**Final Project**

Ricky Cartner

Columbia College

CIS 212 Programming II

Sharon Gumina

April 29, 2021

YouTube Video: https://youtu.be/L81DRu2qx9E

For my final project I decided to create an application with a front-end Graphical User Interface (GUI) for scraping data from a website (WUnderground), storing it in a database (SQLite) and presenting it to the user as needed. Upon completion, this application will be put into use by the Beaufort County Mosquito Control branch of SCDHEC as an addition tool for monitoring weather locations throughout the lowcountry.

My project, titled “WUnderground Weather Information Collection”, was first conceived when I was asked if it was possible to collect weather data from the WUnderground website for a specific set of weather devices. The Beaufort County Mosquito Control has dozens of weather devices planted throughout the lowcountry to track multiple weather conditions, such as the temperatures, rain fall, humidity, and wind speed. These stations transmit their data to the WUnderground website where people are to access from the web to manually review. This was a tedious task to perform for dozens of stations, so I was asked if it could be automated. I was able to put together a rudimentary process that would pull a hard coded range of data (ie. several specific fields of data). But I felt like this was too limited. This project increased the amount of data being returned and gave the user more control.

The application previously mentioned, the application will accept a date and location parameter, fetch the information from the location’s website for the entire month-to-date. It would then store this information in the SQLite database (for faster future retrieval). It will then query the data for the requested date, and sort it into three different types of display. A display of the information for the specific, requested day; the average numbers for the month-to-date of the requested day. For example, if the user selected to see the weather information for April 15th, the average section would display the average weather from April 1st to April 15th. Finally, it displays the information for the entire month.

The process to create this involved (in addition to python) many hours of research in a tool which was new to me called PyQt5. This is what was used to create the GUI as well as used in database retrieval (in some situations). It also involved a working knowledge of SQL and database storage as well as a knowledge of HTML/CSS, as this was required in order to understand how the website was displaying the data on the screen. To code the project, I mainly used PyCharm IDE and I housed it both on my location computer as well as GitHub. I hope to host it on here in order to offer it free to anyone who wishes to use it.

This was an extremely time and knowledge intensive project and took much longer to complete than I originally thought. The more that I worked with and tested, the more bugs that I found. Not just within my code, but in front end usability. I kept putting myself in the user’s position asking, “what if I do this?” or “what if I do that?” or “whose idea was it to make the application do…(blank)?” These questions were the root cause of MANY late nights. But in the end, I think all of my work paid off and I am happy to present a working version of my application (version 1.0).

While building my application, I decided to separated my code into multiple files of a manageable size. This allowed me to compartmentalize my code so that I knew which files were performing which specific function(s). These files are as follows: main.py, model.py, database.py, get\_web\_data.py, views.py, export\_to\_excel.py, get\_list\_of\_websites.py. Of these, the two most used are the views.py and database.py. The view.py being my heavy lifter, performing the majority of the actions on the screen and the database.py interacting with my database. I also separated my database and images into two separate folders inside the root folder.

