**ASSINMENT-18.1**

**Name:** BOMMIDENI PRANITHA

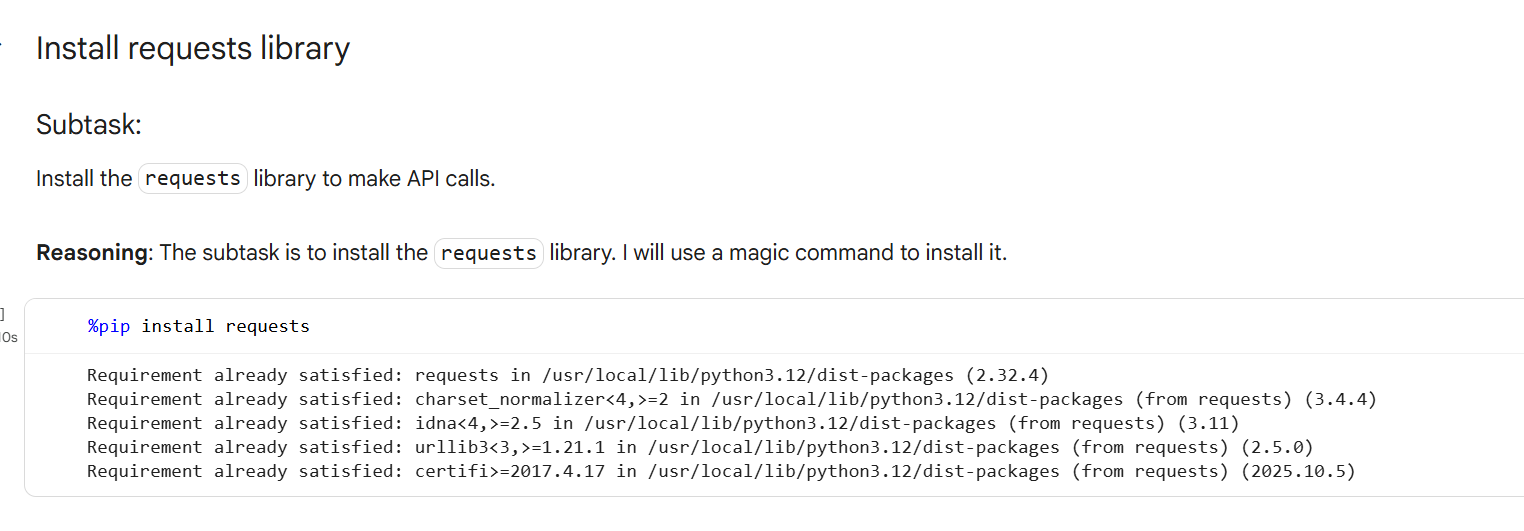
**HTNO:**2403A51351

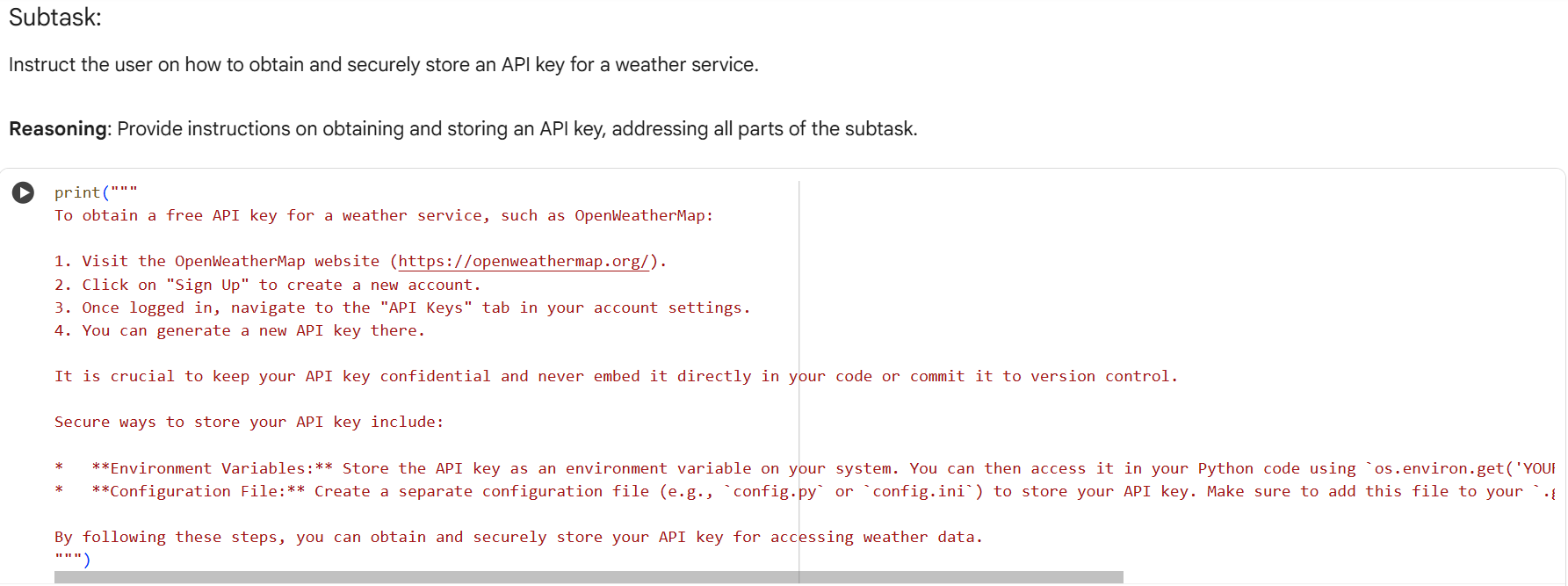
**Batch:**14

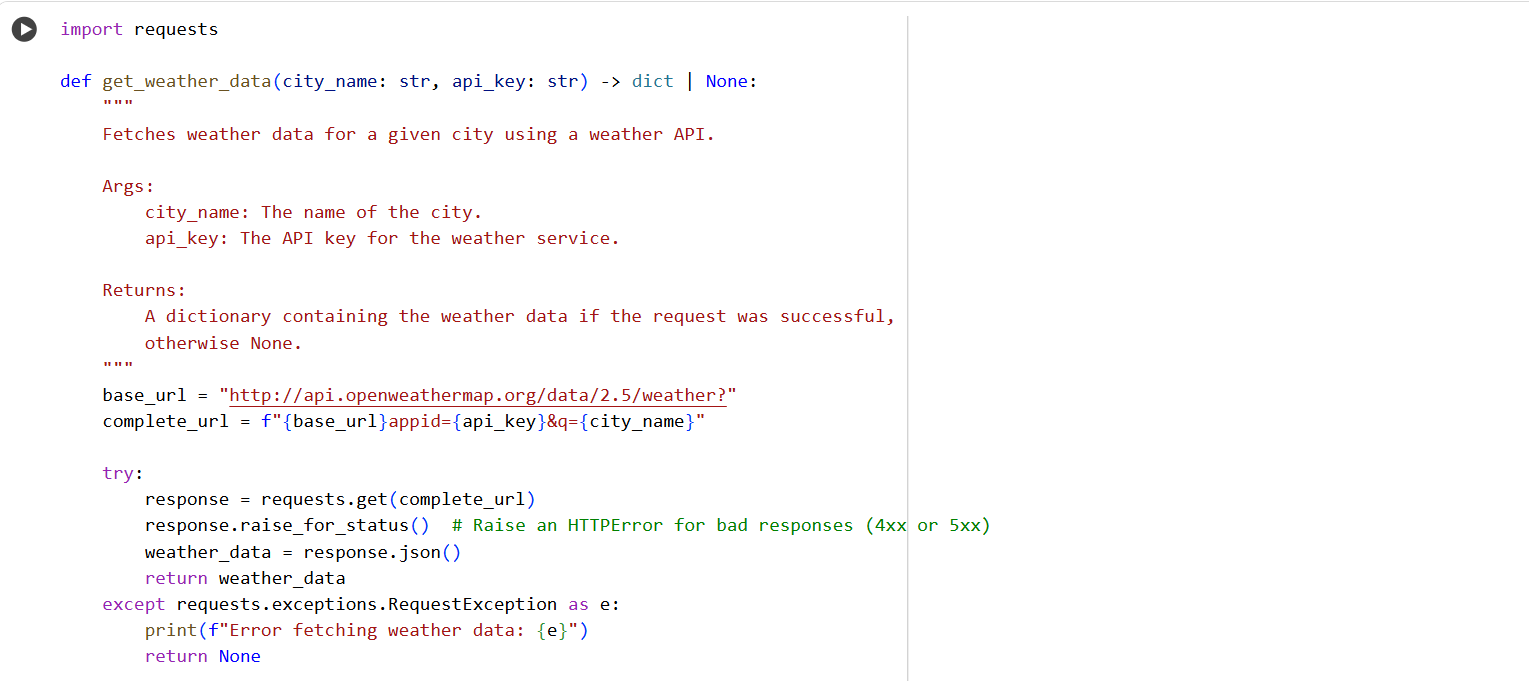
Task-1: Weather API Integration

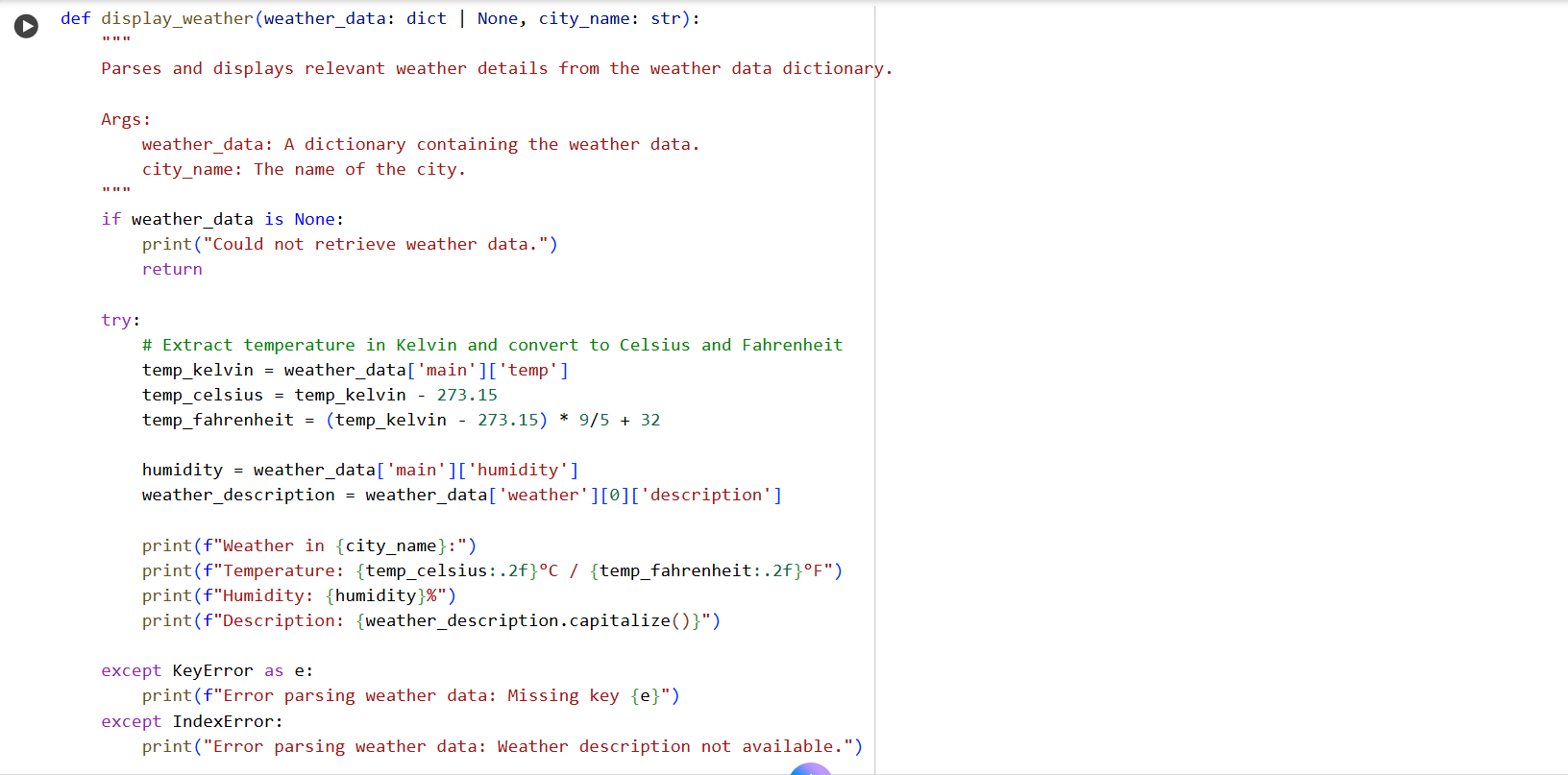
**Prompt:**

Generate Python code using the requests library to connect to a Weather API and fetch current weather details for a given city. Include error handling for invalid city names, API key issues, and network failures. Display temperature, humidity, and weather description in a readable format.









