newww.py

```
import requests #used to getting or importing data
from tkinter import * #tkinter library
from tkinter import ttk #used fr using cursur arrow in ur output
import json #usng json bcz the data we get from api is in another from it convert it in json
import datetime # using it to import current time and date
import csv # using it to import cvs file in the code
from tkinter import Canvas, Scrollbar, Label, Frame #USED TO ADD A SCROLL BAR USING CANVAS

#creating linked list
class Node:
def __init__(self, data):
self.data = data
self.next = None

class LinkedList:
def __init__(self):
self.head = None

def append(self, data):
new_node = Node(data)
if not self.head:
self.head = new_node
return
last_node = self.head
while last_node.next:
last_node = last_node.next
last_node.next = new_node

def get_list(self):
city_list = []
current_node = self.head
while current_node:
city_list.append(current_node.data)
current_node = current_node.next
return city_list

# function for sorting
def insertion_sort(arr):
for i in range(1, len(arr)):
key = arr[i]
j = i - 1
while j >= 0 and arr[j] > key:
arr[j + 1] = arr[j]
j -= 1
arr[j + 1] = key

# function to fetch weather data from api
def fetch_weather_data(city):
api_key = "8b50c3073268c55d616807fcde55db2a"
data = requests.get(f"https://api.openweathermap.org/data/2.5/weather?q={city}&appid=
{api_key}").json()
return data
def linear_search(city_list, target):
for city in city_list:
if target.lower() in city.lower():
```

```
return True
return False

def update_combobox(event):
typed = city_name.get()
if typed == "":
matching = list_name
else:
matching = [city for city in list_name if linear_search([city], typed)]
com['values'] = matching

#func for combobox
##def update_combobox(event):
# typed = city_name.get()
# if typed == "":
# matching = list_name
#else:
# matching = [city for city in list_name if typed.lower() in city.lower()]
#com['values'] = matching


def data_get():
city = city_name.get()
data = fetch_weather_data(city)
if 'weather' in data:
w_label1.config(text=data["weather"][0]["main"])
wb_label1.config(text=data["weather"][0]["description"])
else:
w_label1.config(text="N/A")
wb_label1.config(text="N/A")
if 'main' in data:
temp_label1.config(text=str(int(data["main"]["temp"] - 273.15)))
per_label1.config(text=data["main"]["pressure"])
else:
temp_label1.config(text="N/A")
per_label1.config(text="N/A")
search_data.append(city)
insertion_sort(search_data)
searched_cities = "\n".join(search_data)
searched_cities_label.config(text=searched_cities)

# switch page function
def switch_page():
for widget in win.winfo_children():
widget.pack_forget()
#front top HEADING
second_page_label = Label(win, text="MDAH WEATHER APP", font=("Time New Roman", 30, "bold"))
#FRONT HEADING
second_page_label.pack()
#applyng sorting
insertion_sort(search_data)
searched_cities = "\n".join(search_data)
searched_cities_label.config(text=searched_cities)
searched_cities_label.place(x=500, y=180, height=200, width=360)
done_button.place(x=200, y=190, height=50, width=100)

def switch_back():
for widget in win.winfo_children():
widget.pack_forget()
```

```
name_label.place(x=480, y=120, height=50, width=400)
com.place(x=25, y=120, height=50, width=450)
w_label.place(x=25, y=260, height=50, width=210)
w_label1.place(x=250, y=260, height=50, width=210)
wb_label.place(x=25, y=330, height=50, width=210)
wb_label1.place(x=250, y=330, height=50, width=210)
temp_label.place(x=25, y=400, height=50, width=210)
temp_label1.place(x=250, y=400, height=50, width=210)
per_label.place(x=25, y=470, height=50, width=210)
per_label1.place(x=250, y=470, height=50, width=210)
done_button.place(x=200, y=190, height=50, width=100)
arrow_button.place(x=212, y=530, height=50, width=50)
searched_cities_label.place(x=500, y=180, height=200, width=360)
arrow_button.place(x=212, y=1, height=50, width=50)
done_button.place(x=200, y=190, height=50, width=100)
########################################
def linear_search(city):
file_path = 'C:/Users/DANISH LAPTOP/Desktop/city_coordinates.csv'
with open(file_path, newline='') as csvfile:
reader = csv.DictReader(csvfile)
for row in reader:
if row.get('Address', '').lower() == city.lower():
return row.get('Latitude'), row.get('Longitude')
return None, None
def store_coordinates(city):
latitude, longitude = linear_search(city)
if latitude is not None and longitude is not None:
file_name = 'C:/Users/DANISH LAPTOP/Desktop/New Text Document.txt'
with open(file_name, 'a') as coordinates_file:
coordinates_file.write(f"{city}: Latitude - {latitude}, Longitude - {longitude}\n")
return f"{city} found. Coordinates stored in coordinates.txt."