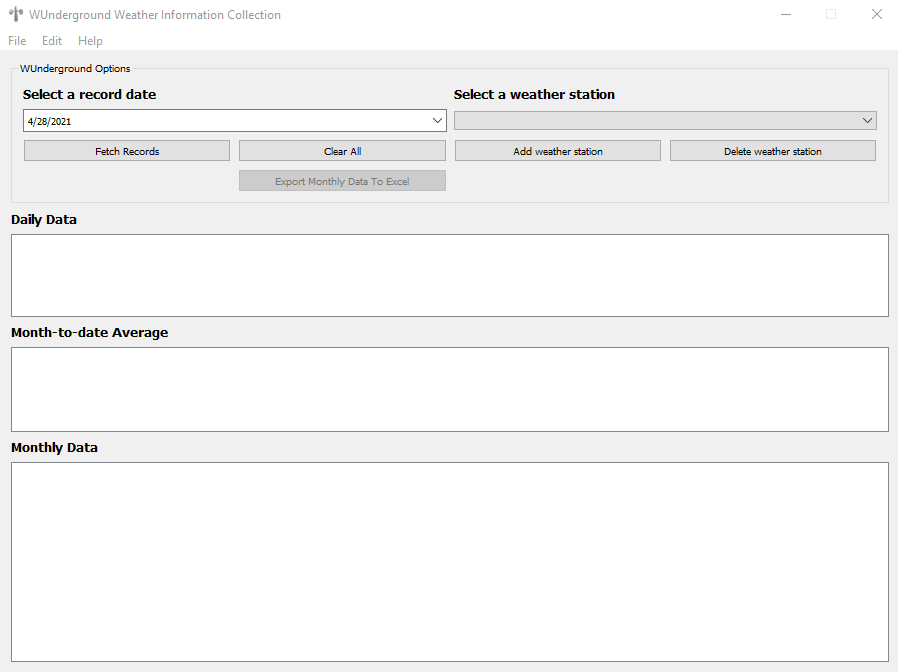
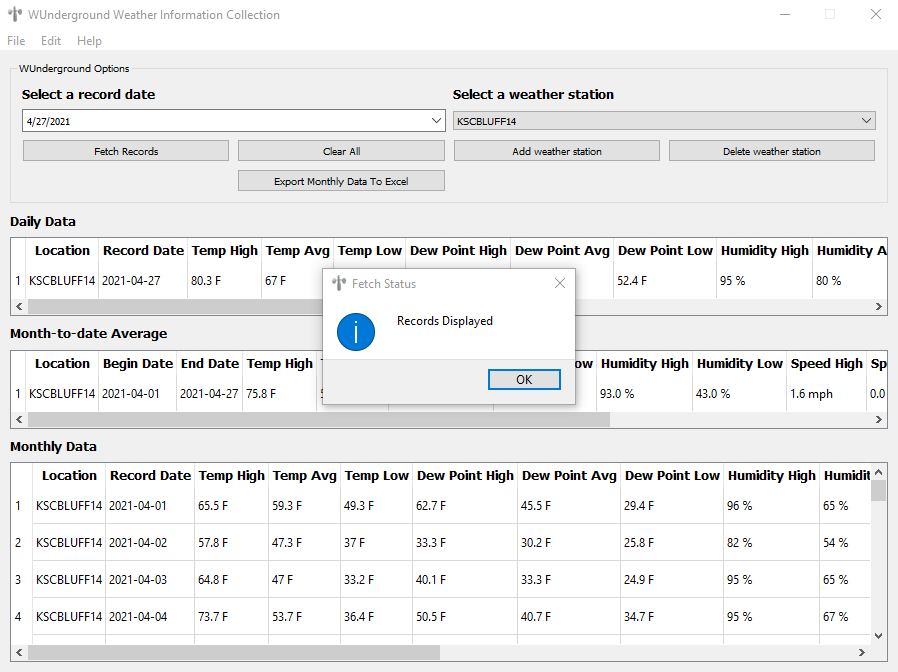
Requirements for running the application are: beautifulsoup4, pyqt5, requests, openpyxl, dateutil, python-dateutil.

The application can be launched in one of three ways: 1) Run the “WeatherInfoCollection.py” file with python; 2) Run the “main.py” file with python; 3) Run the “WeatherInfoCollection.exe” found in the /dist folder (Note: The .exe file in the /dist folder can be moved to any location, provided the database and images folder go in the same location).

The process for using the application is relatively straight forward. The user will launch the application and will be presented with a user interface. From here, they will select a date and location for the data they wish to review. When they submit the request, the application will query the included SQLite database for the information (or pull it from the website if the data does not already exist). It will then display the information to the user and offer them the opportunity to export the monthly information in an XLSX format into users’ temporary folder. After which, the spreadsheet will then open with the users’ default spreadsheet viewer where they can review the file and save to another location, or just close without saving.