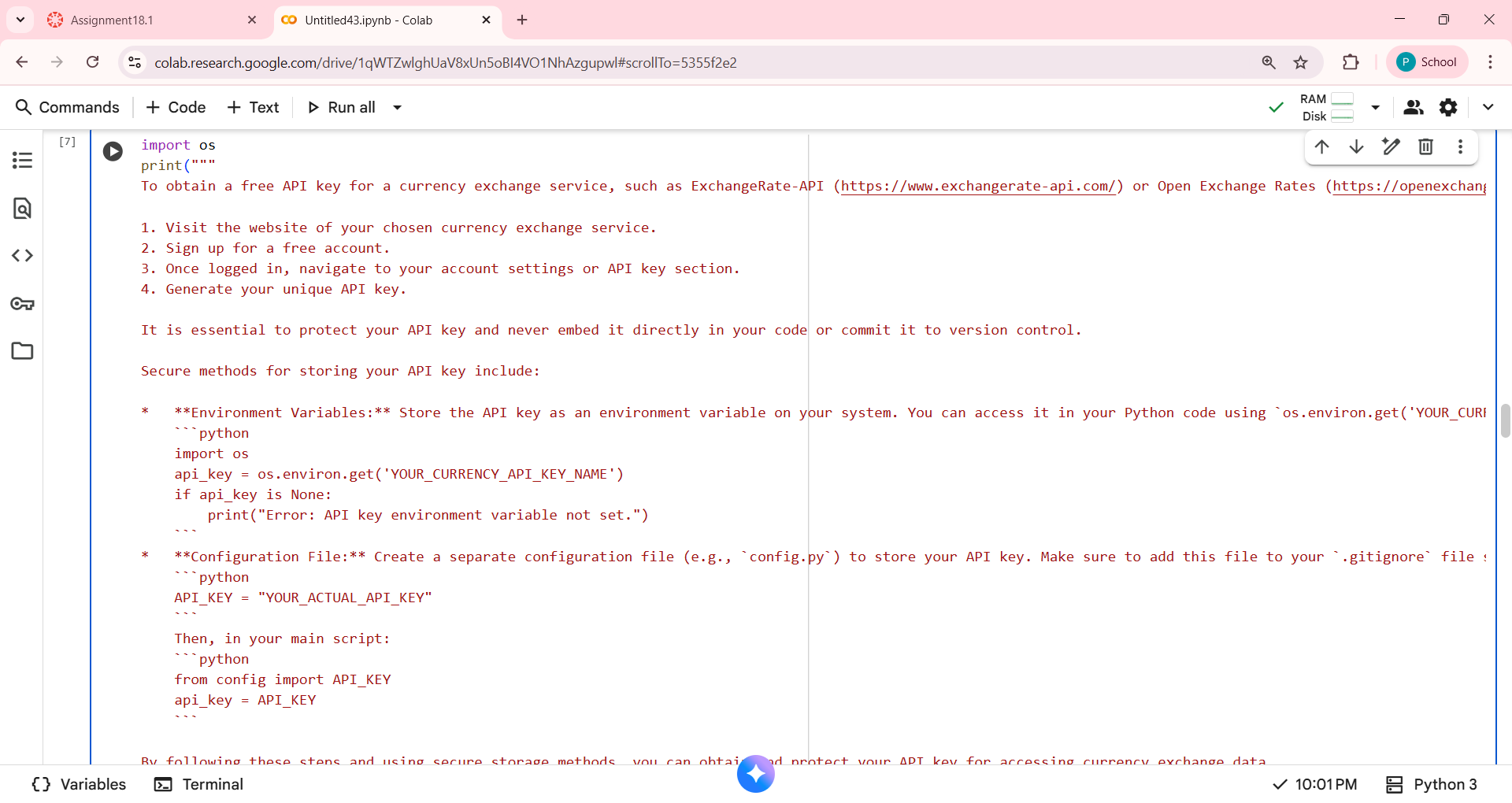
**Explanation:**

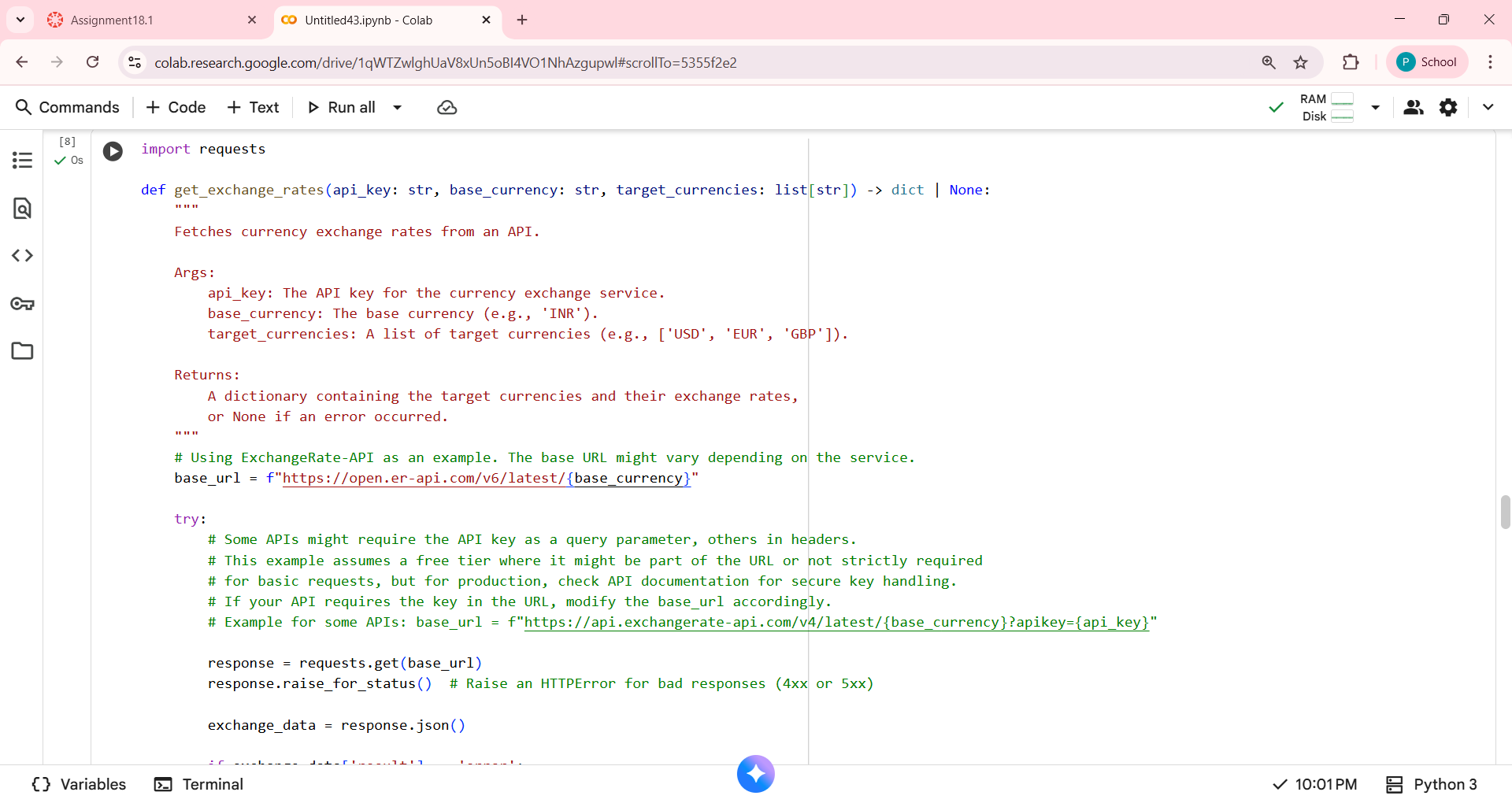
* The requests library, necessary for making API calls, was already installed in the environment.
* Detailed instructions were provided on how to obtain a free API key from OpenWeatherMap and recommended secure storage methods like environment variables and configuration files.
* A Python function get\_weather\_data was successfully created to fetch weather data for a given city using an API key, including error handling.
* A Python function display\_weather was created to parse the fetched weather data, convert temperature from Kelvin to Celsius and Fahrenheit, and display the temperature, humidity, and weather description in a readable format, including error handling for missing data.

