Table of contents:

Project roles:

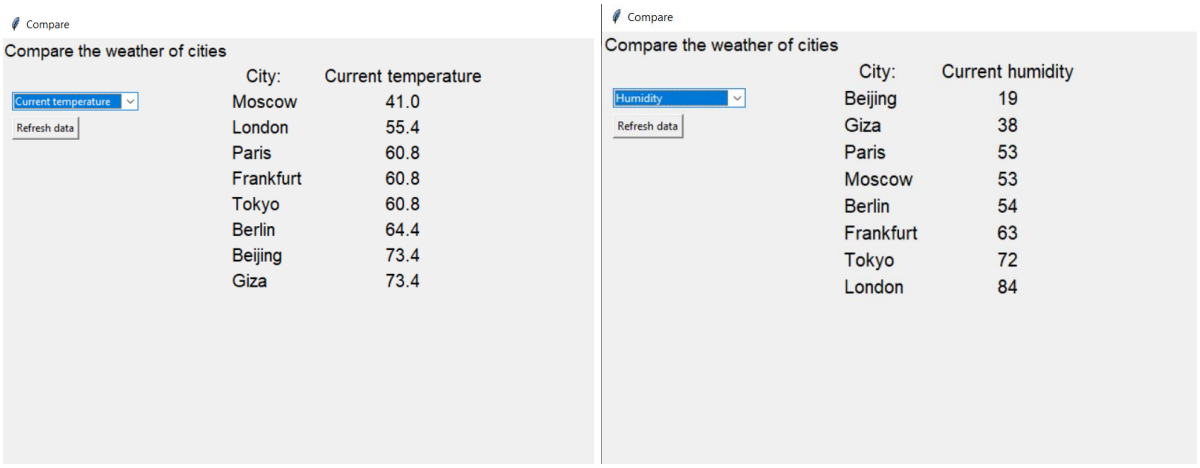
Alex Demange- Programming/ App design/testing

Abstract:

The purpose of this app is to allow people to easily view and compare the current weather in different cities around the world using the free openweathermap api. Most other apps do not allow for people to easily compare different aspects of the current weather around the world. The app has the ability for the user to input and view the weather in real time of up to 8 different cities around the world. It also allows the user to compare the data such as the current temperature or humidity between the chosen cities. Preset options also exist for those who want to see the functions of the comparison tool but do not want to input their own data sets.

Narrative:

The goal of this project was to allow easy comparison between the weather of many different cities at the same time. The ambitions of this project were high at the start, each city would have a small picture next to the stats as well as on the home screen next to its designated button describing the current weather and the exact time at the given location. As development went along, it was clear that not all of these features could be implemented due to time constraints and a lack of experience using python. This did not affect the overall function of the application as it still has all the features considered at the design stage such as the ability to input and display individual data points. The comparison tool was the primary purpose of this project, offering the user the ability to compare not only the temperature between their chosen cities, but data like humidity and pressure all on one screen. This is not a feature often seen on standard weather applications, usually only being able to compare the current temperature of different locations easily.



Design Considerations:

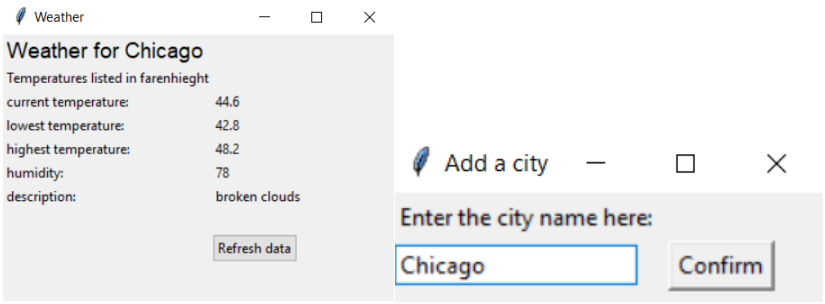
The app should have comparable features to other weather applications that exist on modern phones. While its not intended to act as a replacement to other weather applications it should give users similar data so that they do not need to swap between apps when using the compare function. The data displayed should also be as accurate as possible for the desired

location. Due to the limits of the api, getting weather data for a precise location is not possible, but a feature should be implemented to update the data if the user desires. Implemented algorithms should be efficient and not modify the original data structures when applicable in order to not break other functions of the application. The purpose of this app is simple, and long slow algorithms causing wait time would decrease the functionality of the application. This application should be able to be run in the background without causing slowdowns to the user's system. The UI should be simple in design so that it is easy to use; Buttons and/or text must not clog up the screen and their function should be clear to the user. This app should also be efficient so that it can be run on older and slower computers.

