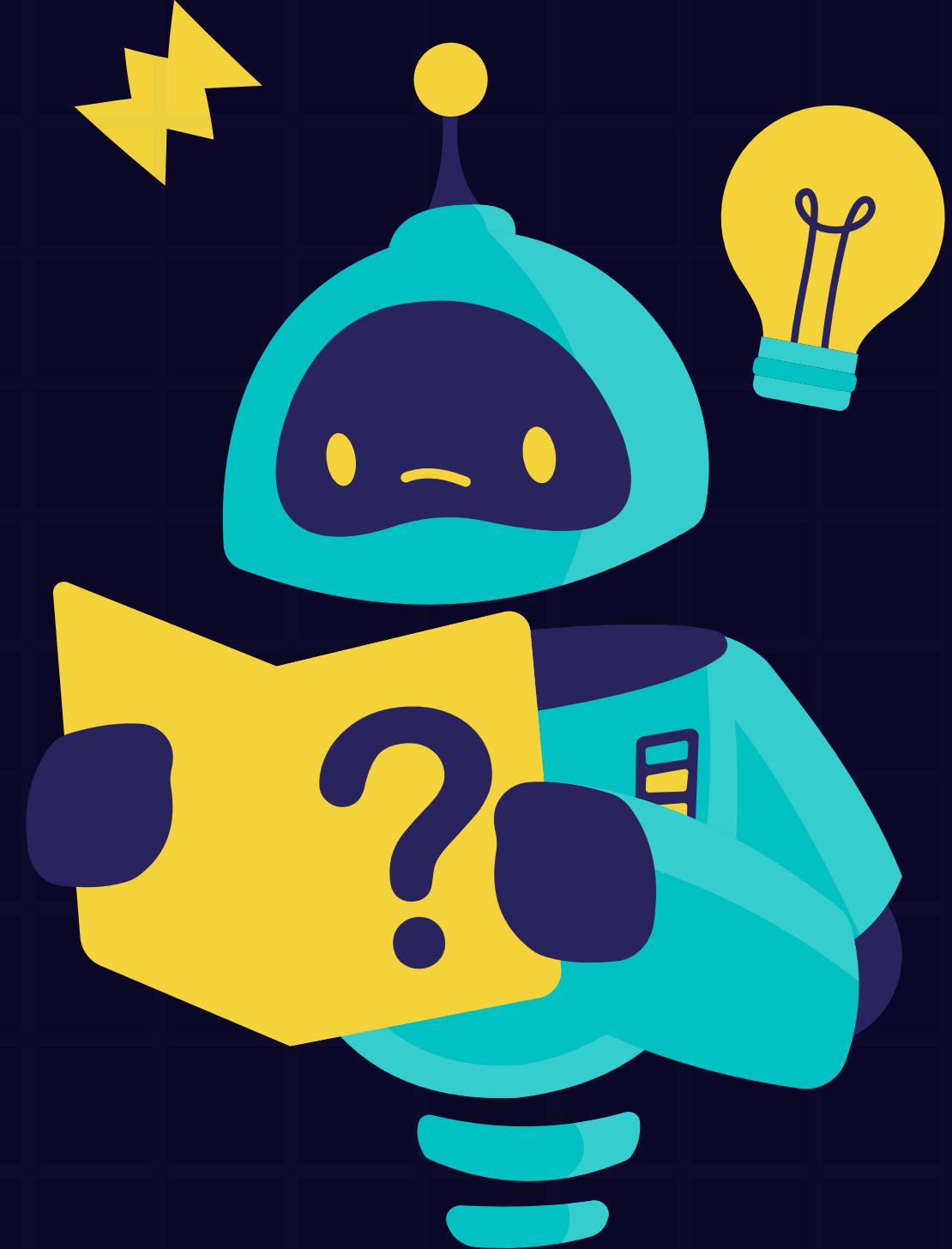




Actual Weather Forecast Web Scrapping base on Location Input with CustomTkinter GUI

College Dropouts Team





Brief Description

Our project builds a tool to scrape real-time weather data from weather.com, including current conditions, hourly, and daily forecasts. It features an interactive graphical interface and allows users to export data to CSV files for accurate weather updates and effective planning.



Library that we use

- BeautifulSoup (for Web Scraping).
- Geopy (for Geolocation).
- Pandas (for Data Handling).
- Tkinter (for GUI).
- CustomTkinter (for Enhanced GUI).
- TkinterMapView (for Maps Integration).
- Datetime (for date and time).
- Regular Expressions (re module).
- pyinstaller (convert the code to .exe)





Result Achieved

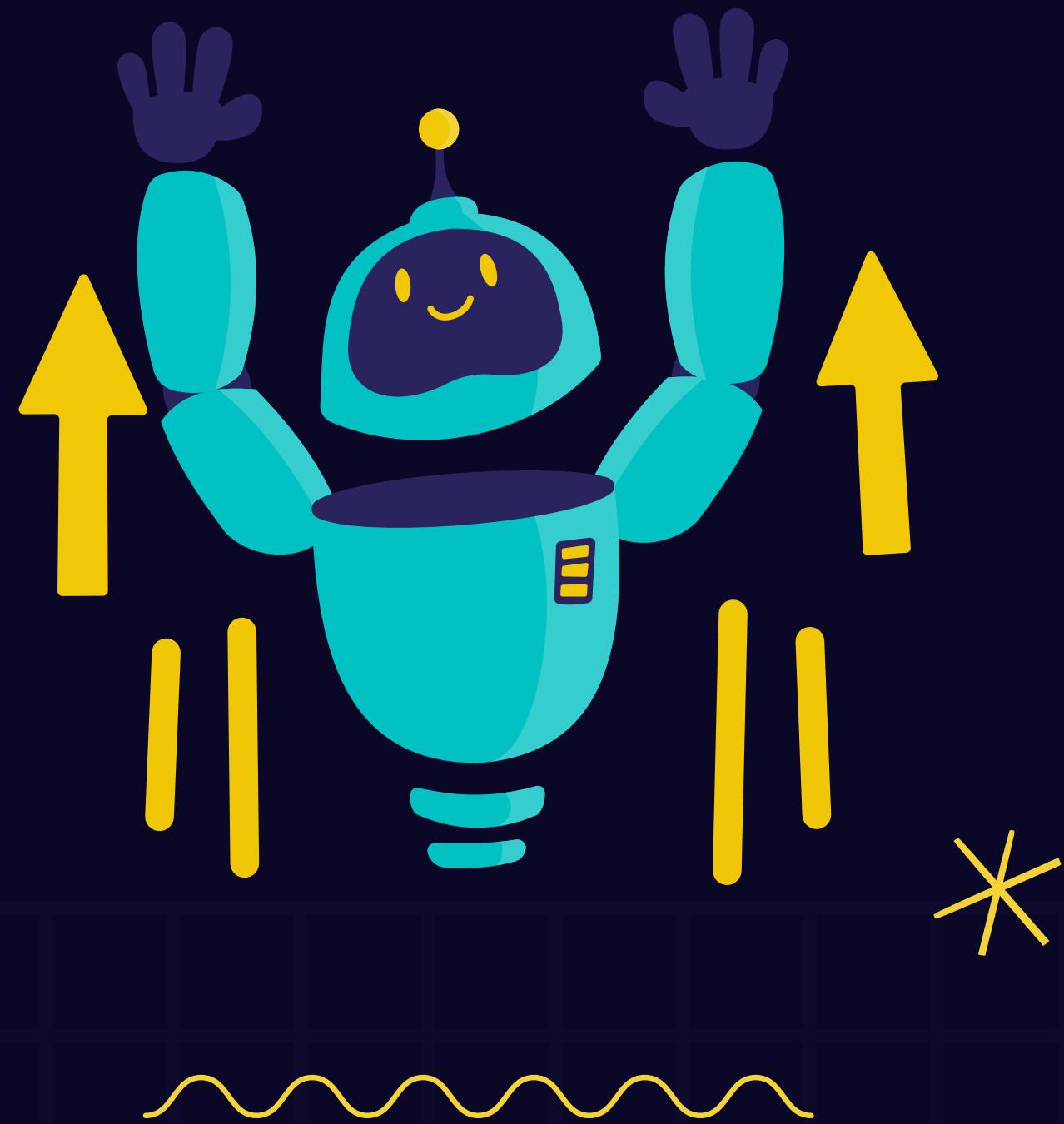
Our program converts locations to coordinates with Geopy, scrapes weather data from weather.com using BeautifulSoup, and converts temperatures to Celsius. It organizes data into Pandas DataFrames, displays it in a Tkinter/CustomTkinter GUI with real-time map integration, and allows CSV exports. The program is also available as an .exe file, so users can run it without installing Python.



Implementation *

get_lat_long(location_name):

