亞洲大學
ASIA UNIVERSITY

Final Project Report
Advanced Computer Programming

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

Group Name : College Dropout

Teacher : DINH-TRUNG VU

2024-06

Chapter 1 Introduction

1.1 Github

1) **Group Github Account**: niican
2) **Group Project Repository**: College_Dropouts
3) **List of submitted file**s:
   - **Weather_scrapping**.py
   - **Weather_scrapping**.exe

1.2 Topic

Real-Time Weather Information Extraction and Visualization Tool

1. Brief Description

Our project builds a tool that scrapes real-time weather data from weather.com, including current conditions, hourly, and daily forecasts. It displays this information in an interactive graphical interface and allows users to export the data to CSV files for further use. This tool provides accurate and detailed weather updates for effective planning and decision-making.

1.3 Project Overview

Advanced Features and Libraries Used:

- 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)

Results achieved:

Our program successfully converts user-inputted location names into latitude and longitude coordinates using Geopy, scrapes weather data from weather.com with BeautifulSoup, and converts temperatures from Fahrenheit to Celsius. It organizes this data into Pandas DataFrames and displays it in a user-friendly GUI created with Tkinter and CustomTkinter, complete with real-time map integration using TkinterMapView. Additionally, the program offers functionality to export weather data to CSV files, providing a comprehensive tool for retrieving, displaying, and saving weather information. The other, our project is converted to .exe , so whoever want to use, just directly run and no need to install the python in their device.

Implementation

1.1 get_lat_long(location_name):

1.1.1 Description:

This function gets the latitude and longitude of a given location name.

1.1.2 Fields:

Uses the Nominatim geocoder from the `geopy` library.

1.1.3 Methods:

`geolocator.geocode(location_name)`: Returns a location object with latitude and longitude.

1.1.4 Parameters:

`location_name` (str) - the name of the location to geocode.

1.1.5 Returns:

Tuple (latitude, longitude) or (None, None) if location not found.

1.2 extract_numeric_temperature(temperature_string):

1.2.1 Description:

Extracts numeric temperature value from a string.

1.2.2 Fields:

None.

1.2.3 Methods:

String manipulation and list comprehension to filter digits and period.

1.2.4 Parameters:

`temperature_string` (str) - the temperature string.

1.2.5 Returns:

Numeric temperature as an integer (or None if not found).

1.3 convert_fahrenheit_to_celsius(fahrenheit):

1.3.1 Description:

Converts a temperature from Fahrenheit to Celsius.

1.3.2 Fields:

None.

1.3.3 Methods:

Uses the formula Celsius=(Fahrenheit−32)×59\text{Celsius} = \frac{(Fahrenheit - 32) \times 5}{9}Celsius=9(Fahrenheit−32)×5.

1.3.4 Parameters:

fahrenheit (int/float) - temperature in Fahrenheit.

1.3.5 Returns:

Temperature in Celsius as an integer (or None if input is None).

1.4 get_today_place_forecast(soup):

1.4.1 Description:

Extracts the current day's forecast from the HTML soup.

1.4.2 Fields:

Uses BeautifulSoup to parse HTML content.

today_forecast: the div containing today's weather forecast.

1.4.3 Methods:

soup.find and soup.find_all: Extract specific HTML elements.

pd.DataFrame: Constructs a DataFrame to store the forecast data.

1.4.4 Parameters:

soup (BeautifulSoup object) - parsed HTML content.

1.4.5 Returns:

Tuple (header_element, df_today_forecast) - the header string and a DataFrame of today's forecast.

1.5 todays_weather(soup):

1.5.1 Description:

Extracts today's detailed weather information.

1.5.2 Fields:

Uses BeautifulSoup to parse HTML content.

weather_today: the div containing today's weather details.

1.5.3 Methods:

soup.find and soup.find_all: Extract specific HTML elements.

pd.DataFrame: Constructs a DataFrame to store weather data.

1.5.4 Parameters:

soup (BeautifulSoup object) - parsed HTML content.

1.5.5 Returns:

Tuple (header_element, df_weather_today) - the header string and a DataFrame of weather details.

1.6 hourly_forecast(soup):

1.6.1 Description:

Extracts hourly weather forecast.

1.6.2 Fields:

Uses BeautifulSoup to parse HTML content.

hourly_forecast: the div containing hourly forecast.