Design and Specification:

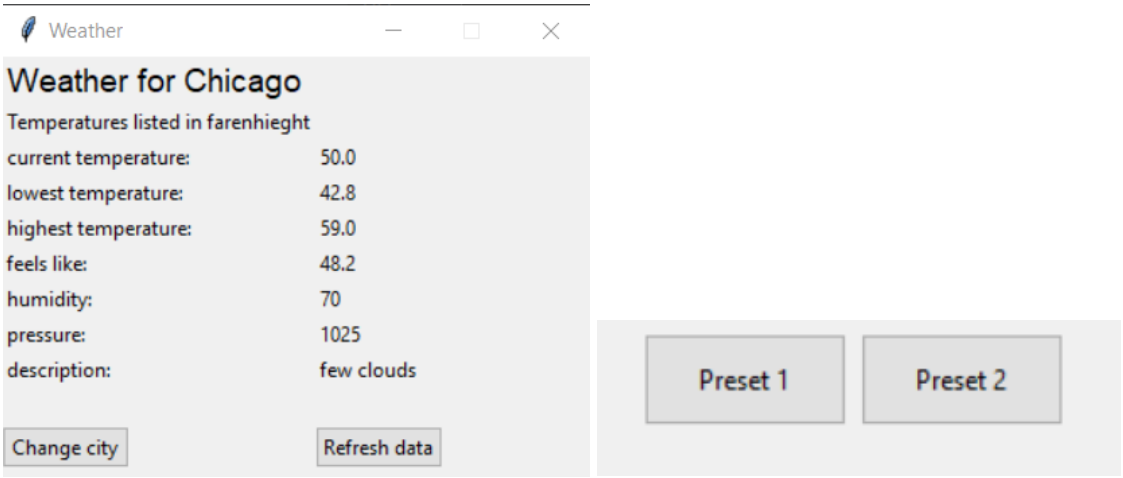
The first steps of designing the project was to get an idea for what tools and functions this app should contain. Research had to be done on what programing language to use and a library to create interfaces for the user to interact with. Python seemed like an obvious choice with considerations given to both java and C++. After python was selected, a library to create interfaces needed to be chosen. Tkinter was selected for this purpose as creating windows was very easy to do and the library had a lot of online support in the form of documentation and tutorials. In terms of app features, the comparison tool was the primary objective for the project, and it was implemented very early on in the project's development just after api calls were set up. The project began with creating a simple UI to make sure all of the functions properly worked. This UI was full of problems such as the buttons being very small and a lot of empty space in the bottom right corner. The biggest issue was that prompting users for input would place the textbox over the buttons, breaking the UI as a whole.

Fixing this poor UI was the next step in the project and was achieved with the use of a frame, allowing for all the text and buttons to be centered on the screen. Implementing this gave the whole screen a much cleaner look and allowed everything to be easily scaled up without worrying about things shifting. The program also needed data to display when the buttons were clicked so getting data from the api was the next logical step. The openweathermap api to use had already been chosen at this point so all that needed to be done was establish a call to the api. This process went smoothly compared to the creation of the main window and a 2-window system for collecting and displaying data was established.



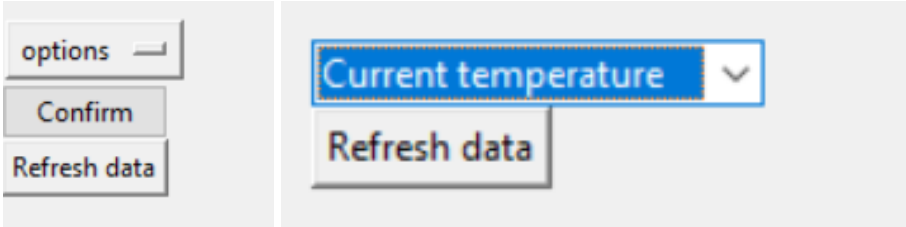
Now it was time to create the comparison tool. This was the most difficult part of the project to develop as it had to dynamically pull the data the user input and sort it based on the selected field. The sorting was the easiest part to implement with a merge sort algorithm being used on a list of the given field and updating the labels with the new order of the data. This did not update the names however, so the names would be displayed in the order of the buttons while the data would be sorted. A function to put the names in the proper order had to be developed next so the sorting algorithm could be properly tested. There were a lot of issues with the original implementation of this algorithm especially involving data sets with duplicate cities. After patching these bugs, the application was in a stable state and attention turned to patching bugs and adding support features.

The data list was expanded to include the current pressure and the feels like temperature of the selected cities. These were being imported with the api call but were not being displayed in order to simplify the data. In order to make the application more similar to other weather apps, these features were added to both the single city display and the comparison tool. Two additional big features added in this stage were the swap button and the preset options. The swap button allows for the user to easily swap one of the selected cities at the push of a button instead of having to relaunch the application and re add all of the cities. The preset button was added to make it easier for the user to use the comparison tool, being able to test between a list of preset cities. The two presets currently provide data for the eastern and western hemispheres of the world. Preset 1 has cities from the U.S. and Canada while preset 2 contains cities across Europe and Asia. With the addition of these features and the removal of bugs such as importing invalid city names, the app was finally in a proper working state.