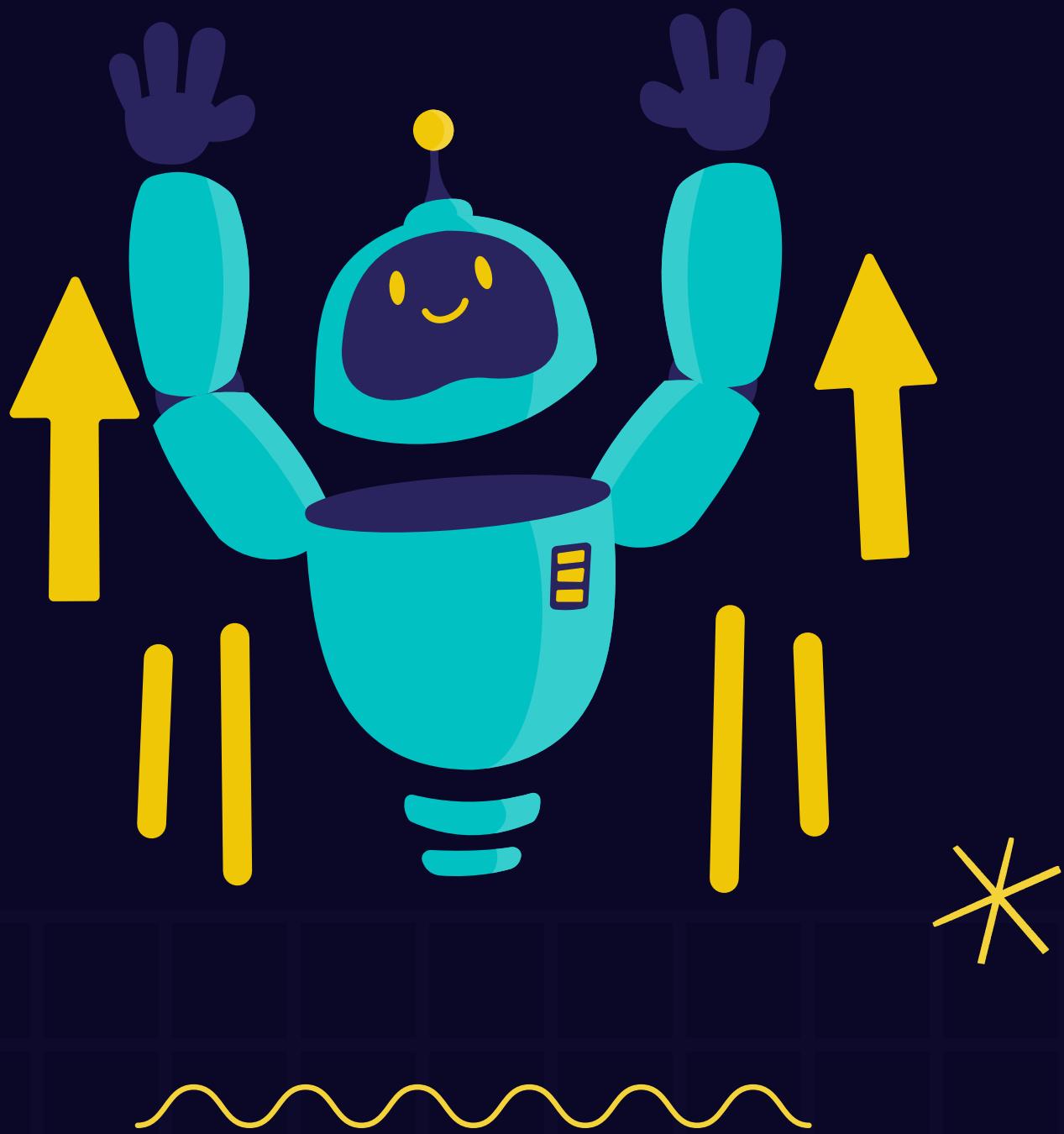
- Description: This function gets the latitude and longitude of a given location name.
- Fields: Uses the Nominatim geocoder from the geopy library.
- Methods: geolocator.geocode(location_name): Returns a location object with latitude and longitude.
- Parameters: location_name (str) – the name of the location to geocode.
- Returns: Tuple (latitude, longitude) or (None, None) if location not found.



Implementation *

extract_numeric_temperature(temperature_string)

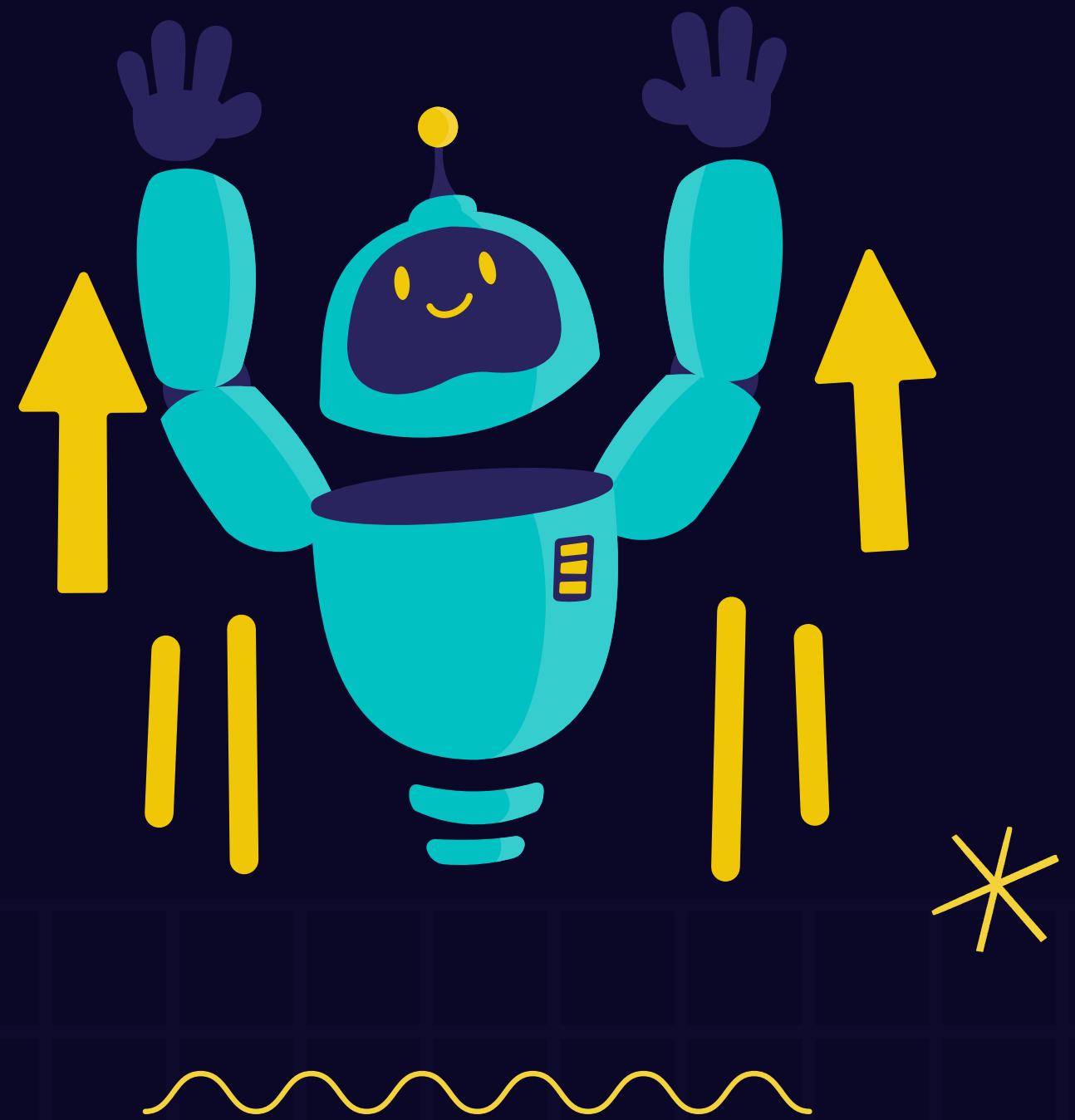
- Description: Extracts numeric temperature value from a string.
- Fields: None.
- Methods: String manipulation and list comprehension to filter digits and period.
- Parameters: temperature_string (str) – the temperature string.
- Returns: Numeric temperature as an integer (or None if not found).



Implementation *

convert_fahrenheit_to_celsius (fahrenheit)

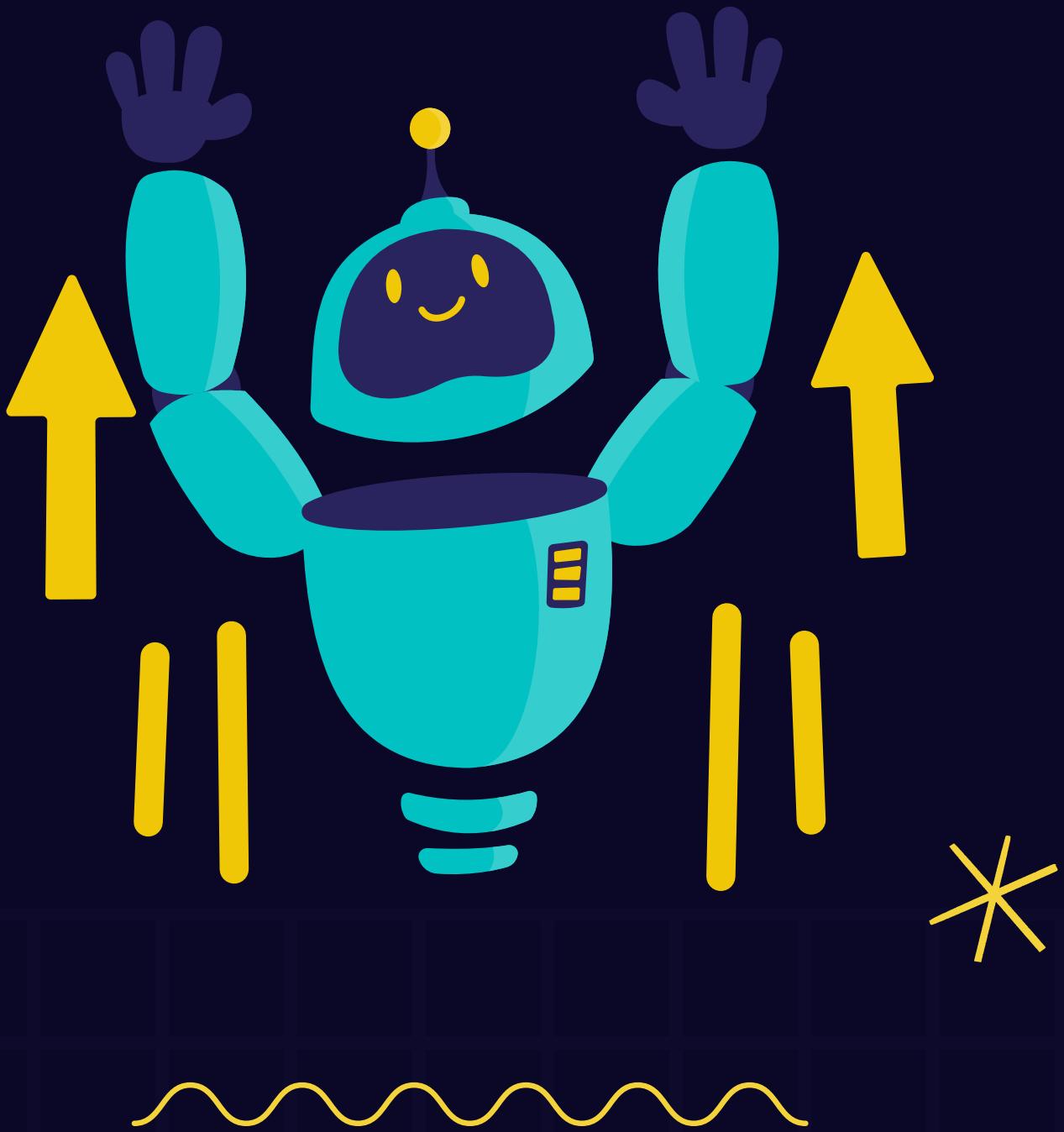
- Description: Converts a temperature from Fahrenheit to Celsius.
- Fields: None.
- Methods: Uses the formula $Celsius = \frac{5}{9}(Fahrenheit - 32)$.
- Parameters: fahrenheit (int/float) – temperature in Fahrenheit.
- Returns: Temperature in Celsius as an integer (or None if input is None).