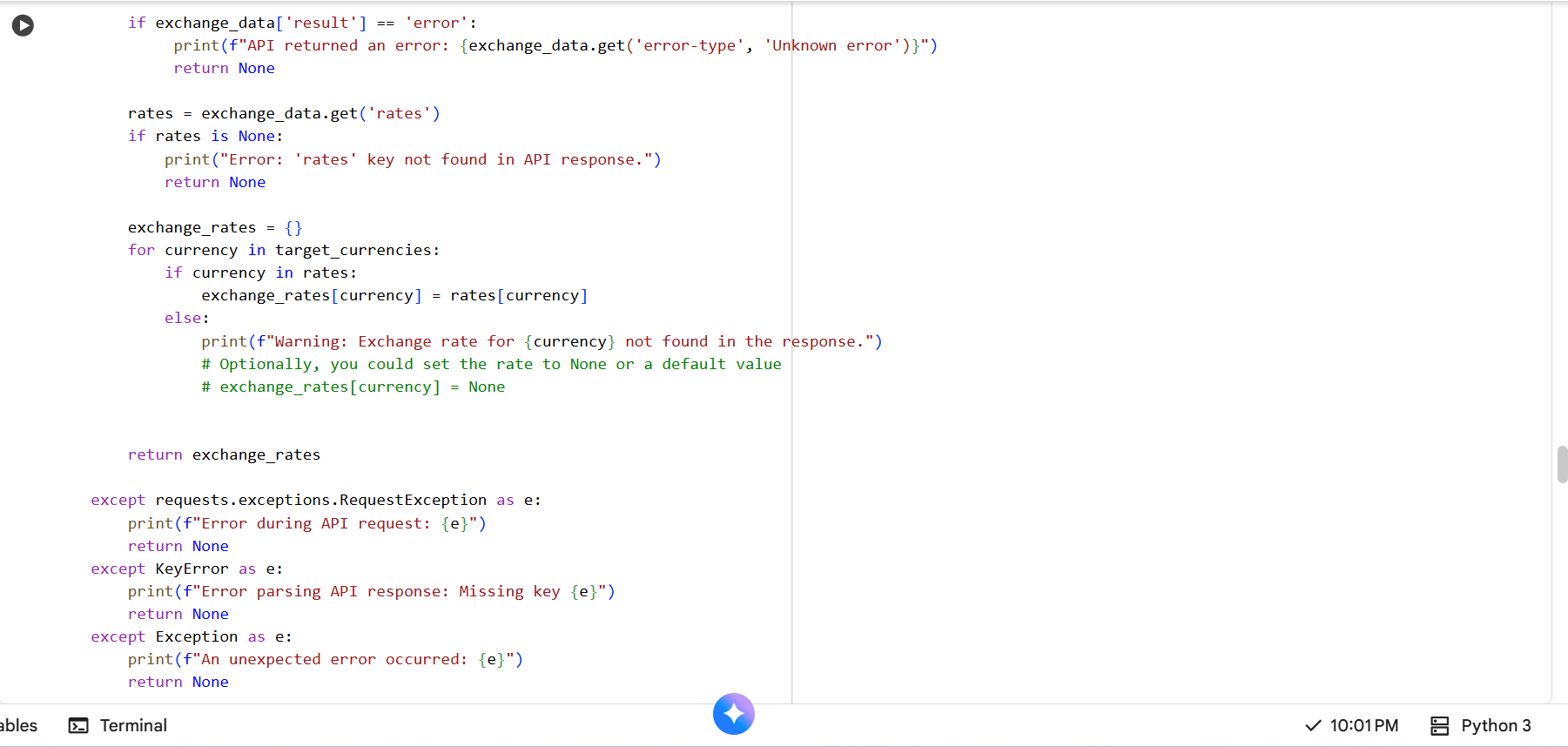
**Task-2: Currency Exchange Rates**

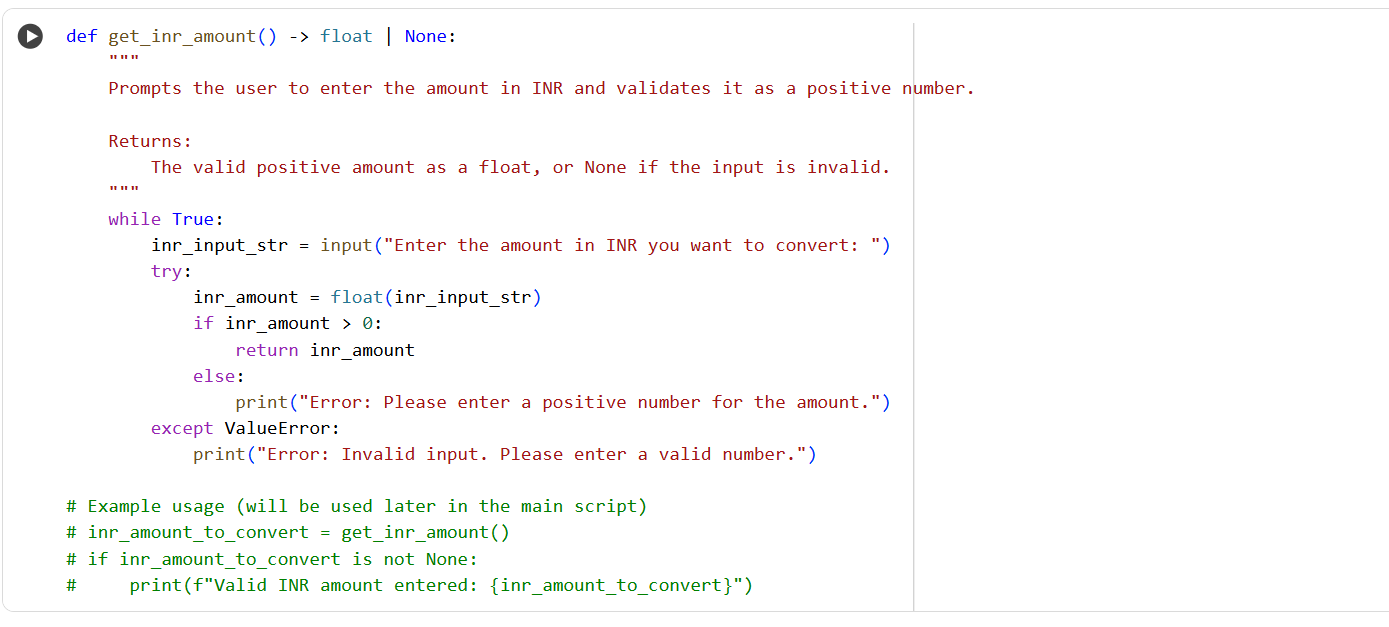
**Prompt:**

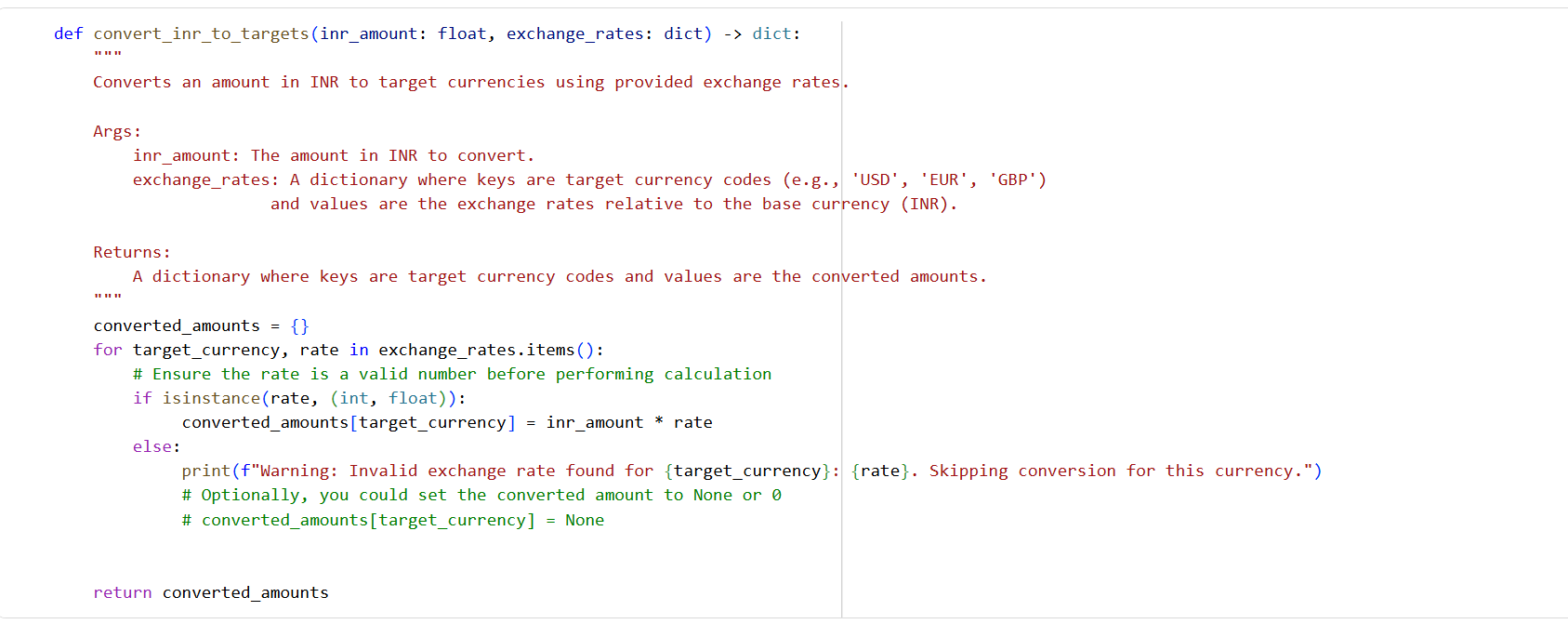
Generate a Python script that uses a Currency Exchange API to convert a user-entered amount in INR to USD, EUR, and GBP. Use the requests library to make the API call. Validate the user input for amount and currency codes. Handle errors such as invalid responses, incorrect currency codes, and API downtime. Display the conversion results in a clear tabular format.

