```
"""
This program is a chatbot that will provide a weather report by scraping data from the OpenWeather API. The user will
be provided with a GUI in which they will have an entry text box where they can type the city they would like a weather
report on. There are three green buttons. The first one will clear the input from the enter city name entry box. The
second button will clear the text in the scrolled text display box. And finally the third and final button will generate
a weather report depending on the city.
"""

# General Libraries
# Specific GUI Window Tabs Libraries
import tkinter as tk
from tkinter import *
from tkinter import ttk
from tkinter.scrolledtext import *
# Using requests as the Python HTML Library
import requests

# Create GUI
# Create Window
# Build Main Window
window = tk.Tk()
# Main Window Title
# AI Weather Assistant
window.title("AI Weather Assistant")
# Window Size
# Wide x Tall
# window.geometry('600x800')
window.geometry("825x800")

# Set style of tabs
style = ttk.Style(window)
# Set location of tabs
# wn = West North
style.configure('lefttab.TNotebook', tabposition='wn')

# tab_control =ttk.Notebook(window)
# Collect and display all tabs starting from left to right.
tab_control = ttk.Notebook(window, style='lefttab.TNotebook')

# Create tabs
# Tab for Wikipedia
tab_weather_search = ttk.Frame(tab_control)
# Add tabs to window
# Name for this tab is AI Weather Report
tab_control.add(tab_weather_search, text='AI Weather Report')
# Create GUI Labels
# Place GUI Labels
# Label on the AI Weather Report will state AI Weather Report
label_weather_report = Label(tab_weather_search, text='AI Weather Report', padx=5, pady=5).grid(column=0, row=0)
# 0,0 is top left of window

# Pack to make visible
tab_control.pack(expand=1, fill='both')


# Function to collect data on city's weather and display it through the scrolled text display grid
# Using the OpenWeather API
def weather_data():
    # The API key that OpenWeather provided me
    open_weather_API_key = 'fb1052215a4d10bcaa759c9da5f5fa20'
    # This variable will store the basic OpenWeather URL
    open_weather_URL = "http://api.openweathermap.org/data/2.5/weather?"
    # A variable that stores and correlates to what the user inputs in the enter a city entry box
    city_name = weather_entry.get()
    # This variable will hold the complete URL address
    complete_open_weather_URL_data = open_weather_URL + "appid=" + open_weather_API_key + "&q=" + city_name + "&units=imperial"
    # Using the get() method with the request module
    # Return the open_weather_request object
    open_weather_request = requests.get(complete_open_weather_URL_data)
    # Using the JSON method on open_weather_request object
    # Convert the JSON file into Python data
    weather_json = open_weather_request.json()

    # An if statement that...
    if weather_json['cod'] != 'City Not Found':
        # This variable will store the values in the 'main' section from the JSON file
        weather_json_section_1 = weather_json['main']
        # This variable will store the current temperature found about the city with OpenWeather
        current_temperature = weather_json_section_1['temp']
        # This variable will store the minimum temperature found about the city with OpenWeather
        current_minimum_temperature = weather_json_section_1['temp_min']
        # This variable will store the maximum temperature found about the city with OpenWeather
        current_maximum_temperature = weather_json_section_1['temp_max']
        # This variable will store the current humidity found about the city with OpenWeather
        current_humidity = weather_json_section_1['humidity']

        # This variable will store the values in the 'weather' section from the JSON file
        weather_json_section_2 = weather_json['weather']
        # This variable will store the description found about the city with OpenWeather
        weather_description = weather_json_section_2[0]['description']

        # This variable will store the values in the 'wind' section from the JSON file
        weather_json_section_3 = weather_json['wind']
        # This variable will store the wind speed found about the city with OpenWeather
        current_wind_speed = weather_json_section_3['speed']

        # Display the weather report collected from the program into the weather_output_display scrolled text
        # Display the name of the city that is receiving the weather report
        # Display the current temperature of the city
        # Display the minimum temperature of the city
        # Display the maximum temperature of the city
        # Display the humidity of the city
        # Display the current wind speed of the city
        # Display the weather description of the city
        # Convert all the variables into strings.
        weather_output_display.insert(tk.END, 'Weather Report For: ' + str(city_name) + '\n' +
                                      'Temperature: ' + str(current_temperature) + '\n' +
                                      'Minimum Temperature: ' + str(current_minimum_temperature) + '\n' +
                                      'Maximum Temperature: ' + str(current_maximum_temperature) + '\n' +
                                      'Humidity: ' + str(current_humidity) + '\n' +
                                      'Current Wind Speed: ' + str(current_wind_speed) + '\n' +
                                      'Weather Description: ' + str(weather_description))


# Function that deletes user input in the enter a city name entry text box
def weather_erase_input():
    #   Delete any input in the city search box
    weather_entry.delete(0, 'end')
```

```
120
121  # Function that deletes what the program outputs
122  def weather_erase_output():
123      # Delete all text placed in the output scrolled text display box
124      weather_output_display.delete(1.0, END)
125
126
127  # Main Home Tab
128  # Create a label to instruct the user on how to use the AI Weather Report program
129  Label(tab_weather_search, text='1) Enter the name of the city:\n2) Click Generate Weather Report Button', padx=5,
130        pady=5).grid(row=1, column=0)
131
132  # Create a label to instruct the user to input the word or phrase they want the AI to search in through OpenWeather
133  Label(tab_weather_search, text="Enter the city name:").grid(row=2)
134
135  # Entry box for user to input their city weather report subject
136  weather_entry = Entry(tab_weather_search)
137  # Create a grid for the weather_entry box
138  weather_entry.grid(row=2, column=1)
139
140  # Buttons
141  # Button to clear all user input in the entry box
142  # Button with text saying Clear Input to instruct user
143  # Button correlates to weather_erase_input function
144  # Button is green with white text
145  button_weather_search_input = Button(tab_weather_search, text='Clear Input', command=weather_erase_input,
146                                       width=25, bg='green', fg='#fff')
147  # Create a grid for the Clear Input button
148  button_weather_search_input.grid(row=7, column=0, padx=10, pady=10)
149
150  # Button to clear all program output in the output display box
151  # Button with text saying Clear Output to instruct user
152  # Button correlates to weather_erase_output function
153  # Button is green with white text
154  button_weather_search_output = Button(tab_weather_search, text='Clear Output', command=weather_erase_output,
155                                        width=25, bg='green', fg='#fff')
156  # Create a grid for the Clear Output button
157  button_weather_search_output.grid(row=8, column=0, padx=10, pady=10)
158
```

```
# Button to generate the weather report depending on city
 # Button with text saying Generate Weather Report
 # Button correlates to weather_data
# Button is green with white text
 button_weather_data_process = Button(tab_weather_search, text='Generate Weather Report',
                                      command=weather_data, width=25, bg='green', fg='#fff')
 # Create a grid for the Generate Weather Report button
 button_weather_data_process.grid(row=9, column=0, padx=10, pady=10)

# Create a scrolled text for the output display box
# Using wrap=WORD to make sure words don't cutoff in the output display box
 weather_output_display = ScrolledText(tab_weather_search, wrap=WORD)
 # Create a grid for the display box
 weather_output_display.grid(row=11, column=0, columnspan=3, padx=5, pady=5)

 # Keep window alive
 window.mainloop()
```