Implementation *

get_today_place_forecast(soup):

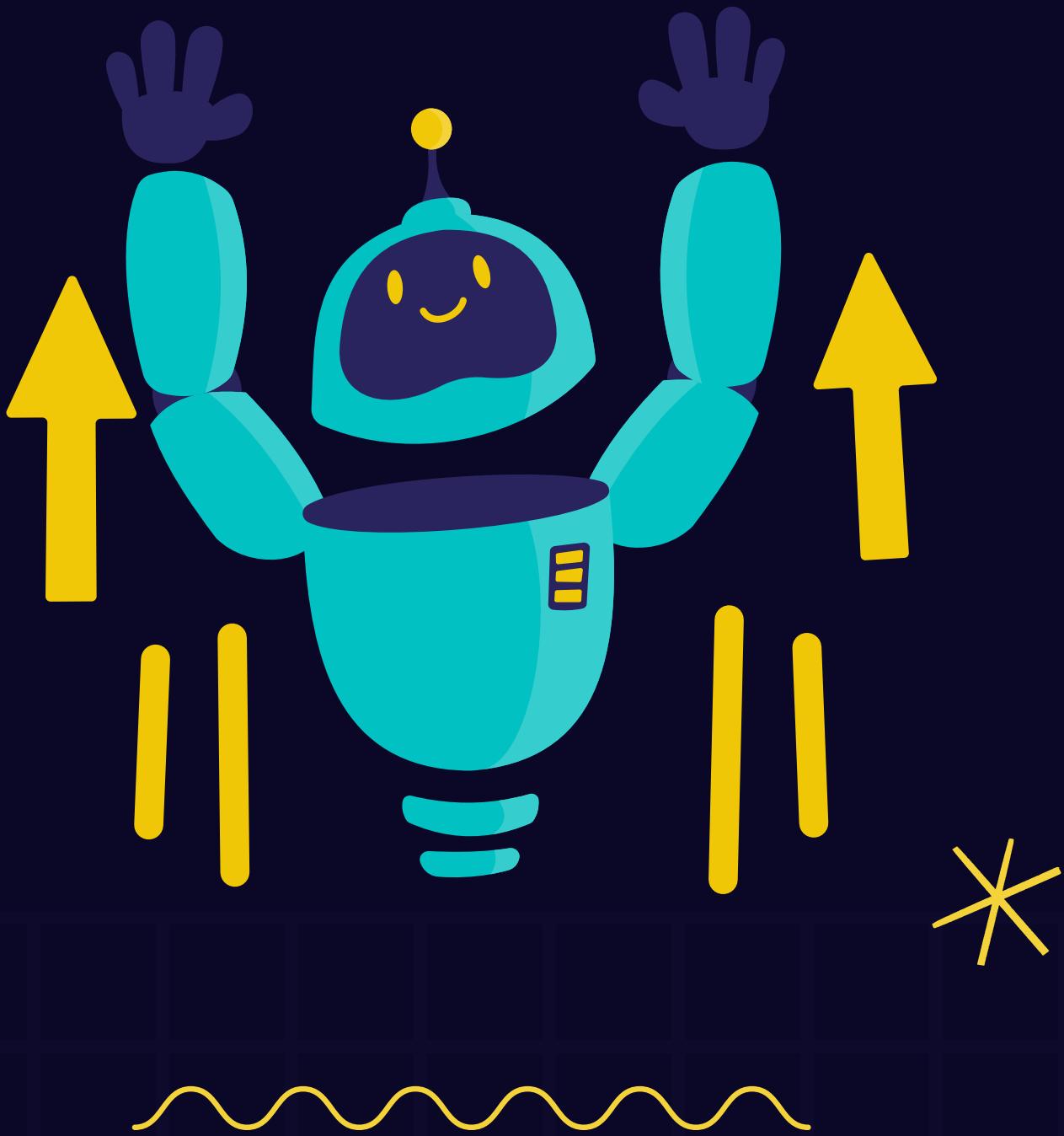
- Description: Extracts the current day's forecast from the HTML soup.
- Fields: Uses BeautifulSoup to parse HTML content.
today_forecast: the div containing today's weather forecast.
- Methods: soup.find and soup.find_all: Extract specific HTML elements. pd.DataFrame: Constructs a DataFrame to store the forecast data.
- Parameters: soup (BeautifulSoup object) – parsed HTML content.
- Returns: Tuple (header_element, df_today_forecast) – the header string and a DataFrame of today's forecast.



Implementation *

today's_weather(soup)

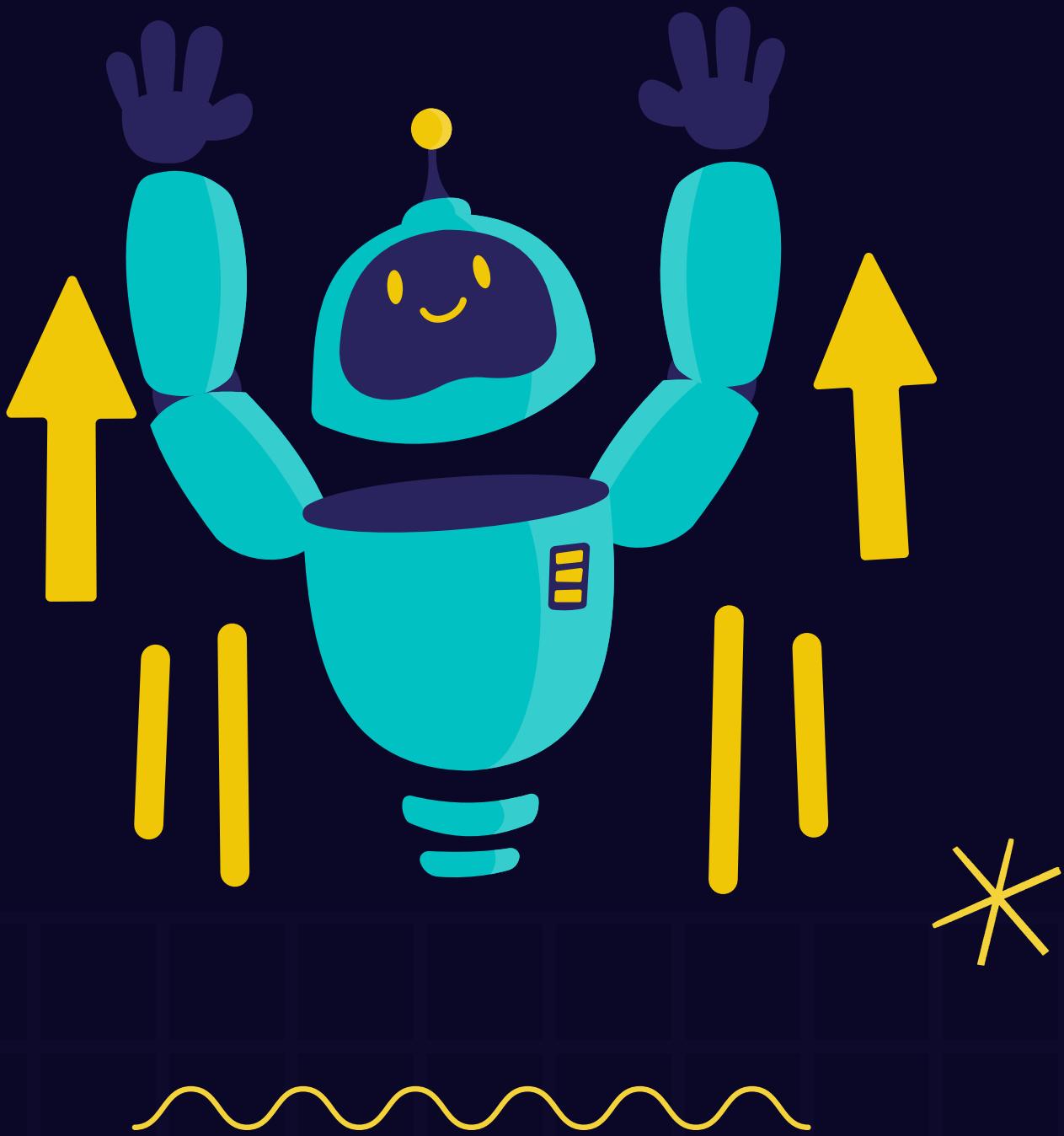
- Description: Extracts today's detailed weather information.
- Fields: Uses BeautifulSoup to parse HTML content.
weather_today: the div containing today's weather details.
- Methods: soup.find and soup.find_all: Extract specific HTML elements. pd.DataFrame: Constructs a DataFrame to store weather data.
- Parameters: soup (BeautifulSoup object) – parsed HTML content.
- Returns: Tuple (header_element, df_weather_today) – the header string and a DataFrame of weather details.