return f"{city} not found in the database."

def save_coordinates_to_file(city, latitude, longitude):
file_path = 'C:/Users/DANISH LAPTOP/Desktop/Coordinates.txt' # Replace with your desired file
path
with open(file_path, 'a') as file:
file.write(f"City: {city}\n")
file.write(f"Latitude: {latitude}\n")
file.write(f"Longitude: {longitude}\n\n")


##########filing#######################to store searched data
def save_to_text(city, weather, weather_desc, temperature, pressure,longitude,latitude):
file_name = "C:/Users/DANISH LAPTOP/Desktop/New Text Document.txt"
current_time = datetime.datetime.now()
formatted_time = current_time.strftime("%Y-%m-%d %H:%M:%S")
longitude, latitude = get_coordinates(city)

with open(file_name, 'a') as file:
file.write(f"City: {city}\n")
file.write(f"Weather: {weather}\n")
file.write(f"Weather Description: {weather_desc}\n")
file.write(f"Temperature: {temperature}°C\n")
file.write(f"Pressure: {pressure}\n")
file.write(f"longitude:{longitude}\n")
file.write(f"latitude:{latitude}\n")
file.write(f"Date and Time: {formatted_time}\n\n")
```

```
def data_get():
city = city_name.get()
data = fetch_weather_data(city)
longitude, latitude = get_coordinates(city)
# Extracting weather details
weather = data.get("weather", [{"main": "N/A"}])[0]["main"]
weather_desc = data.get("weather", [{"description": "N/A"}])[0]["description"]
temperature = str(int(data.get("main", {"temp": "N/A"})["temp"] - 273.15))
pressure = str(data.get("main", {"pressure": "N/A"})["pressure"])


if 'weather' in data:
w_label1.config(text=weather)
wb_label1.config(text=weather_desc)
else:
w_label1.config(text="N/A")
wb_label1.config(text="N/A")
if 'main' in data:
temp_label1.config(text=temperature)
per_label1.config(text=pressure)
else:
temp_label1.config(text="N/A")
per_label1.config(text="N/A")

# Add city to search_data list and sorting it
search_data.append(city)
search_data.sort()

#idhr sy Save data krhy to text file
save_to_text(city, weather, weather_desc, temperature, pressure,longitude,latitude)

# Update searched cities label
searched_cities = "\n".join(search_data) # Adjusting spaces and new lines
searched_cities_label.config(text=searched_cities)
#displaying data which we stored in a ffile
def display_file_data():
file_name = "C:/Users/DANISH LAPTOP/Desktop/New Text Document.txt" # Update the file path with
my username

try:
with open(file_name, 'r') as file:
content = file.read()
display_label.config(text=content)
except FileNotFoundError:
display_label.config(text="File not found or empty")
#CREATING A FUNCTION FOR CSV FILE TO ACCESSED OUR DATA SET

def get_coordinates(city):
file_path = 'C:/Users/DANISH LAPTOP/Desktop/city_coordinates.csv' # Update with your file path
with open(file_path, newline='') as csvfile:
reader = csv.DictReader(csvfile)
for row in reader:
if row.get('Address', '').lower() == city.lower():
return row.get('Latitude'), row.get('Longitude')
return None, None

def update_coordinates():
city = city_name.get()
```

```
latitude, longitude = get_coordinates(city)
if latitude is not None and longitude is not None:
coordinates_label.config(text=f"Latitude: {latitude}, Longitude: {longitude}")
else:
coordinates_label.config(text="Coordinates not found")


win = Tk()
win.title("mdah proj dsa")
win.config(bg="turquoise")
win.geometry("1300x700")

name_label = Label(win, text="Searched Cities Are : ⬇️", font=("Time New Roman", 15, "bold"))
name_label.place(x=500, y=120, height=50, width=360)
city_list = LinkedList()
city_names = [
"Quetta","Peshawar","Balochistan", "Multan",
"Islamabad","Karachi","Hyderabad","Multan","Mure","Larkana","Lahore","Sialkot","Sukhur","Sindh",
"Arunachal Pradesh", "Assam", "Andaman and Nicobar Islands","Balochistan", "Bangkok",
"Beijing", "Bihar",
"Chandigarh", "Chhattisgarh", "Dadra and Nagar Haveli", "Daman and Diu", "Delhi", "Goa",
"Gujarat", "Haryana", "Jakarta", "Jammu and Kashmir", "Jharkhand", "Karnataka", "Karachi",
"Kerala", "Kuala Lumpur", "Lakshadweep", "Madhya Pradesh", "Maharashtra", "Manipur",
"Meghalaya",
"Mumbai", "Mizoram", "Nagaland", "National Capital Territory of Delhi", "Odisha", "Puducherry",
"Punjab", "quetta", "Rajasthan", "Seoul", "Shanghai", "Sikkim", "Singapore", "Tamil Nadu",
"Telangana", "Tokyo", "Tripura", "Uttar Pradesh", "Uttarakhand", "West Bengal", ]

for city in city_names:
city_list.append(city)

list_name = city_list.get_list()
city_name = StringVar()

com = ttk.Combobox(win, text="MDAH WEATHER APP", values=list_name, font=("Time New Roman", 20,
"bold"), textvariable=city_name)
com.place(x=25, y=120, height=50, width=450)

w_label = Label(win, text=" WEATHER CLIMATE", font=("Time New Roman", 14,"bold"))
w_label.place(x=25, y=260, height=50, width=210)

w_label1 = Label(win, text=" ", font=("Time New Roman", 14,))
w_label1.place(x=250, y=260, height=50, width=210)

wb_label = Label(win, text=" WEATHER DESCRIPTION", font=("Time New Roman", 13,"bold"))
wb_label.place(x=25, y=330, height=50, width=210)

wb_label1 = Label(win, text="", font=("Time New Roman", 13,))
wb_label1.place(x=250, y=330, height=50, width=210)

temp_label = Label(win, text=" TEMPERATURE°C", font=("Time New Roman", 17,"bold"))
temp_label.place(x=25, y=400, height=50, width=210)

temp_label1 = Label(win, text=" ", font=("Time New Roman", 20,))
temp_label1.place(x=250, y=400, height=50, width=210)

per_label = Label(win, text="PRESSURE(atm)", font=("Time New Roman", 18,"bold"))
per_label.place(x=25, y=470, height=50, width=210)
```

```
per_label1 = Label(win, text=" ", font=("Time New Roman", 20,))
per_label1.place(x=250, y=470, height=50, width=210)

done_button = Button(win, text="Done", font=("Time New Roman", 20, "bold"), command=data_get)
done_button.place(x=200, y=190, height=50, width=100)

arrow_button = Button(win, text="➡️", font=("Arial", 20), command=switch_page,)
arrow_button.place(x=212, y=1, height=50, width=50)

access_button = Button(win, text="Access Stored Data:⬇️ ", font=("Time New Roman", 12,"bold"),
command=display_file_data )
access_button.place(x=930, y=25, height=50, width=200)

# CREATING DISPLAYING LABEL WITHH SCROLL BAR TO DISPLAY TEXTFILE DATA USING CANVAS >>
canvas = Canvas(win, width=330, height=520)