1.6.3 Methods:

soup.find and soup.find_all: Extract specific HTML elements.

pd.DataFrame: Constructs a DataFrame to store the hourly forecast data.

1.6.4 Parameters:

soup (BeautifulSoup object) - parsed HTML content.

1.6.5 Returns:

Tuple (header_element, df_hourly_forecast) - the header string and a DataFrame of the hourly forecast.

1.7 daily_forecast(soup):

1.7.1 Description:

Extracts daily weather forecast.

1.7.2 Fields:

Uses BeautifulSoup to parse HTML content.

daily_forecast: the div containing daily forecast.

1.7.3 Methods:

soup.find and soup.find_all: Extract specific HTML elements.

pd.DataFrame: Constructs a DataFrame to store the daily forecast data.

1.7.4 Parameters:

soup (BeautifulSoup object) - parsed HTML content.

1.7.5 Returns:

Tuple (header_element, df_daily_forecast) - the header string and a DataFrame of the daily forecast.

1.8 get_weather(latitude, longitude):

1.8.1 Description:

Fetches weather data for a specific latitude and longitude.

1.8.2 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).

1.8.3 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).

1.8.4 Parameters:

latitude (float) - latitude of the location, longitude (float) - longitude of the location.

1.8.5 Returns:

String with concatenated weather details.

1.9 export_to_csv():

1.9.1 Description:

Exports weather data to CSV files.

1.9.2 Fields:

Uses data_to_save to get data.

1.9.3 Methods:

Iterates over data_to_save to save each DataFrame as a CSV file.

df.to_csv(filename): Saves the DataFrame.

1.9.4 Parameters:

None.

1.9.5 Returns:

None (prints confirmation message for each file saved).

1.10 dropdown_func(event):

1.10.1 Description:

Handles changes in the dropdown selection for forecasts.

1.10.2 Fields:

Uses global variables to update the UI based on selection.

1.10.3 Methods:

combobox.get(): Gets the selected value from the dropdown.

Updates UI labels with forecast data.

1.10.4 Parameters:

event (event object) - the event triggering the function.

1.10.5 Returns:

None.

1.11 show():

1.11.1 Description:

Displays the weather data on the GUI.

1.11.2 Fields:

Uses various global variables to display data.

1.11.3 Methods:

CTkLabel: Creates labels for displaying data.

.grid(): Places the labels in the grid layout.

1.11.4 Parameters:

None.

1.11.5 Returns:

None.

1.12 get_location():

1.12.1 Description:

Retrieves location, fetches weather data, and updates the GUI.

1.12.2 Fields:

Uses global variables for location, latitude, longitude, etc.

1.12.3 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.

1.12.4 Parameters:

None.

1.12.5 Returns:

None.

1.13 maps(latitude, longitude):

1.13.1 Description:

Displays a map for the given latitude and longitude.

1.13.2 Fields:

Uses TkinterMapView to display the map.

1.13.3 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.

1.13.4 Parameters:

latitude (float) - latitude of the location, longitude (float) - longitude of the location.

1.13.5 Returns:

None.

1.14 main():

1.14.1 Description:

Sets up the main GUI layout and event loop.

1.14.2 Fields:

Uses `tkinter` and `customtkinter` to create the GUI.

1.14.3 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().

1.14.4 Parameters:

None.