Implementation *

hourly_forecast(soup)

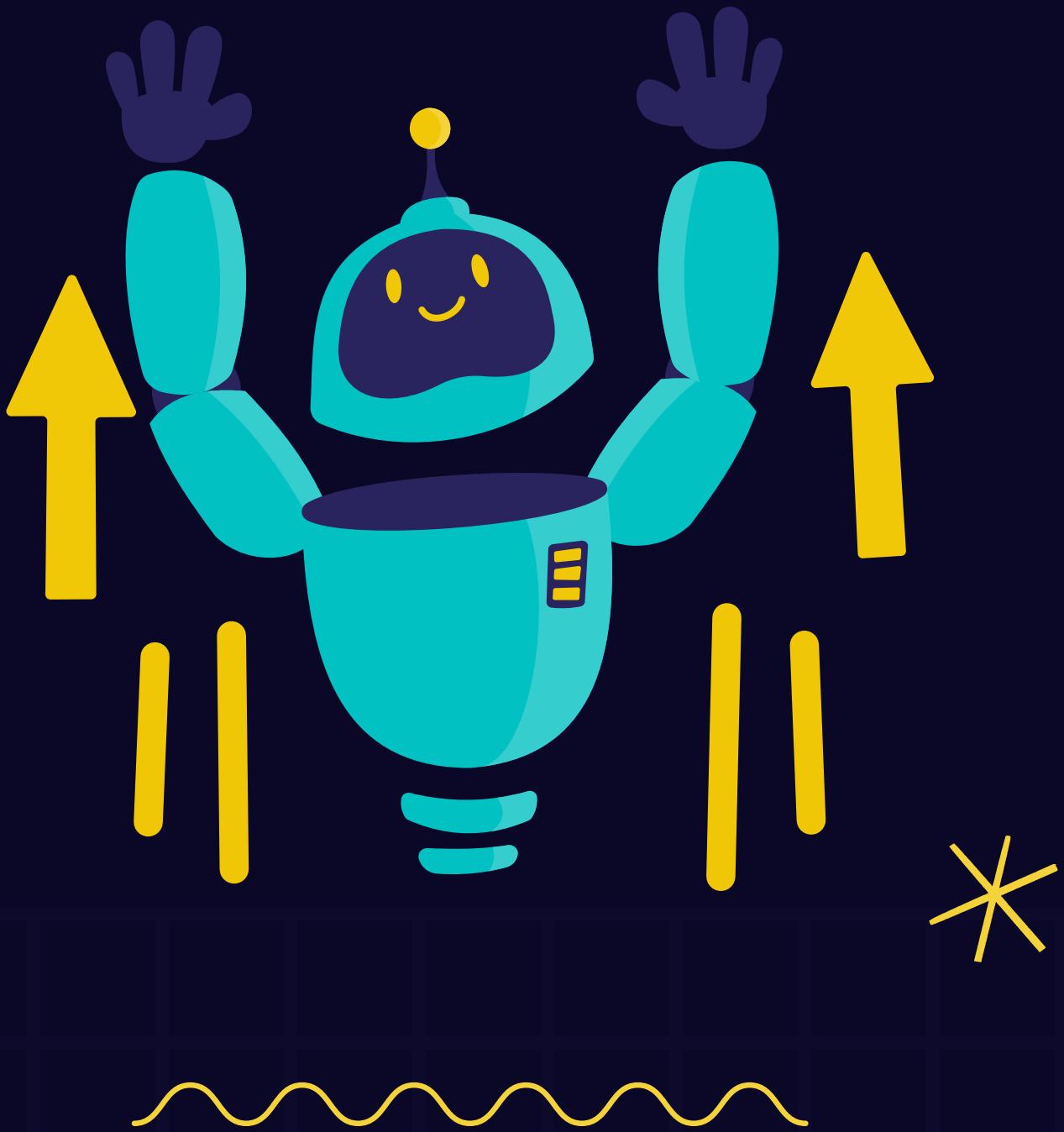
- Description: Extracts hourly weather forecast.
- Fields: Uses BeautifulSoup to parse HTML content.
hourly_forecast: the div containing hourly forecast.
- Methods: soup.find and soup.find_all: Extract specific HTML elements. pd.DataFrame: Constructs a DataFrame to store the hourly forecast data.
- Parameters: soup (BeautifulSoup object) – parsed HTML content.
- Returns: Tuple (header_element, df_hourly_forecast) – the header string and a DataFrame of the hourly forecast.



Implementation *

daily_forecast(soup)

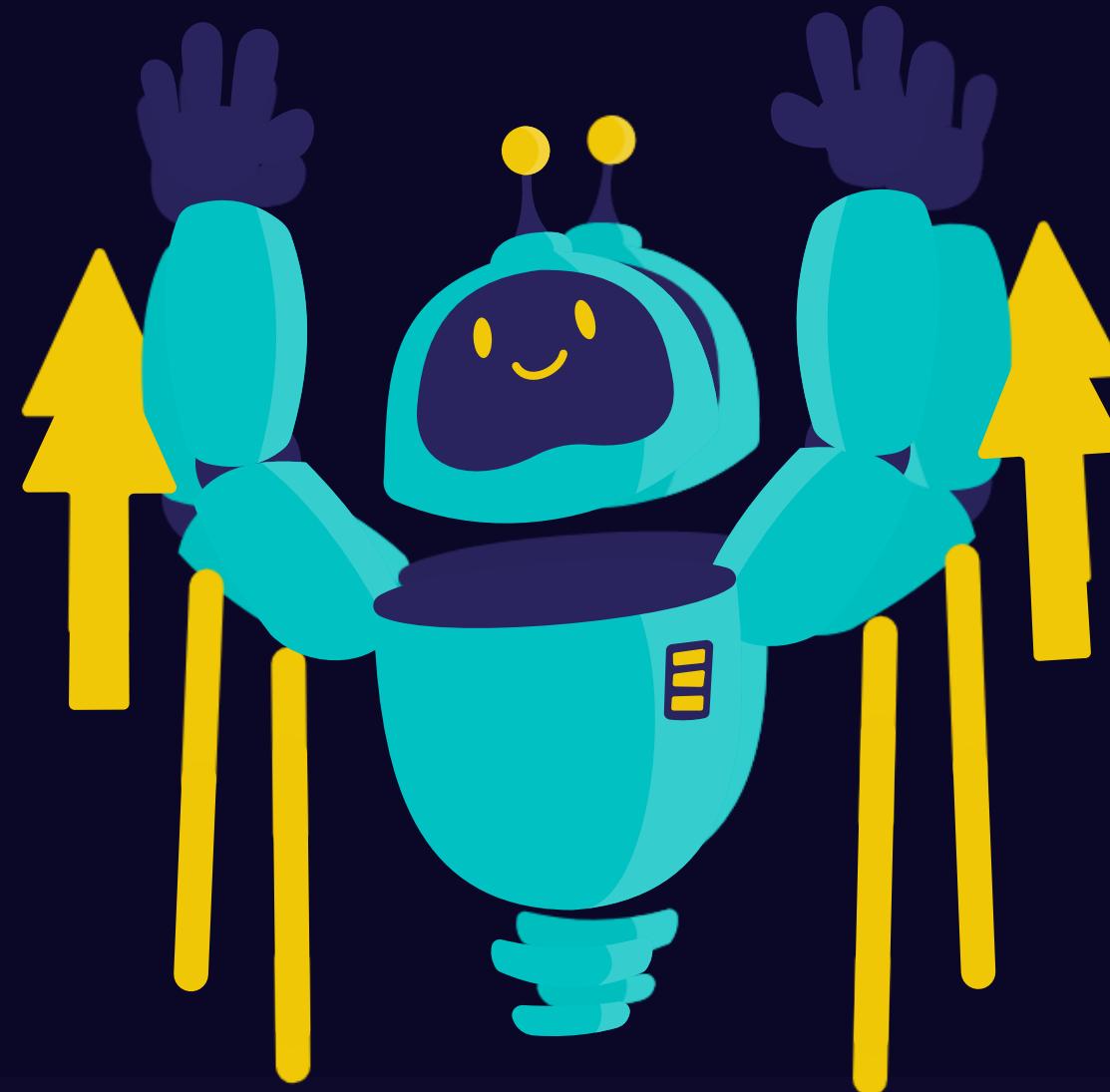
- Description: Extracts daily weather forecast.
- Fields: Uses BeautifulSoup to parse HTML content.
daily_forecast: the div containing daily forecast.
- Methods: soup.find and soup.find_all: Extract specific HTML elements. pd.DataFrame: Constructs a DataFrame to store the daily forecast data.
- Parameters: soup (BeautifulSoup object) – parsed HTML content.
- Returns: Tuple (header_element, df_daily_forecast) – the header string and a DataFrame of the daily forecast.



Implementation *

get_weather(latitude, longitude)

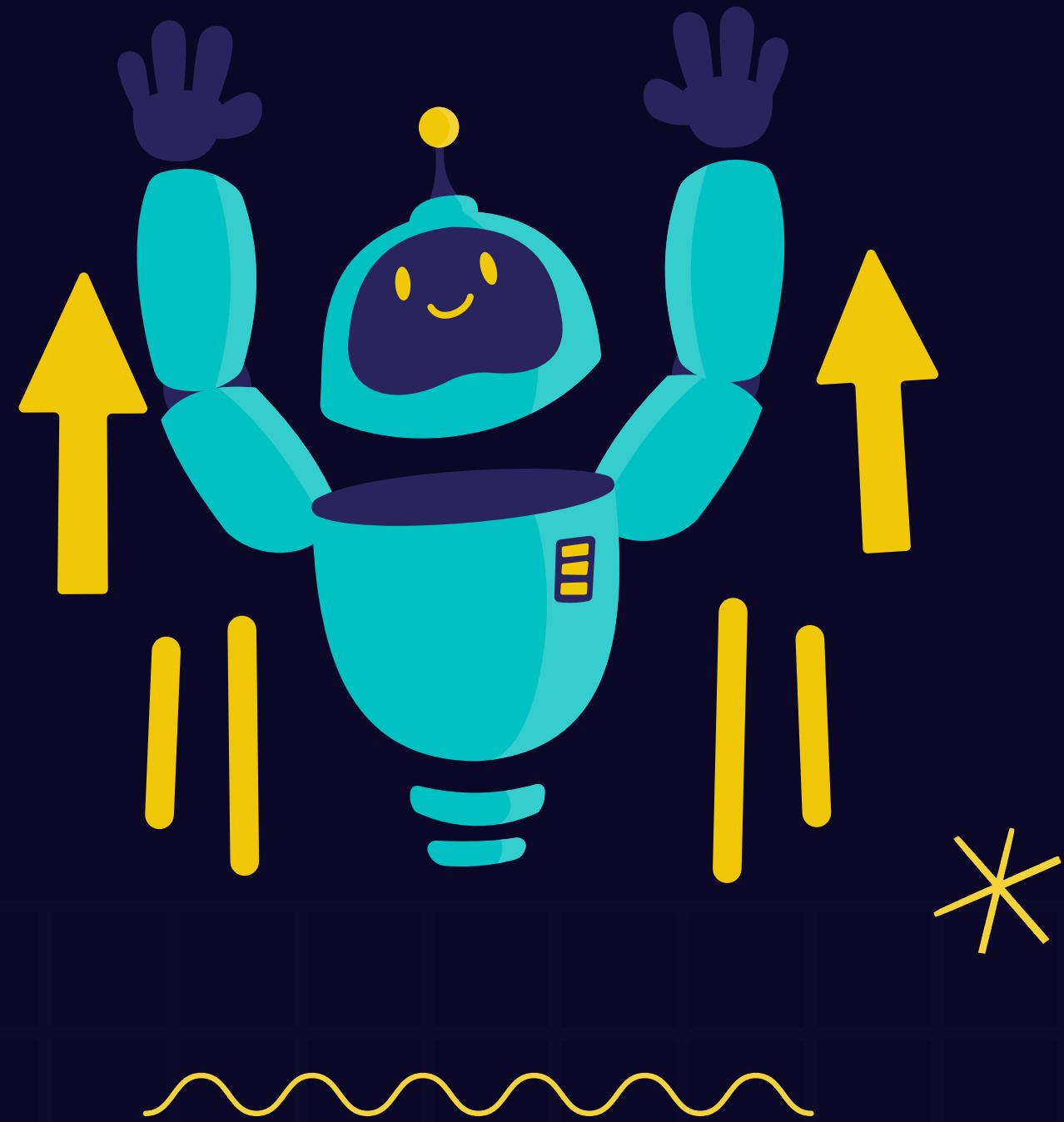
- Description: Fetches weather data for a specific latitude and longitude.
- Fields: Uses requests to fetch data from a URL. Uses BeautifulSoup to parse HTML content. Various data extraction fields (e.g., temperature_today, condition_span).
- Methods: requests.get(url): Fetches the web page.
- BeautifulSoup(response.text, "html.parser"): Parses the HTML.
- Calls to helper functions (get_today_place_forecast, todays_weather, hourly_forecast, daily_forecast).
- Parameters: latitude (float) – latitude of the location, longitude (float) – longitude of the location.
- Returns: String with concatenated weather details.



