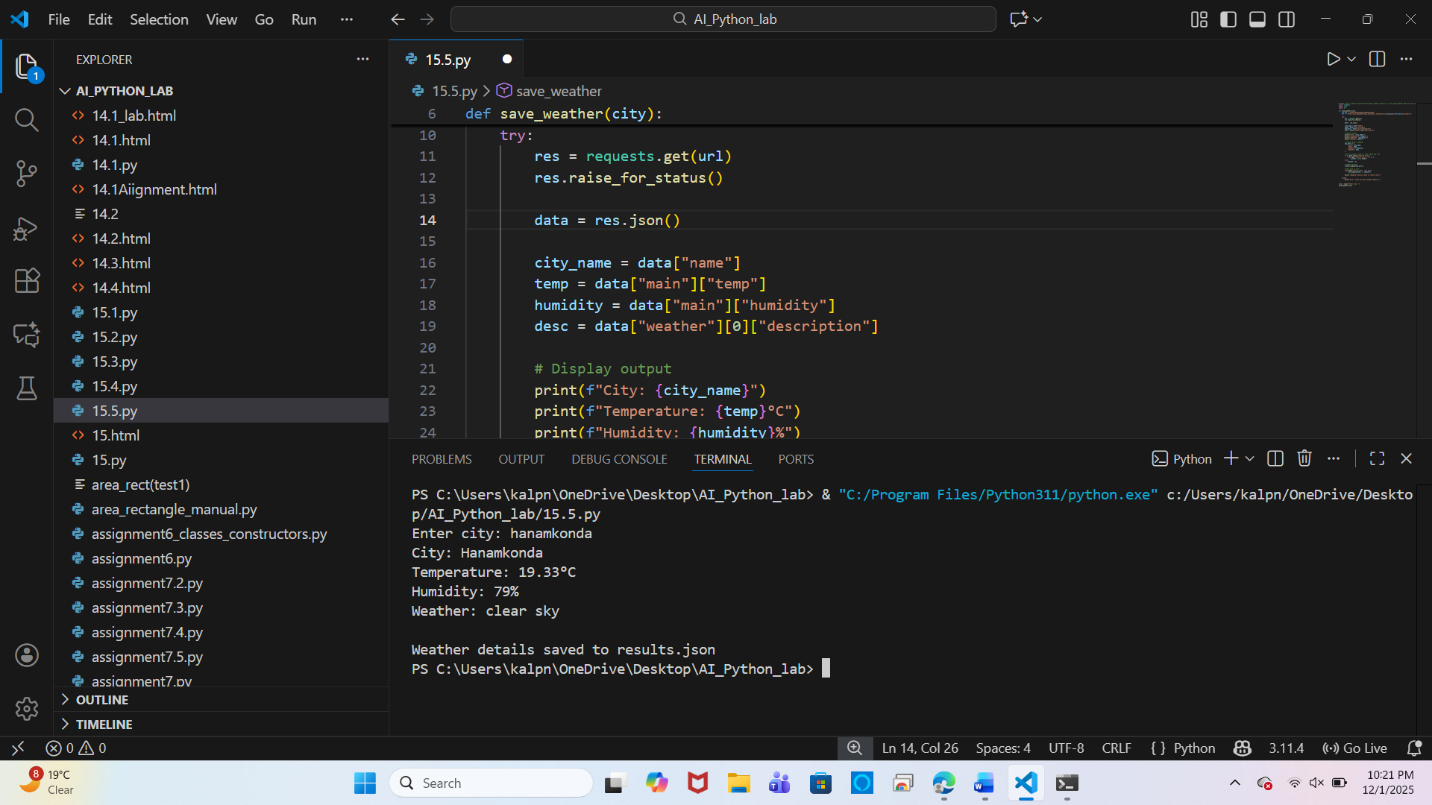
**Task 5: Store API Results Locally**

* **Instructions:**
  + **Extend your function from Task 4.**
  + **Save the extracted API results into a local file (results.json or results.txt).**
  + **Each new request should append results without overwriting old ones.**

****

****

**Output:**

****