Using the temp folder is a process that I began using a long time ago to prevent having to create a specific folder to download files into because they were seldom cleaned up. With the temp folder, Windows has tools built-in that assist in clearing old data, so I didn’t have to worry too much about clutter.

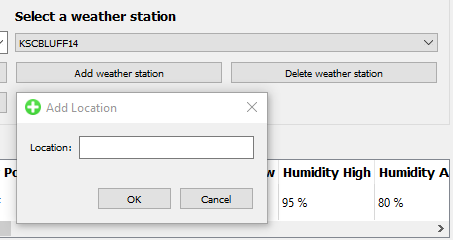
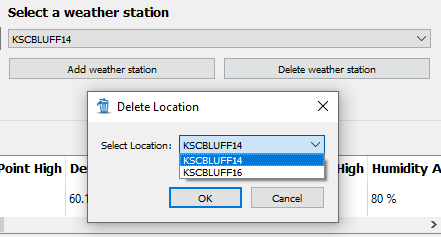
**Initial view** **After records were returned**

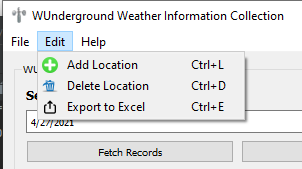
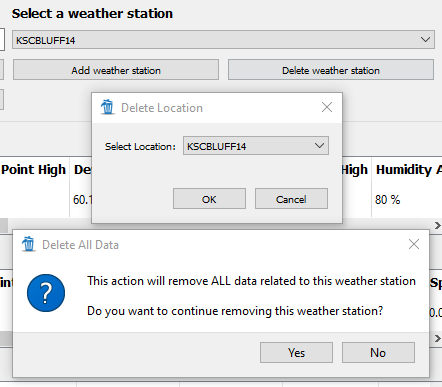
 

\* Note in the screen shots above, that the “Export Monthly Data To Excel” button is disabled if no data is present. This also applies to the export option in the Edit menu.

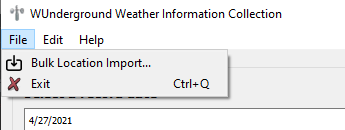
This application also provides the user with the ability to make changes to the weather stations by adding or deleting a location. When a user adds a new location, it is added into the database, and the drop-down box is emptied and reloaded with new data. The process for deleting a record is similar except two delete transactions are performed. The first will delete the data from the table of locations; the second will delete all of the data from the history table (this is the table that contains all of the weather data). Because this can be a serious transaction, the user receives a second message box explaining that it will delete all of the data. The user must confirm that they wish to proceed.

These transactions can be made from two locations: 1) The buttons on the right side of the application or, 2) through the Menu bar (as seen below).