Implementation *

export_to_csv()

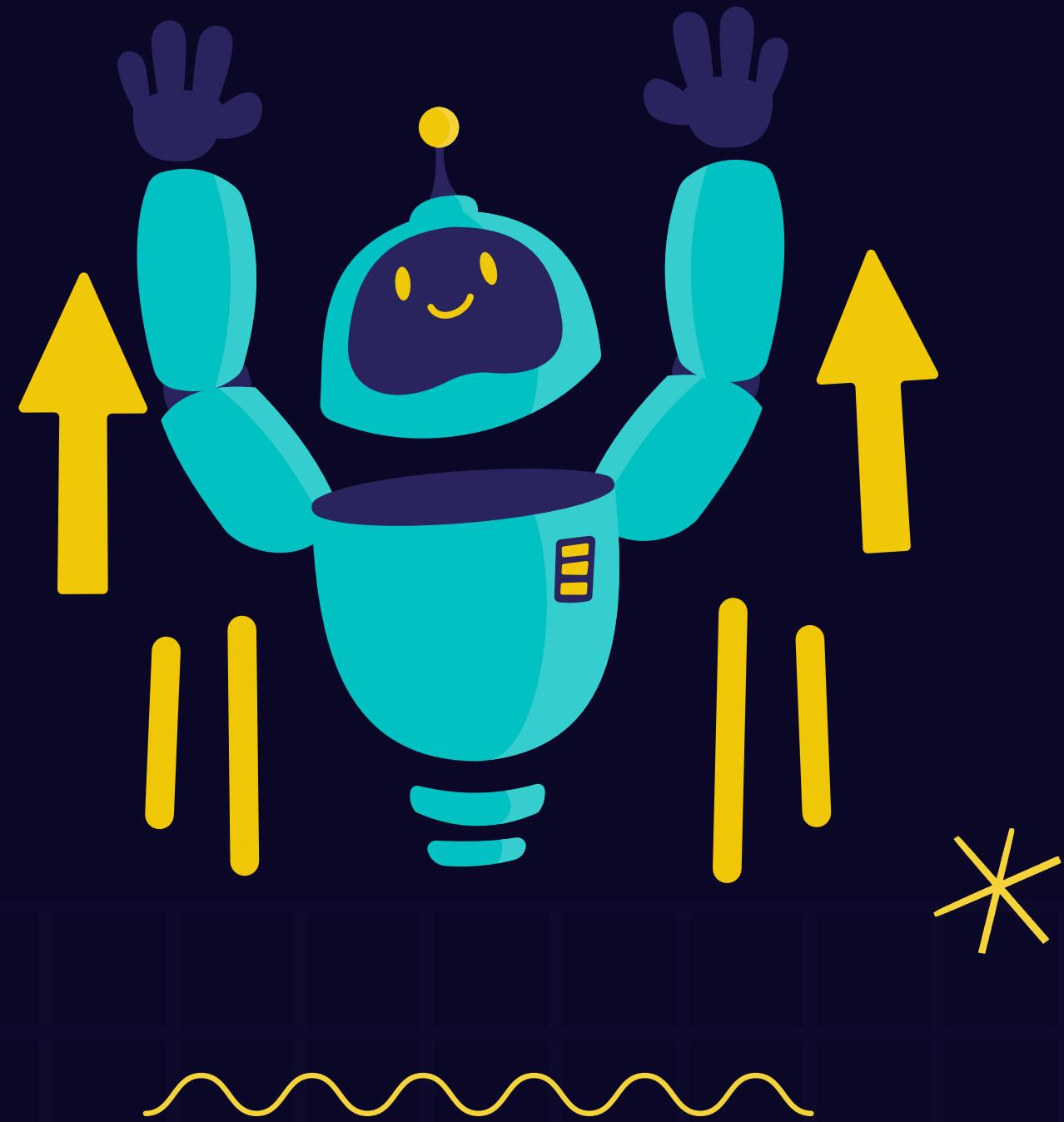
- Description: Exports weather data to CSV files.
- Fields: Uses data_to_save to get data.
- Methods: Iterates over data_to_save to save each DataFrame as a CSV file. df.to_csv(filename): Saves the DataFrame.
- Parameters: None.
- Returns: None (prints confirmation message for each file saved).



Implementation *

dropdown_func(event)

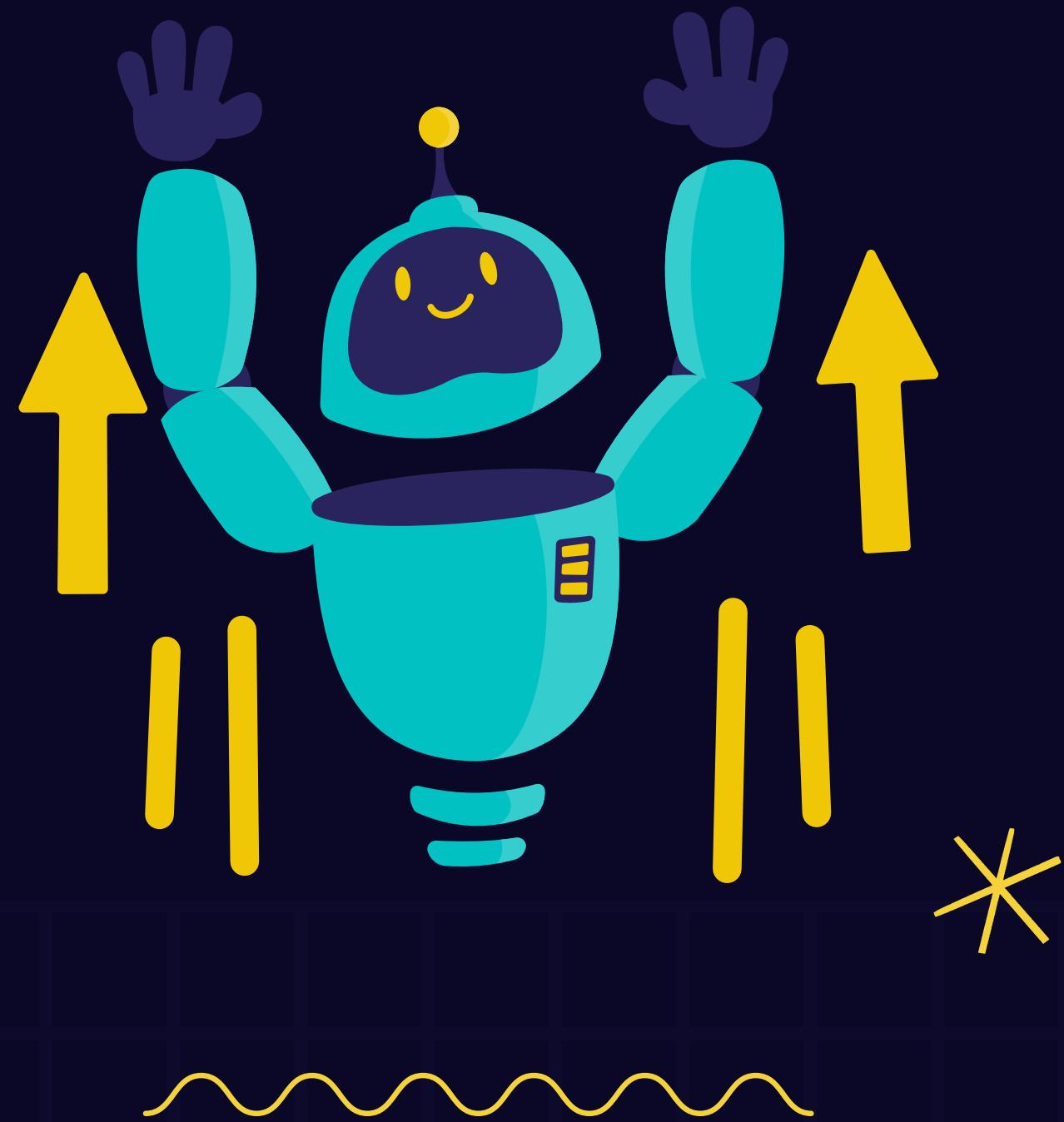
- Description: Handles changes in the dropdown selection for forecasts.
- Fields: Uses global variables to update the UI based on selection.
- Methods: combobox.get(): Gets the selected value from the dropdown. Updates UI labels with forecast data.
- Parameters: event (event object) – the event triggering the function.
- Returns: None.