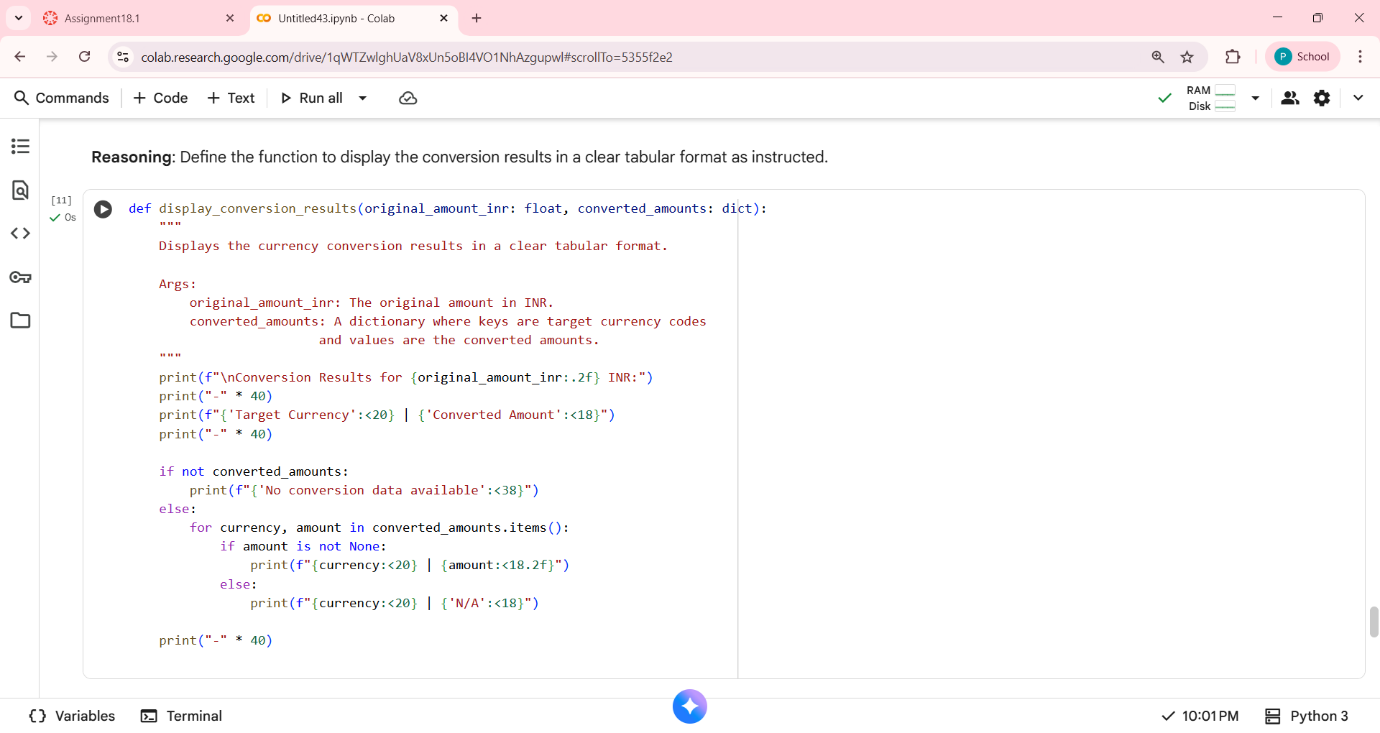












**Explanation:**

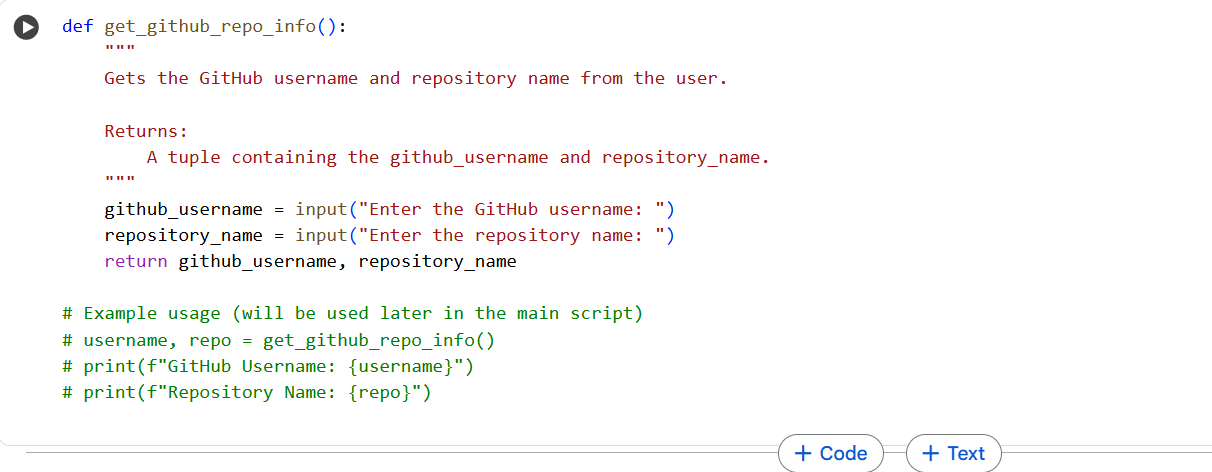
Installing the requests library, providing instructions for obtaining and securely storing an API key, creating a function to fetch exchange rates with error handling, a function to validate user input for the INR amount, a function to convert INR to target currencies using the fetched rates, and finally, a function to display the conversion results in a table. The next steps involve integrating these functions into a complete script and implementing secure API key storage.

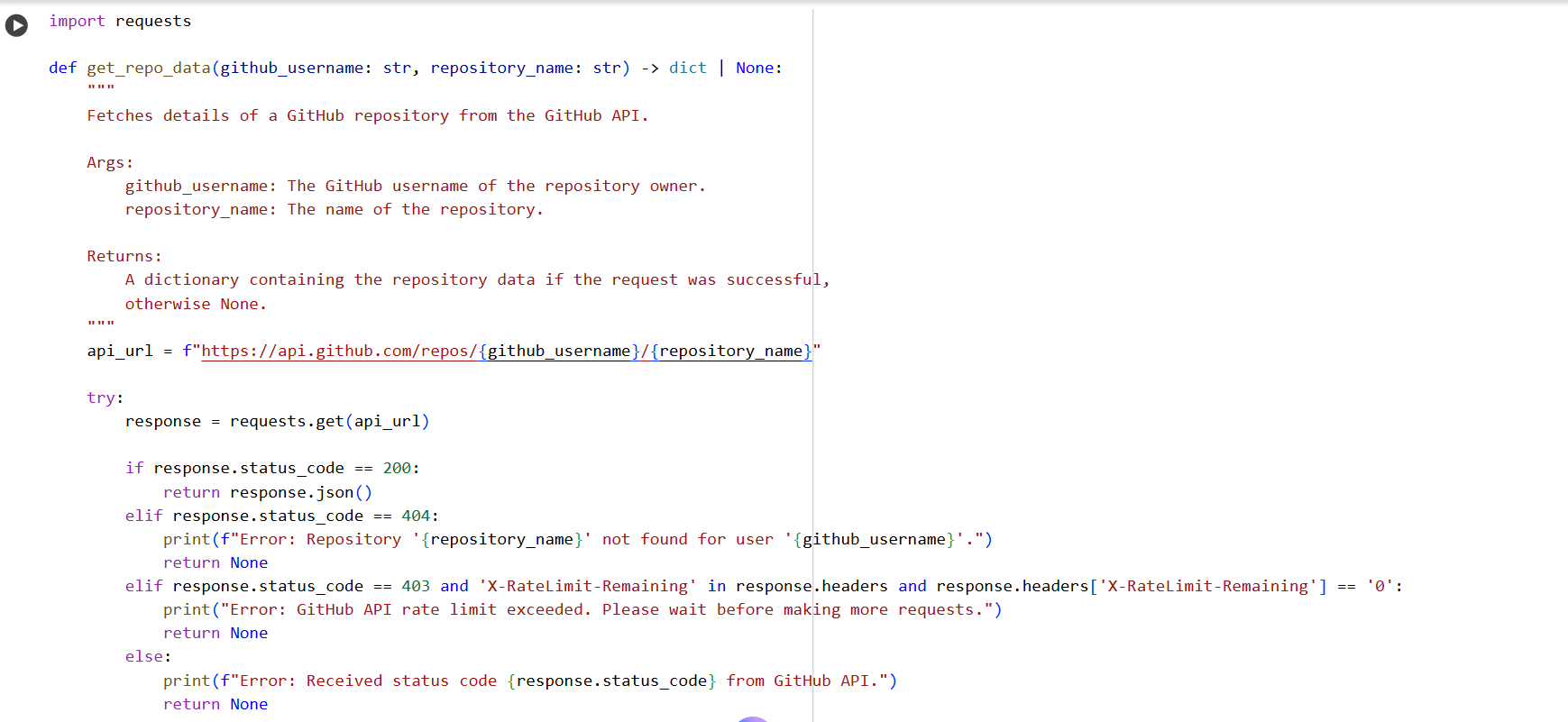
**Task 3 – GitHub Repository Info Fetcher**