canvas.place(x=890, y=120)

# Create a scrollbar
scrollbar = Scrollbar(win, command=canvas.yview)
scrollbar.place(x=1220, y=120, height=520)

# Create a frame inside the canvas
frame = Frame(canvas)
canvas.create_window((0, 0), window=frame, anchor='nw')

# Add the label to the frame
display_label = Label(frame, text="", font=("Time New Roman", 10), justify=LEFT, anchor="w",
padx=10, pady=10, wraplength=300)
display_label.pack()

# Configure the canvas scrolling
canvas.configure(yscrollcommand=scrollbar.set)

# Update scroll region when the frame changes
def on_frame_configure(event):
canvas.configure(scrollregion=canvas.bbox("all"))

frame.bind("", on_frame_configure)

# Function to update label text
def update_label_text():

label_text = "" * 20
display_label.config(text=label_text)
update_label_text()

#CREATING GRAPH
def display_graph():
# Prepare data for the graph
temperatures = []
pressures = []
cities = []

for city in search_data:
data = fetch_weather_data(city)
if 'main' in data:
temperatures.append(data["main"]["temp"] - 273.15) # Convert temperature from Kelvin to Celsius
pressures.append(data["main"]["pressure"])
cities.append(city)
```

```
# Create a canvas for the graph
canvas = Canvas(graph_label1, width=360, height=240, bg='white')
canvas.pack()
y_values = [-50,-45,-40,-35,-30,-25,-20,-15,-10,-5,-0,+5,+10,+15,+20,+25,+30,+35,+40,+45,+50]

for i in range(len(temperatures)):
x = (pressures[i] - 600) * 36 / 100
y = (temperatures[i] + 1) * 4
canvas.create_oval(x, y, x + 5, y + 5, fill='red')
canvas.create_text(x + 7, y + 7, text=cities[i], font=("Arial", 9))

# Add labels and axes
canvas.create_text(180, 10, text="Temperature/Pressure relation graph for Searched Cities",
font=("Arial", 10))
canvas.create_text(280, 230, text="Pressure (atm)", font=("Arial", 8))
canvas.create_text(25, 120, text="Temperature (°C)", font=("Arial", 8), angle=90)

# Draw the y-axis values with a reduced gap
# for i in y_values:
# y_val = (i + 50) * 4.8
# canvas.create_text(15, 240 - y_val, text=str(i), font=("Arial", 6), anchor='e')

# these r x-axis values
for i in range(600, 1300, 100): # Loop fr x-axis range
x_val = (i - 600) * 36 / 100
canvas.create_text(x_val, 230, text=str(i), font=("Arial", 6))

# graph button
graph_button = Button(win, text="Display Graph➡️", font=("Time New Roman", 12,"bold"),
command=display_graph )
graph_button.place(x=330, y=565, height=50, width=150)

com.bind("", update_combobox)
#displaying searched cities
search_data = []
searched_cities_label = Label(win, text="", font=("Time New Roman", 14,))
searched_cities_label.place(x=500, y=180, height=200, width=360)

#displaying graph label
graph_label1 = Label(win, text="", font=("Time New Roman", 14,))
graph_label1.place(x=500, y=400, height=240, width=360)
#####
coordinates_label = Label(win, text="", font=("Time New Roman", 10,"bold"))
coordinates_label.place(x=25, y=555, height=90, width=280)

any_button = Button(win, text=" Get Coordinates⬇️ ", font=("Time New Roman", 12,"bold"),
command=update_coordinates)
any_button.place(x=75, y=522, height=30, width=160)

win.mainloop()
```