Implementation *

show()

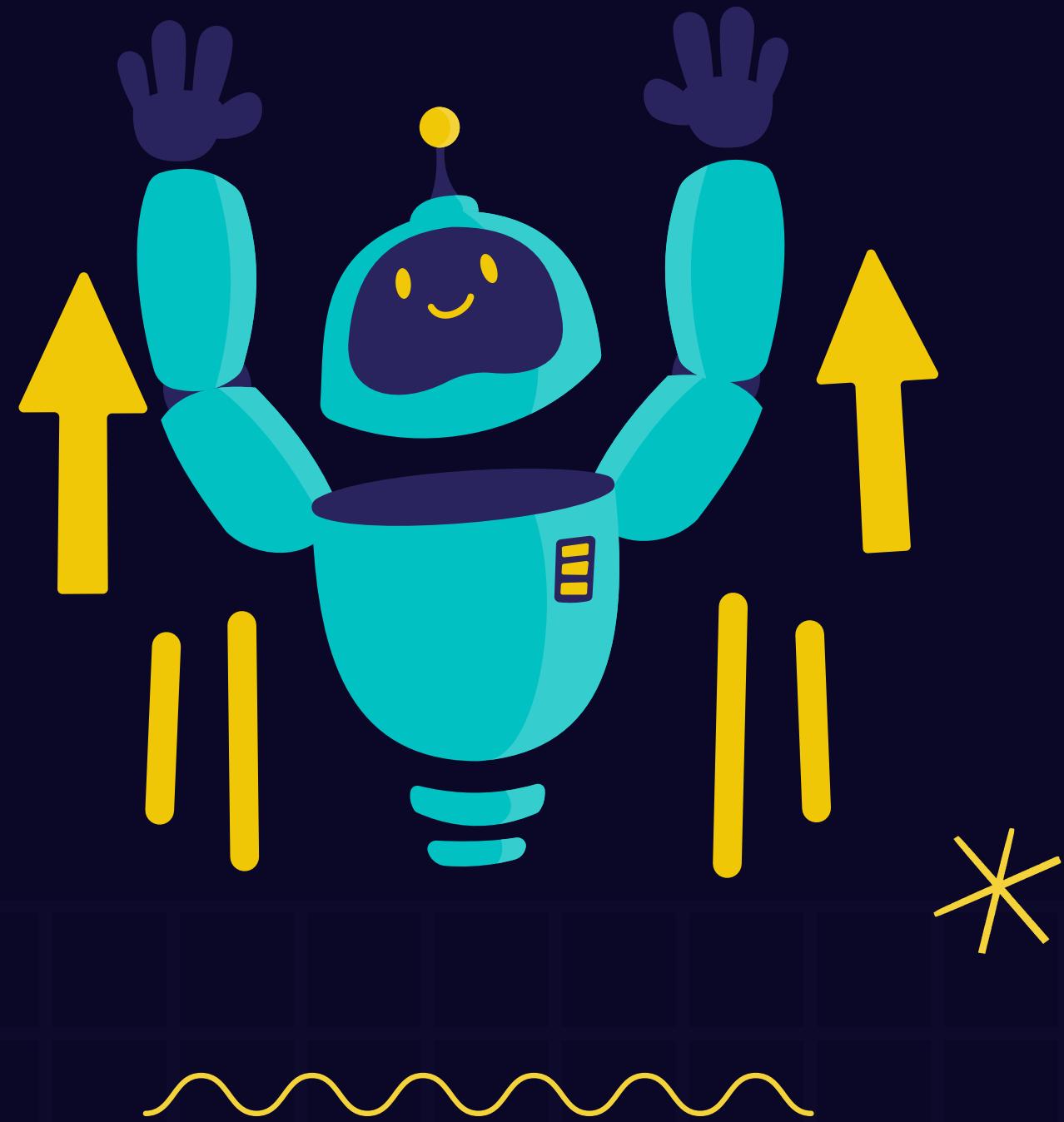
- Description: Displays the weather data on the GUI.
- Fields: Uses various global variables to display data.
- Methods: CTkLabel: Creates labels for displaying data.grid():
Places the labels in the grid layout.
- Parameters: None.
- Returns: None.



Implementation *

get_location()

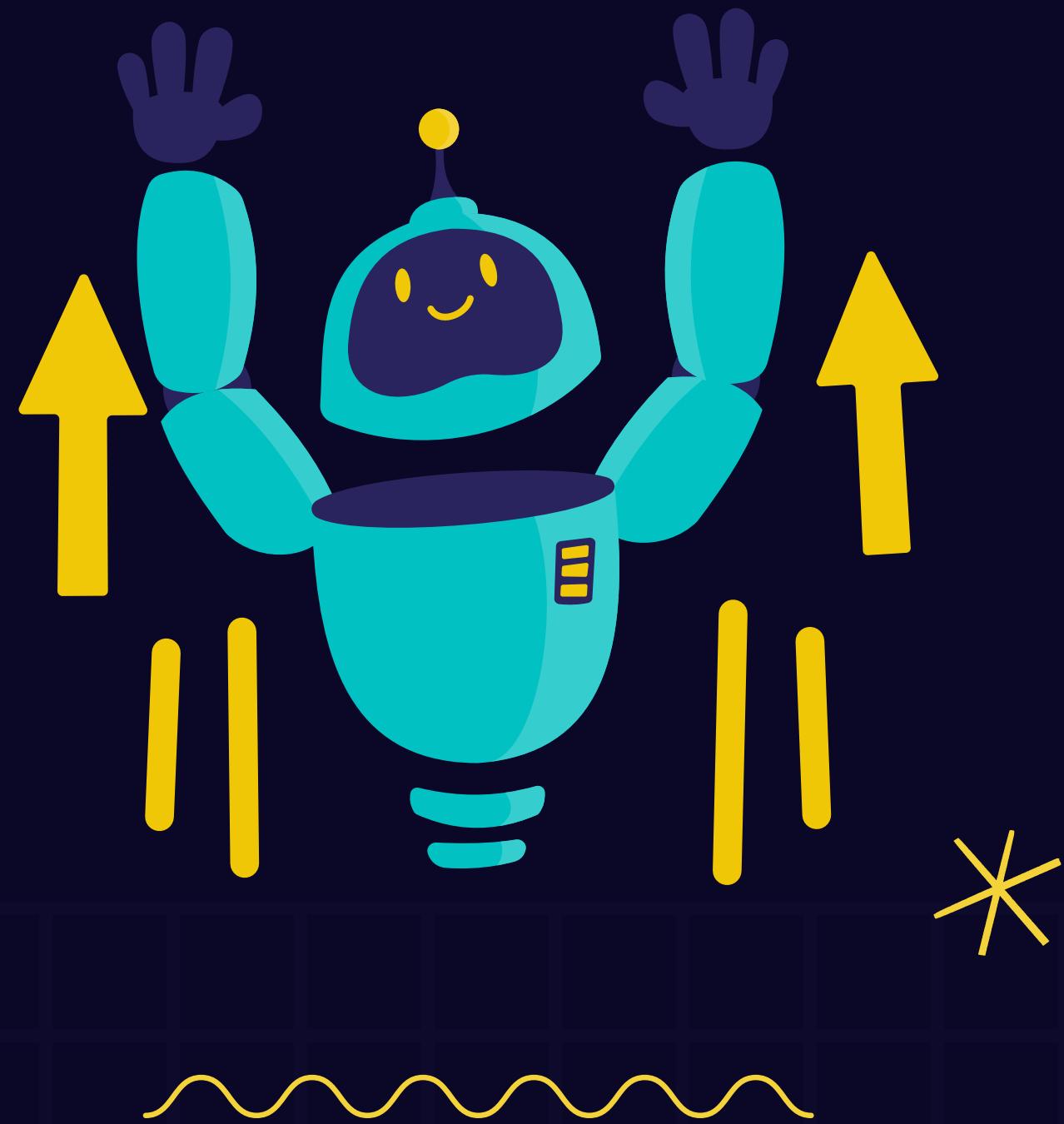
- Description: Retrieves location, fetches weather data, and updates the GUI.
- Fields: Uses global variables for location, latitude, longitude, etc.
- Methods: `location_entry.get()`: Gets the location from the input field.
- Calls `get_lat_long`, `maps`, `get_weather`, and `show`. Uses `requests` and `BeautifulSoup` to parse the HTML content.
- Parameters: None.
- Returns: None.



Implementation *

maps(latitude, longitude)

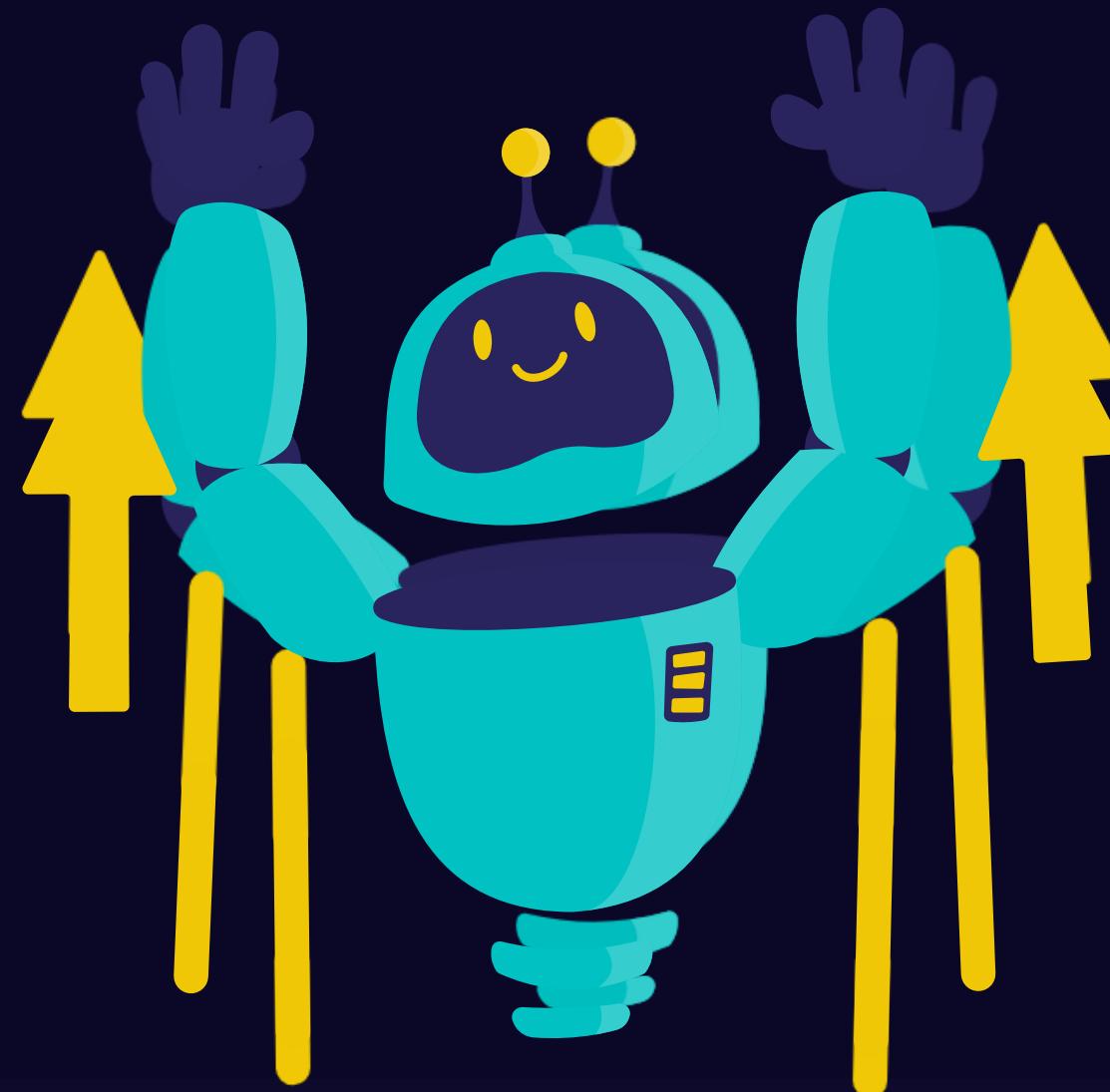
- Description: Displays a map for the given latitude and longitude.
- Fields: Uses TkinterMapView to display the map.
- Methods: TkinterMapView(): Initializes the map view.
- gmap.set_tile_server(): Sets the tile server URL.
- gmap.set_position(): Sets the position of the map marker.
- Parameters: latitude (float) – latitude of the location, longitude (float) – longitude of the location.
- Returns: None.