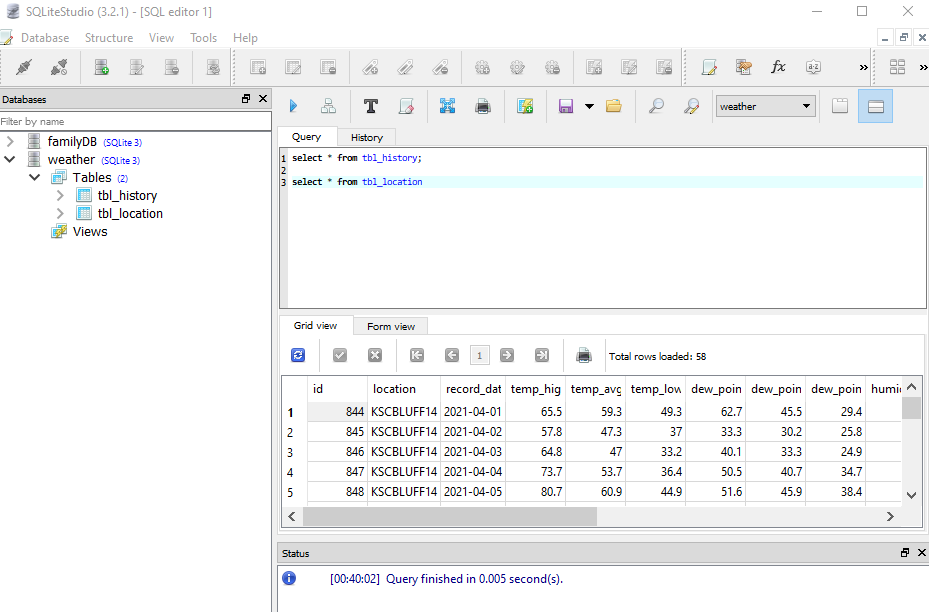
 

When the user opens the application for the first time, there will not be any weather stations to choose from. They will have two options, 1) Manually enter each location via the tools shown above, or 2) I have included a Bulk Import option, which is available from the File menu.



This tool will ask the user to choose a csv file to import. There are only three requirements for using this tool: 1) The file must be a CSV file (note, I have coded the tool to ONLY allow the user to select a CSV file); 2) The file should not have a header (if a header is used, it will be imported as a weather station); 3) The data to import must be in the first column.

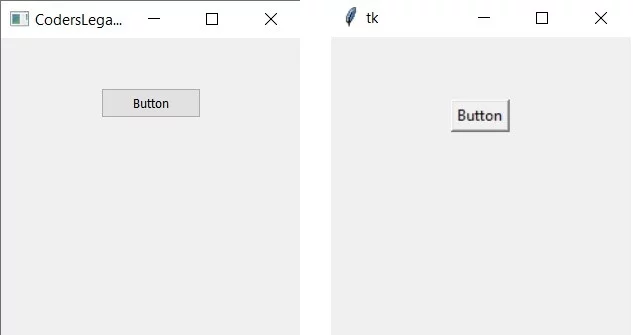
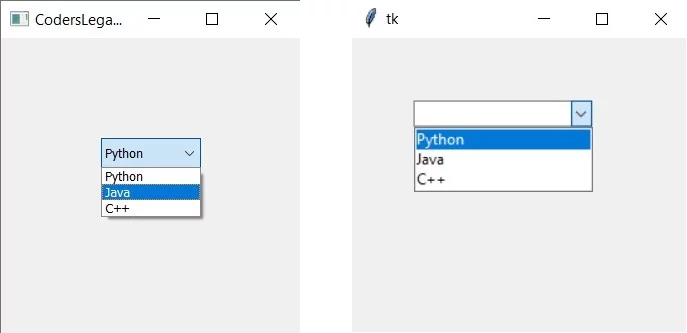
This brings me to my database. My database is a simple one with only two tables: [tbl\_location] and [tbl\_history]. One table stores the weather station location and the other stores all of the weather data. It is difficult to manage a table strictly from code so I used an application called SQLiteStudio. This is a free and open-source (FOSS) that is built for managing SQLite databases. It was invaluable with identifying some issues that were encountered when building the queries. And to also quickly clear out my tables when I needed to perform specific testing.



I have written many python programs in different forms, but nothing like this. I was already familiar with using TKinter for creating GUI’s so I decided to challenge myself to use something different and more robust. So, I decided to take a stab as PyQt5. Right away I noticed an issue with using this and that was with the documentation because is used with Python and C++. Although it is well documented, in my research to perform certain tasks, I constantly ran across code examples for C++. Eventually, I discovered how to translate the C++ code into Python, but I still have not mastered this. For example, in C++, to set the title of a window, the code would be: window::setWindowTitle(), but for Python it is: window.setWindowTitle(). Not much of a difference in this case, but there were other examples in which I was not able to translate to the “Python” way; therefore, I had to take a different approach.

Another issue that I discovered with using PyQt5, is that it is highly geared toward people developing with QT Designer. Although, I would like to learn how to use this WYSIWYG developer tool, it was not something that I wanted to take on. I wanted to be more of a purest and code from scratch. Therefore, I ran into some dead ends as I didn’t have the benefit of using some of the built-in features such as easily creating a loading splash screen.