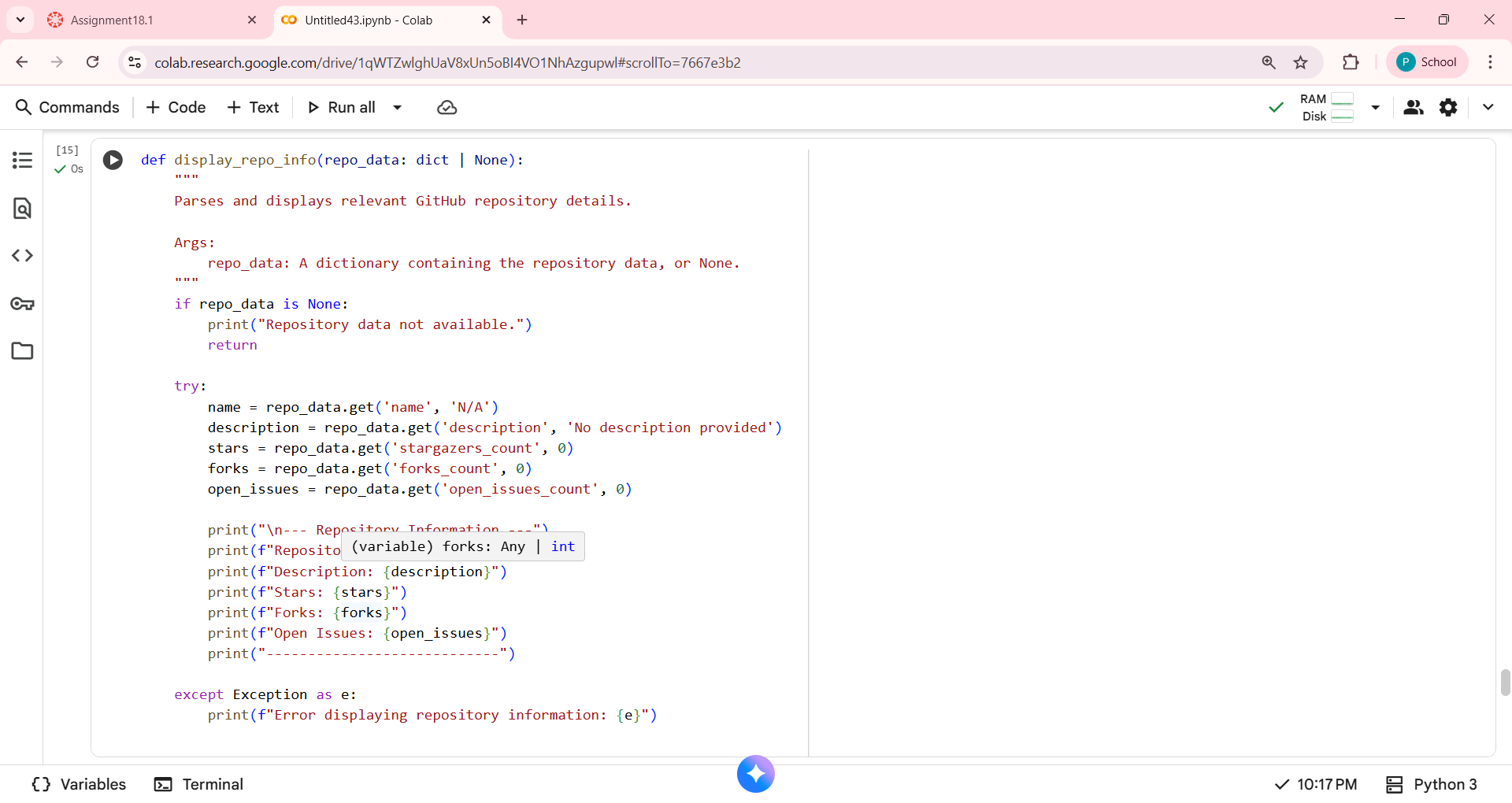
**Prompt:**

Write a Python program using the requests library that connects to the GitHub API and fetches details of a given repository. The program should send a GET request to the GitHub API, handle errors such as rate limits, 404 (repository not found), and invalid input. It should display the repository name, description, number of stars, forks, and open issues in a readable format.











**Explanation:**

 Installing the requests library, obtaining user input for the repository, creating a function to fetch data with robust error handling (including 404 and rate limits), and a function to display the information clearly. The next step involves integrating these functions into a main script for execution.

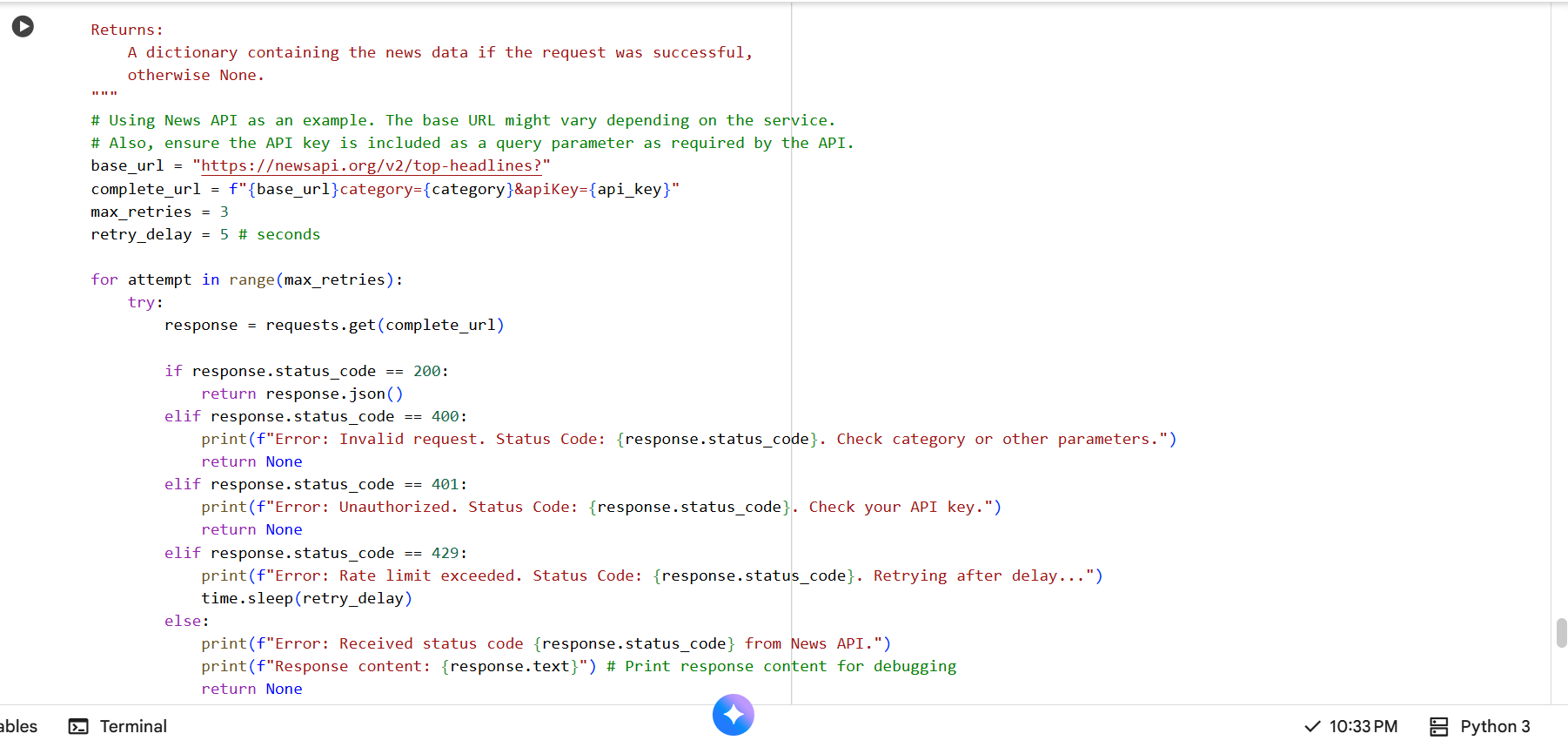
**Task 4 – Real-Time Application: News Headlines Aggregator**