Implementation *

main()

- Description: Sets up the main GUI layout and event loop.
- Fields: Uses tkinter and customtkinter to create the GUI.
- Methods: Initializes the main Tkinter window and widgets.
Sets up layout using .grid(). Binds event handlers
(get_location, export_to_csv, dropdown_func). Starts the
Tkinter main event loop with root.mainloop().
- Parameters: None.
- Returns: None.



Result (input = Taiwan)



Our Project

The screenshot shows a desktop application window titled "Actual Weather Scrapping". The main input field says "Enter location: Taiwan". Below it, the "Today Condition" section displays the following data for Luona, Nantou County:

High / Low	31°C/22°C
Wind	Wind Direction 3 mph
Humidity	70%
Dew Point	74°
Pressure	Arrow Up 29.83 in
UV Index	4 of 11
Visibility	9 mi
Moon Phase	Waxing Gibbous

Current temperature is 29°C, with a note "Mostly Cloudy". The daily forecast shows temperatures increasing from 29°C in the morning to 31°C by 1 pm. A map of Taiwan highlights Nantou County.

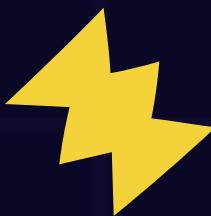
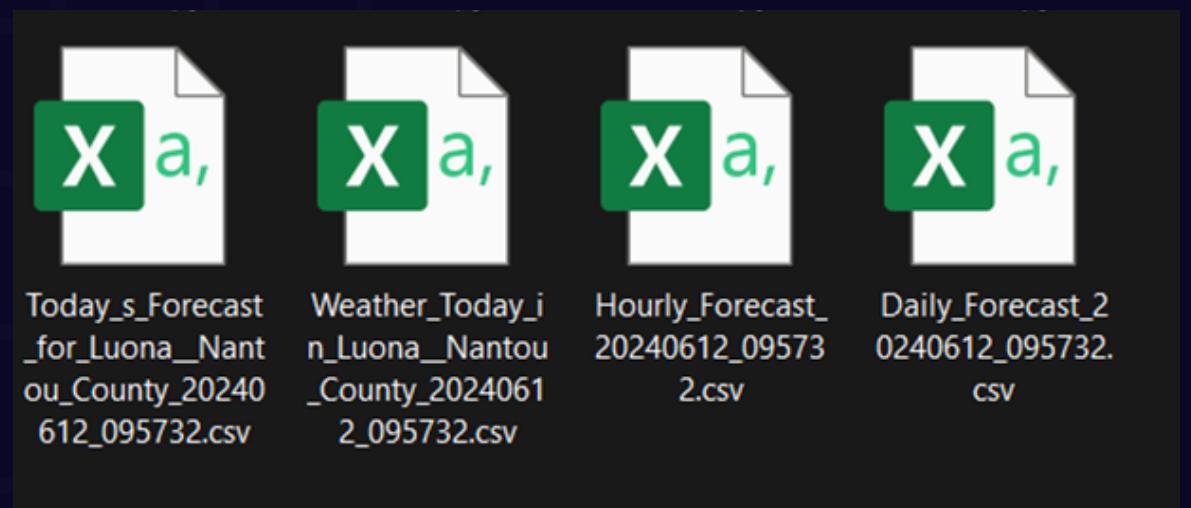
Website

The screenshot shows a web browser window displaying the weather.com website. The search bar shows "weather.com/weather/today/l/23.60.120.84?par=google". The main content area shows the following weather information for Luona, Nantou County:

29° Mostly Cloudy
Day 31° • Night 23°

A radar map of East Asia is visible at the bottom.

