CIS 41B - Lab 4: Multi-threading, multiprocessing, System, Web API, review GUI

Write a weather look up application that lets a student check the temperature and weather condition of California cities that are homes to some 4-year universities.

Part 1: Web access

The current weather data is fetched from an API of OpenWeatherMap.org: <https://openweathermap.org/current>

You will use the API to get weather data "By city name", as the header shows on the webpage. Follow the example for the API call with city and state name.

api.openweathermap.org/data/2.5/weather?q=city\_name,state\_code,country\_code&appid=API\_key

where city\_name is one of 12 city names of your choice, state\_code is "CA", country\_code is "US" and API\_key is your own API key.

To obtain your API key, click on the "Sign In" link of the OpenWeatherMap link above and go to "Create an account".   
You need a log in id, an email address, and a password. You only sign up for the free account, which limits you to 60 data fetches per minute maximum (which should be more than enough to complete this lab).

For the cities used in this lab, you can choose any 12 CA cities that are the locations of your favorite 4-year universities. These could be schools you want to attend or have attened, a family member / friend attends, you've had a good impression with, etc.

As the OpenWeatherMap website states, the data you download from the API is JSON format.

Each data record is fairly large, with nested structures, so it is useful to download it to a JSON file with indentation so it's easier to see the nested structures in order to get data out of them.

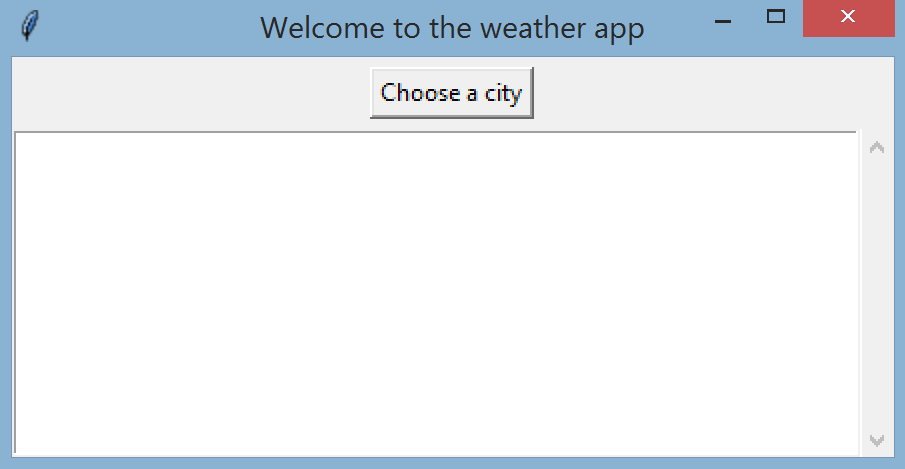
Out of each JSON record you need 2 data values: the 'temp' and the weather condition 'description'  
All quoted strings are fields that you should look for.

Part 2: GUI and System

The application has 2 classes for the 2 GUI windows.

The application starts with a main window that contains a title, a button, and a listbox with a scroll bar.

* The list box can show 12 items, and should have a width large enough to fit the text strings for weather condition that will be displayed.
* The title and button strings are your choice of text.



In the \_\_init\_\_ method of the main window is the code to make the 12 API calls to get the weather data from OpenWeatherMap for the 12 cities. Start by writing a loop to make the 12 API calls, and measure the time it takes to get data for all 12 cities. (It took about 2.5s on my computer, so if you don't see the main window show up right away, it's expected.)

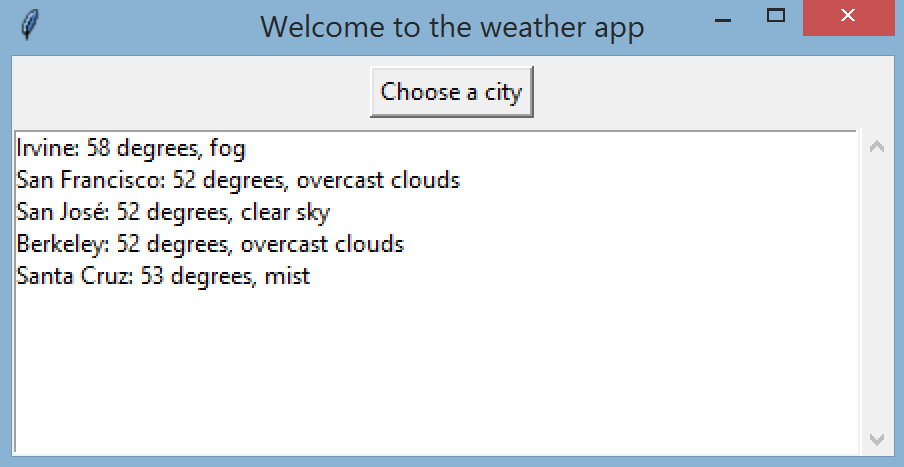
|  |  |
| --- | --- |
| When the user clicks the button to choose a city from the main window, a dialog window shows up with radiobuttons for city choices.   * The cities are in alphabetical order. * Each city choice should have the name of a school that is located in the city. * The first city is selected by default   There is an OK button for the user to commit the choice.  When this dialog window shows up, the main window is deactivated and the user cannot click on the main window to activate it.  If the user clicks 'X' to close the window, then the user is back at the main window. |  |