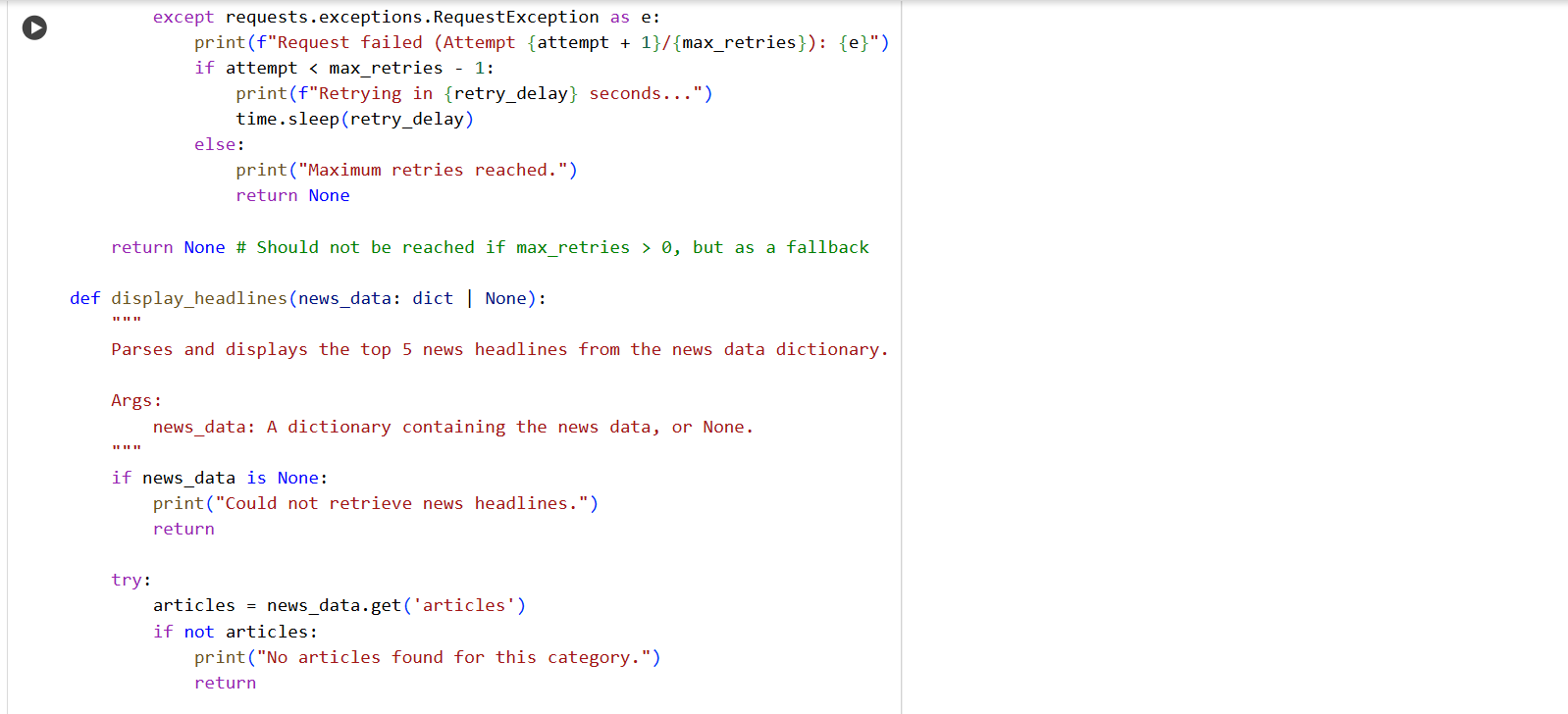
**Prompt:**

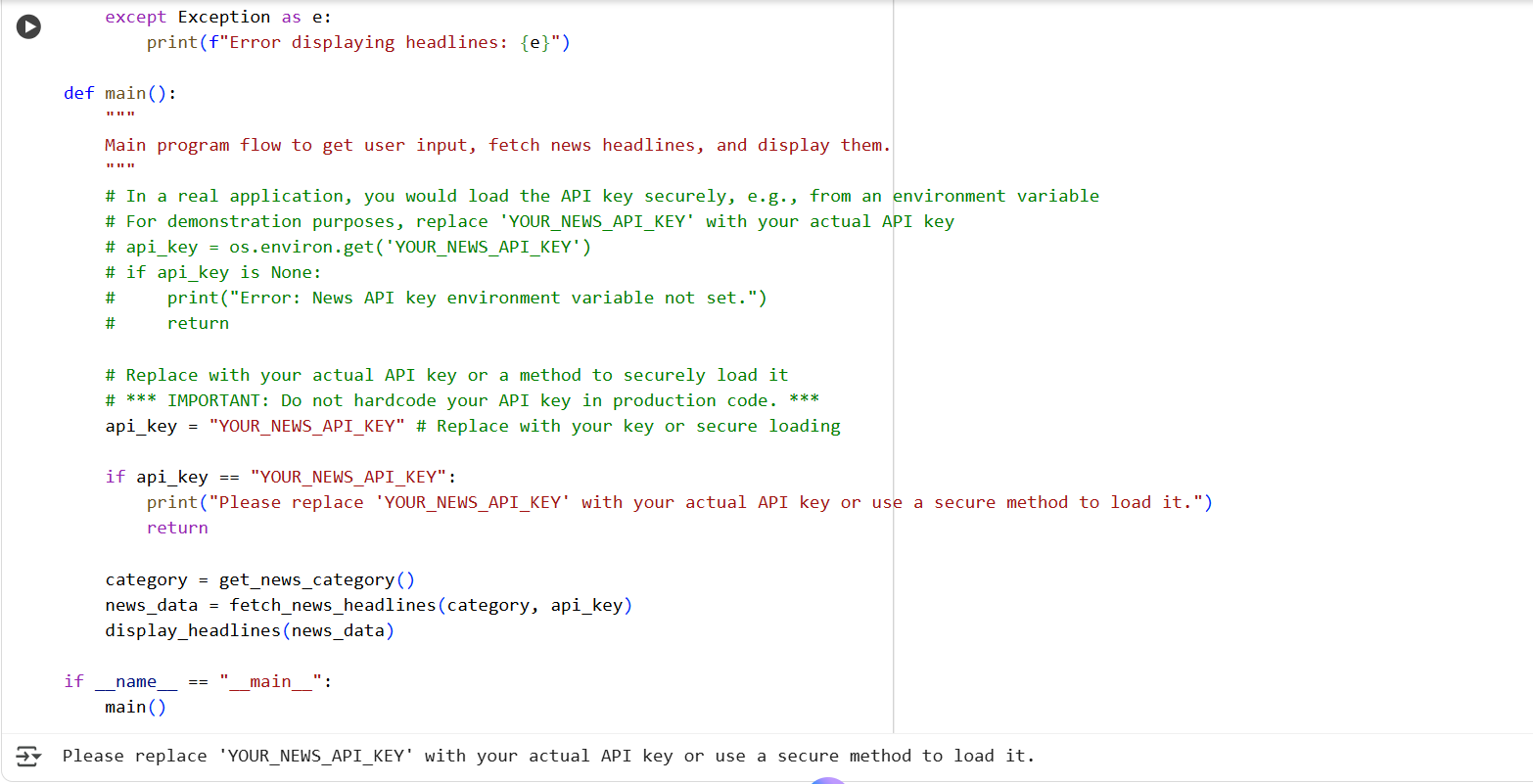
Write a Python script that uses a free News API to fetch the top 5 headlines for a user-specified category (such as sports, technology, or health). Use the requests library to make the API call. Validate the category input, handle errors like invalid category, missing API key, and request failures. Implement a retry mechanism if the first request fails. Display the headlines in a clean, numbered list format.











**Explanation:**

It prompts for a category, validates the input, and retrieves the top 5 headlines using the API key. It includes error handling and a retry mechanism for API calls. The fetched headlines are then displayed in a numbered list.