Restrictions and Constraints:

The biggest limitation of the application is defiantly the UI. Due to my inexperience with tkinter and python as a whole designing a clean UI was a big challenge. Many times I would implement a feature and then find a way to implement it better while researching for the next feature to implement. An example of this is the user input selection feature on the compare screen. At first, it used a dropdown box with all of the different options the user could select. This feature worked fine but required the user to push a button in order to update the screen, making it bad for rapidly scrolling through fields. It also had an extra button on the screen, complicating the UI. A better solution for this was the combobox module which automatically updates the data when a new option is chosen. It also has a dropdown arrow, making it more obvious where a new user should click in order to change what is being compared.



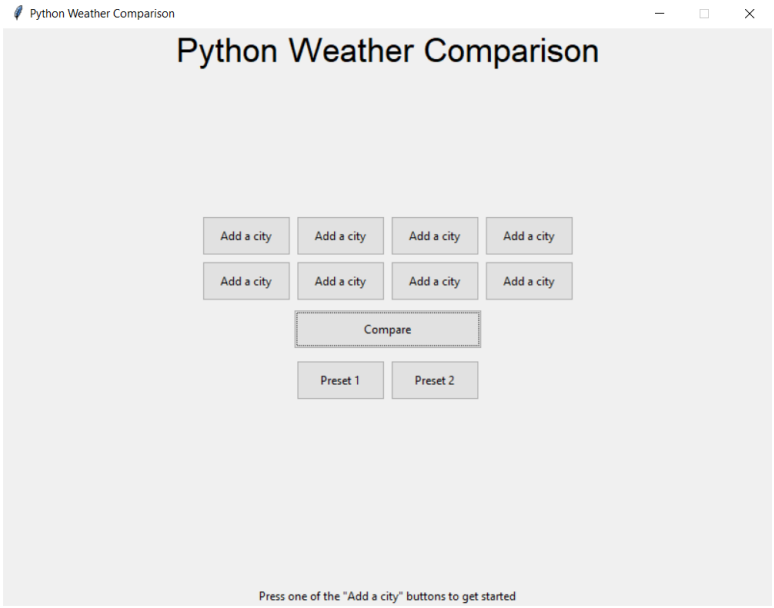
One feature that the application lacks compared to possible competitors is persistant storage of the users inputs and presets. In its current state, the cities input are cleared if the application is closed or the program is stopped. Being able to simply launch the application and view the data would be a big advance in ease of use. The preset options are a way of fixing this, however they have a similar issue in that modifications to the preset will not persist if the application is closed.

Another issue I wish that I had time to add is support for listing temperatures in Celsius. This feature was implemented at one point, and remnants of it can be found in the code, but every time the button was clicked, it would break the sorting algorithms used in the compare screen as some temperatures would be in Celsius and others in Fahrenheit. Storing both temperatures or reverting them all back to Fahrenheit when on the compare screen would be an easy solution to this issue but time constraints pushed this problem back on the list of issues to solve and ultimately lead the feature to be removed.

Testing:

Each new feature was tested on a prototype build before being implemented into the project in order to see how it would look and interact with the existing elements. A lot of issues involving buttons appearing in the wrong location under unique circumstances were caught with this testing process. This also allows for two different versions to be compared side by side to see which version performs better. This technique was essential for testing the layout of the main window. The main window size was tested on a variety of screens with different resolutions. The result of this testing showed that a small window size in the center of the screen looked the best on most displays. The three window sizes considered for the main window were 600 by 600, 800 by 600 and 1000 by 1000. 600 by 600 was ruled out as the window looked too small and oddly shaped. On the other hand, 1000 by 1000 was way to large and drew attention to the relative simplicity of the UI. This left 800 by 600 as the last option for the window size. While this still feels slightly too big, it gives the buttons space to breath and

allows for future space for development without needing to resize the window or existing widgets.



Testing the comparison tool was the hardest part, as cities had to be re added every time a change was made. The compare screen had more bugs than any other part of the application so testing it was very difficult to do. To help alleviate the problem, a preset button was added into the program to automatically fill the list with city names. With this feature, testing the compare function could be done in seconds. It does have the downside of not being able to test with custom data sets but swapping a couple cities is a lot easier than adding 8 new ones. The presets could also be easily modified in the code if custom datasets needed to be extensively tested.

Conclusion:

All in all, I feel that the project was a success, even with its poor UI and relatively few implemented features compared to the original ideas presented. Key features such as the comparison tool work as designed and can be expanded upon in the future. The project was a great way to further my skills with python. Going into this project, I had almost no experience with python, the most experience being basic console programs so taking on this project was a huge step up. The program heavily relied on many basic python features such as loops and if statements and I got a lot of practice using them in more practical settings. The application also used many functions in order to display windows and take user input. Learning how to use python functions is a valuable tool for continuing with python as functions are an essential part of any software application or complex program. As for the future of the application, development may continue adding additional planned features such as support for displaying temperatures in Celsius or adding persistent storage for the cities input by the user as well as redesigning the user interface. Creating a version of this that can be run on a raspberry pi display would be a logical next step, and the small window size would give a jumpstart for adaptation.

Repository Details:

A link to the repository with all of the code can be found at as well as a copy of this report can be found at:

https://github.com/ademange1/Python-Weather-Comparison

To run the program, download the files and run main.py.