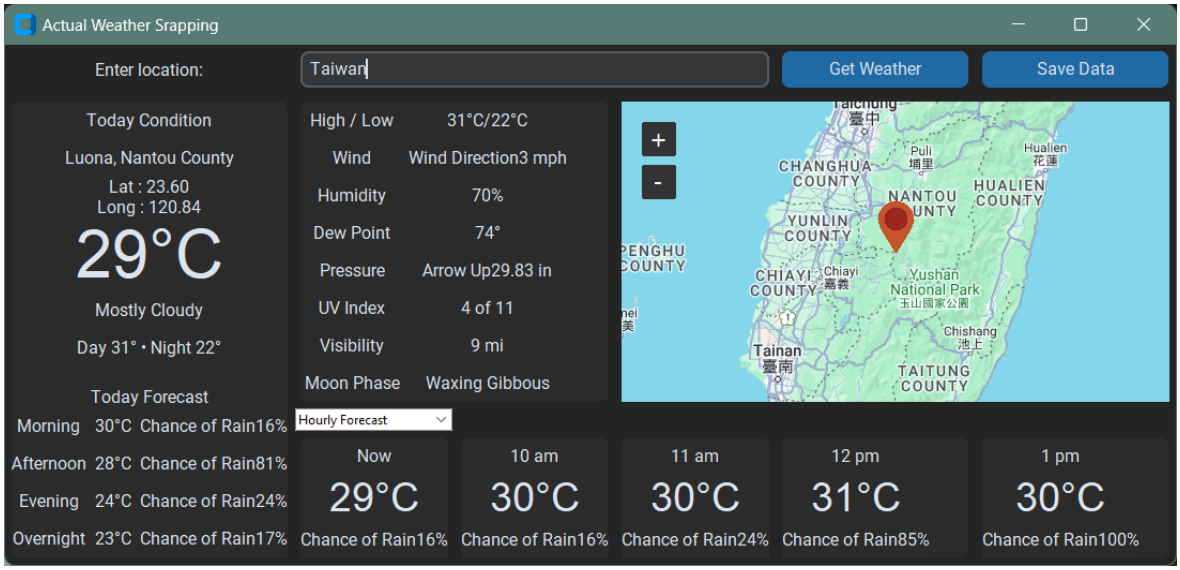
1.14.5 Returns:

None.

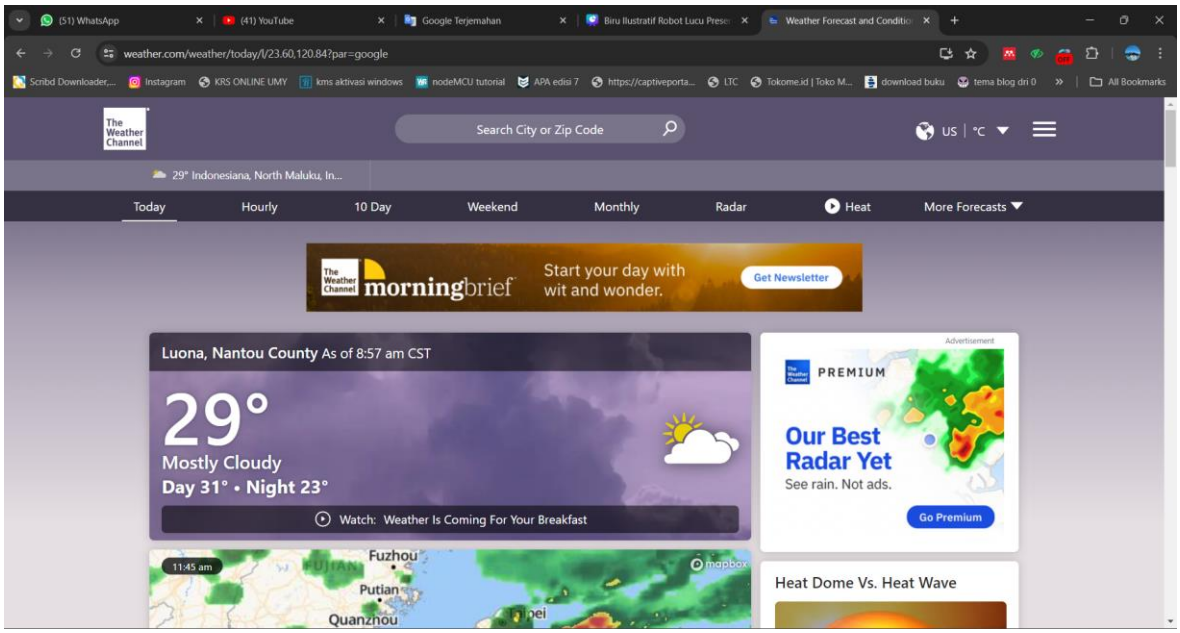
Chapter 2 Results

1.1 Result 1

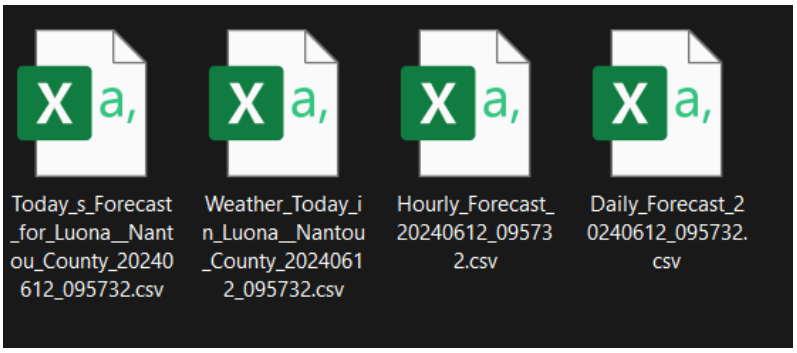
With location “Taiwan”



Real-Time information from the web.