If the user selects a city and clicks OK:

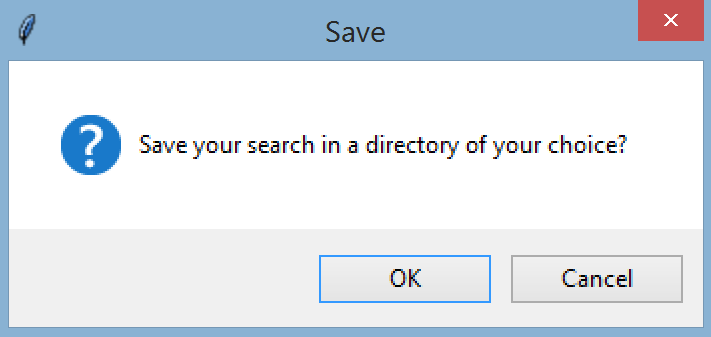
* the dialog window closes
* the main window displays the current temperature and weather condition as one line of text in the main window listbox.

If the user selects more cities, more lines of text are added to the listbox.

Each line of text contains the city name, the current temperature (rounded to the nearest whole number), and a description of the current weather condition.



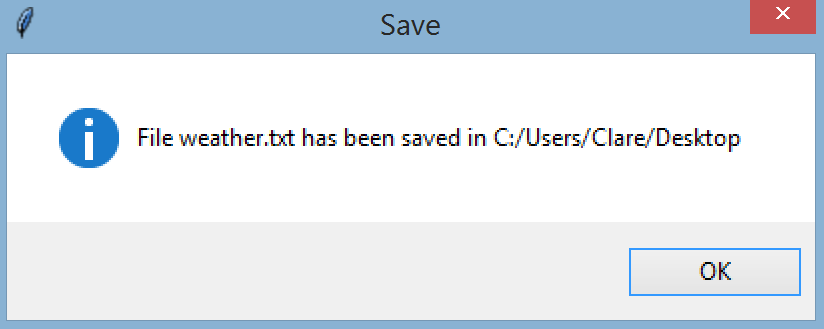
When the user clicks X to close the main window, if the user has made at least one selection that shows up in the listbox, then a messagebox window pops up to ask if the user wants to save the search results in the listbox. The user can click OK to save or click Cancel to not save and quit out of the application.



If the user selects OK, a directory select window shows up and displays the current directory where the python file is running. The user can use the window to get to different directories to choose a directory.

If the user clicks 'X' to close the directory select window, then no file is saved and the application ends.

If user chooses a directory, then file "weather.txt" is created in the chosen directory, and it contains all the lines of text in the listbox. Then a messagebox window appears with a confirmation message, showing the full path of the chosen directory and the new filename.



Part 3: Multithreading

Comment out the loop with the 12 API calls. In its place, add code that creates 12 child threads, one for each API call. Measure the time it takes to do all 12 API calls.

The application should still have the same functionality as before, but the time for the API calls should be noticeably shorter.

Part 4: Multiprocessing

Make a copy of your .py file with threads and use the copy for processes.

Change the 12 threads for API calls into code that creates a multiprocessing pool. Then use the appropriate pool method to run the 12 API calls. Measure the time it takes to do all the API calls.

The application should still have the same functionality as before.

At the end of the multiprocessing source file, in a comment block,

* Rank the 3 ways to make the 12 API calls (in series, multithreading, multiprocessing) in order from slowest to fastest time.
* Explain why they are in that order.

Turn in:

lab4thread.py and lab4process.py