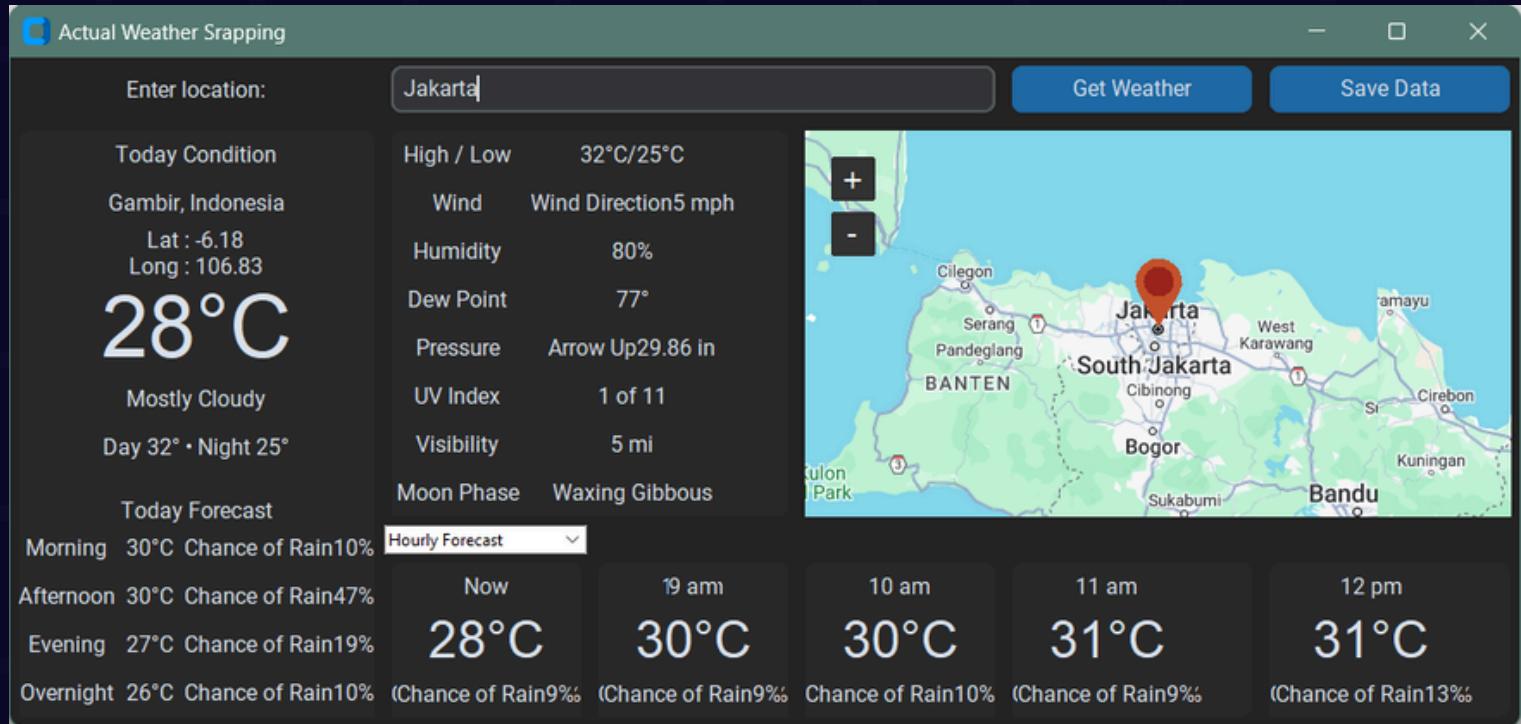
Saved data



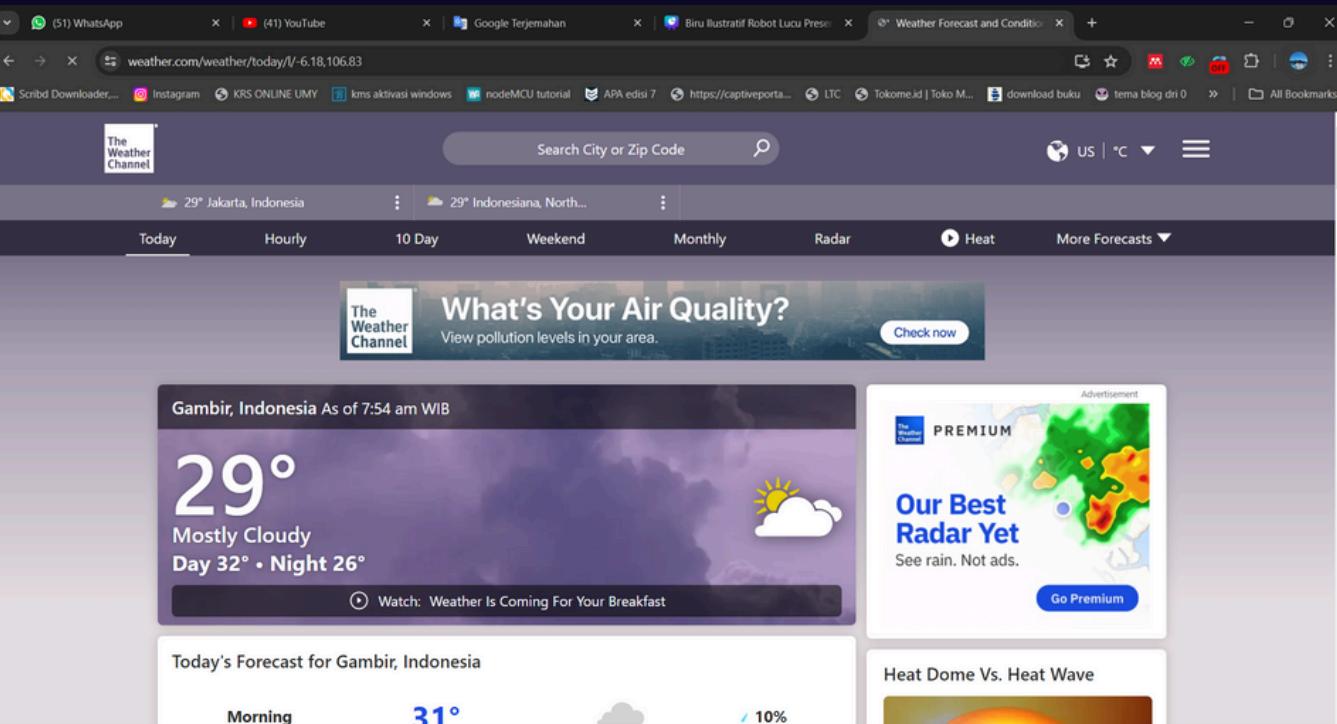
Result (input = Jakarta)



Our Project



Website



Saved data

Today_s_Forecast_for_Gambir_Indonesia_20240612_095952.csv	Weather_Today_in_Gambir_Indonesia_20240612_095952.csv	Hourly_Forecast_20240612_095952.csv	Daily_Forecast_20240612_095952.csv
---	---	-------------------------------------	------------------------------------



Converted to .exe file

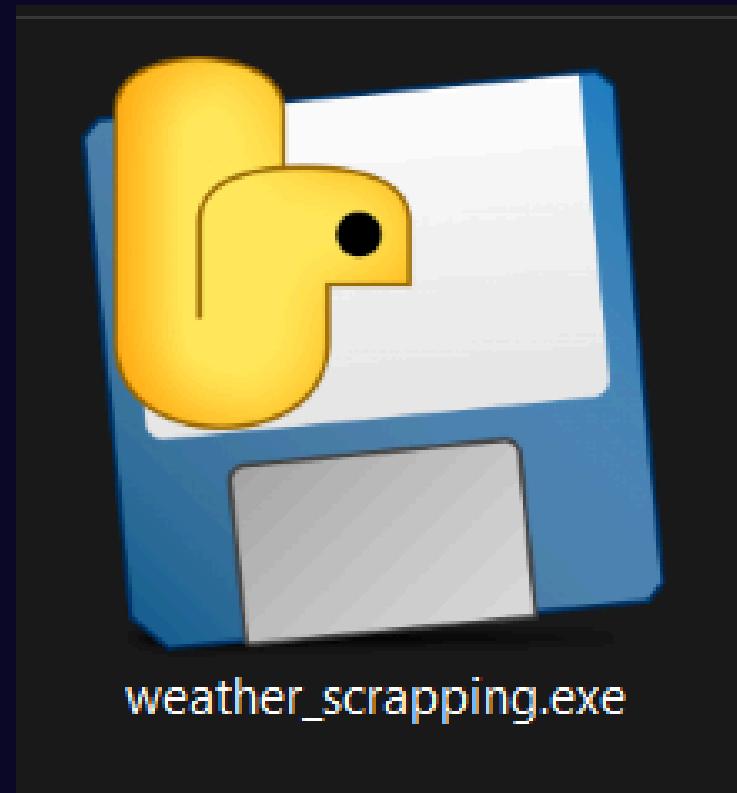


Prompt

```
Anaconda Prompt  
(acpogram) C:\D file\ad comp\final>pyinstaller weather_scraping.py --onefile -w  
665 INFO: PyInstaller: 6.8.0, contrib hooks: 2024.7  
665 INFO: Python: 3.12.3 (conda)  
690 INFO: Platform: Windows-11-10.0.22631-SP0  
690 INFO: Python environment: C:\Users\user\anaconda3\envs\acpogram  
691 INFO: wrote C:\D file\ad comp\final\weather_scraping.spec  
698 INFO: Module search paths (PYTHONPATH):  
['C:\\\\Users\\\\user\\\\anaconda3\\\\envs\\\\acpogram\\\\Scripts\\\\pyinstaller.exe',  
 'C:\\\\Users\\\\user\\\\anaconda3\\\\envs\\\\acpogram\\\\python312.zip',  
 'C:\\\\Users\\\\user\\\\anaconda3\\\\envs\\\\acpogram\\\\DLLs',  
 'C:\\\\Users\\\\user\\\\anaconda3\\\\envs\\\\acpogram\\\\Lib',  
 'C:\\\\Users\\\\user\\\\anaconda3\\\\envs\\\\acpogram',  
 'C:\\\\Users\\\\user\\\\AppData\\\\Roaming\\\\Python\\\\Python312\\\\site-packages',  
 'C:\\\\Users\\\\user\\\\AppData\\\\Roaming\\\\Python\\\\Python312\\\\site-packages\\\\win32',  
 'C:\\\\Users\\\\user\\\\AppData\\\\Roaming\\\\Python\\\\Python312\\\\site-packages\\\\win32\\\\lib',  
 'C:\\\\Users\\\\user\\\\AppData\\\\Roaming\\\\Python\\\\Python312\\\\site-packages\\\\Pythonwin',  
 'C:\\\\Users\\\\user\\\\anaconda3\\\\envs\\\\acpogram\\\\Lib\\\\site-packages',  
 'C:\\\\Users\\\\user\\\\anaconda3\\\\envs\\\\acpogram\\\\Lib\\\\site-packages\\\\win32',  
 'C:\\\\Users\\\\user\\\\anaconda3\\\\envs\\\\acpogram\\\\Lib\\\\site-packages\\\\win32\\\\lib',  
 'C:\\\\Users\\\\user\\\\anaconda3\\\\envs\\\\acpogram\\\\Lib\\\\site-packages\\\\Pythonwin',  
 'C:\\\\D file\\\\ad comp\\\\final']  
1310 INFO: checking Analysis  
1310 INFO: Building Analysis because Analysis-00.toc is non existent  
1311 INFO: Running Analysis Analysis-00.toc  
1311 INFO: Target bytecode optimization level: 0  
1311 INFO: Initializing module dependency graph...  
1311 INFO: Caching module graph hooks...  
1325 INFO: Analyzing base_library.zip ...  
3967 INFO: Loading module hook 'hook-encodings.py' from 'C:\\\\Users\\\\user\\\\anaconda3\\\\envs\\\\acpogram\\\\Lib\\\\site-packages\\\\
```

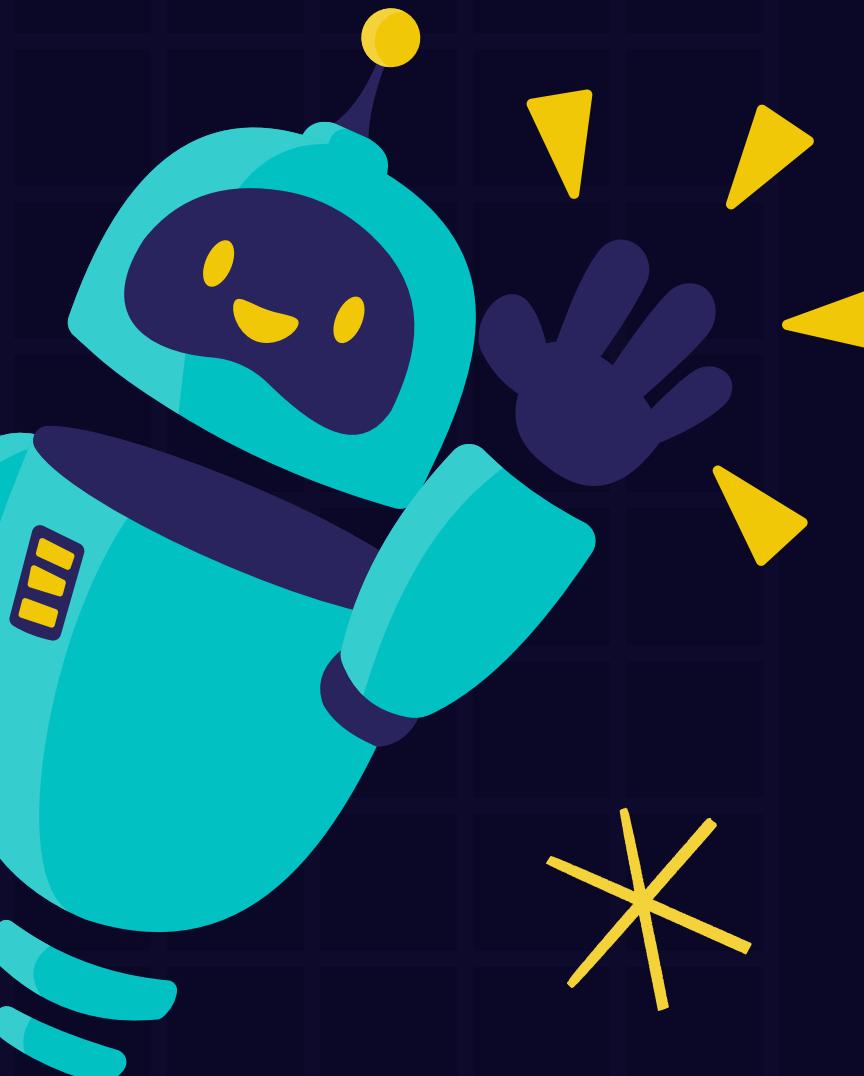


Output





Conclusion



This program effectively combines web scraping, data processing, and a user-friendly interface to deliver comprehensive weather information for any specified location. This functionality is useful for users needing real-time weather updates and forecasts for planning and decision-making.