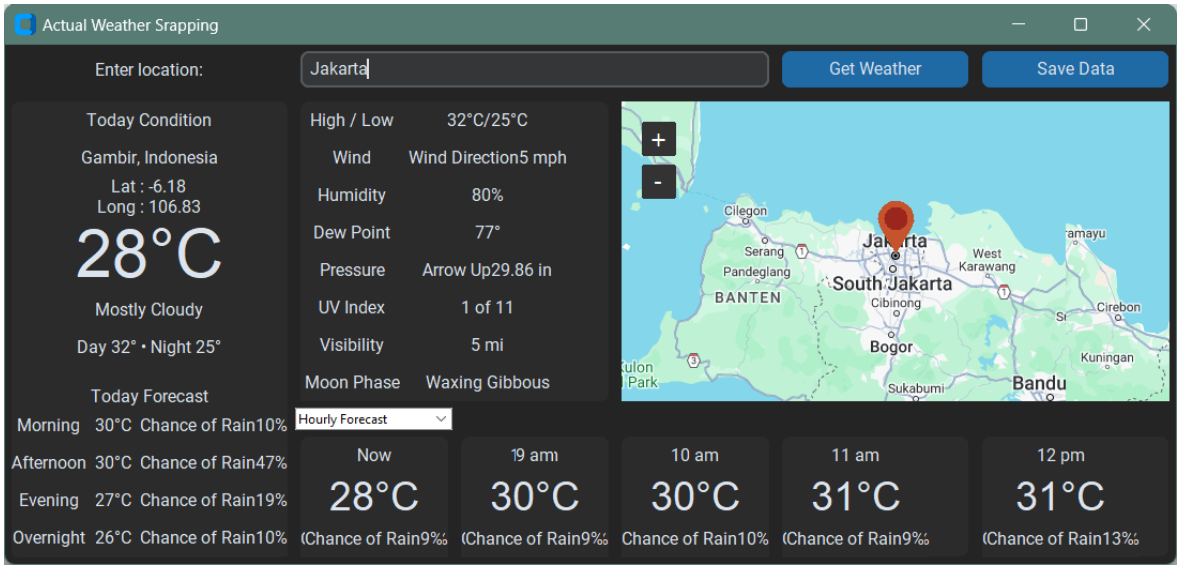


Saved data

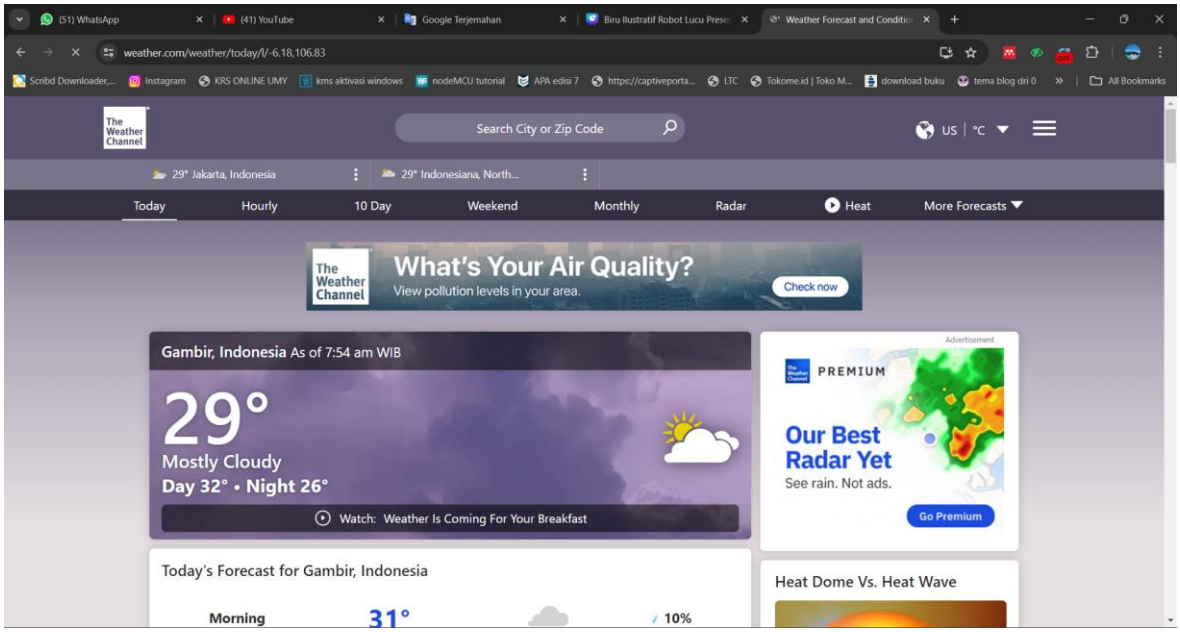


1.2 Result 2

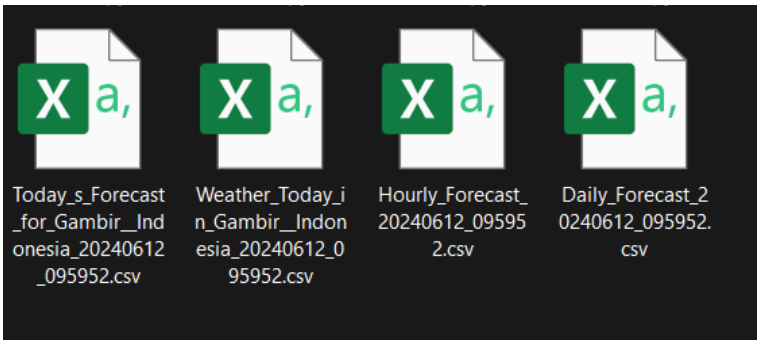
With location “Jakarta”



Real-Time information from the web.

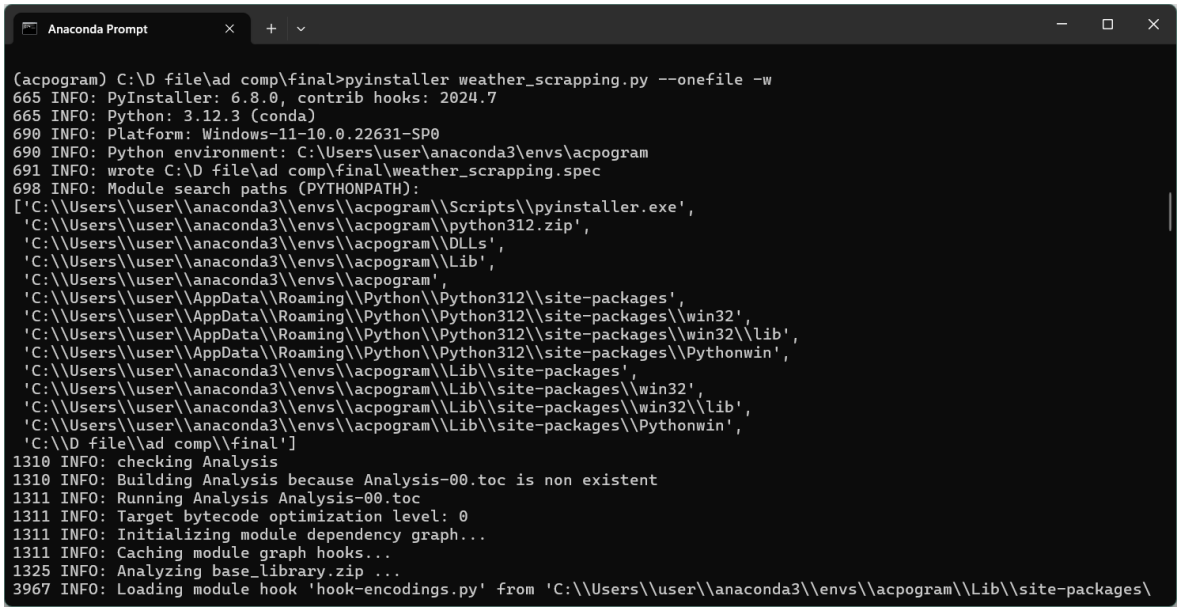


Saved data



1.3 Result 3 (convert to executable file)

Prompt



Output

Conclusions

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.