Although I just mentioned a few of the hassles of using PyQt5, I would like to mention some of the positives. The GUIs that it creates have a more modern look to them than TKinter. Also, Tkinter does not include as many advanced widgets as PyQt5. For example, PyQt5 includes some special widgets such as: QProgressBar, QSpinBox, QDial, and QDateEdit. These are features that are not included with TKinter. TKinter seems to have a slightly blurry look to it than PyQt5. This in itself is a major turn-off for me. It makes an application feel old and tired. Here are some examples of the visual differences:

The bottom line concerning my interest in using PyQt5 was that I wanted to learn something new. I didn’t know about all of the details between the two designers, but now I do and I can use this new knowledge to create a better application in the future.

Another new venture for me was trying out a new IDE. I was using VS Code and had dabbled with other IDE’s such as Sublime but decided to take the plunge and perform the majority of my code with PyCharm. As it turns out, I really enjoyed working with this tool. I found the learning curve much simpler than VS Code. VS Code is written as a universal tool for use in many programming languages. This is pretty unrivaled today; however, I wasn’t using different programming languages, I was only using one. PyCharm was written specifically for python. Most of the time that a tool is written for a specific use, it works better than a universal one because the developers were able to focus on making it work in the best possible way to interact with that language.

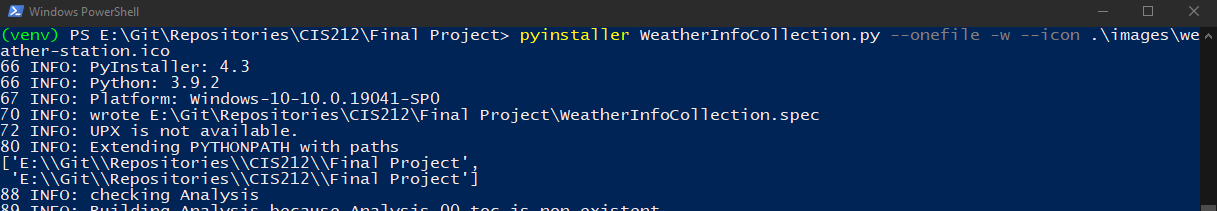
Off the bat, something that irked me with VS Code was that every time I opened the application, I had to change the location that the command line was pointing to. Otherwise, every time that it tried to run my program within the tool, if something interacted with my OS, it would try to use the default user folder. I could have gone into the settings and changed it to look at my development folder by default, but then if I was working on another project, the application would still be looking in the wrong folder. Enter PyCharm. This tool was smart enough to know that if I opened a project in a specific folder, it would use that location as the default location of the command line. This made my work much easier. I was also using a virtual environment to house my packages and python interpreter (python.exe) (I will get more into this later), but VS Code always forced me to activate my virtual environment location when I started the program, whereas PyCharm remembered the last interpreter; allowing me to open a project and just click run to start.

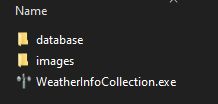
Now onto virtual environments. I found this to be the second most fascinating part of my learning experience and one that I will use for every project. First off, I had to understand what it meant when the IDE asked for the location of my interpreter. Basically, this is the path to the python.exe file that is being used to run the code. The location of this interpreter is also where any site-packages are installed to such as PIP, OpenPyXl, PyQt5 and any other packages that you have to install to run your code. If you use the default interpreter that was created when you installed python, this means that for every project that you work on where you have to PIP Install some package, you will end up filling up your computer with old or unused packages. This is where virtual environments come in.

A virtual environment is a folder location that is created when you run “python3 -m venv [temp-env]”. This will create a folder called “temp-env” which will contain a copy of the python interpreter (python.exe) as well as the standard python library and various supporting files. In order to use this new environment, you must activate it. You do this by typing “temp-env\Scripts\activate.bat” or if you are using powershell command line use “temp-env\Scripts\Activate.ps1” This will allow your project to now use the virtual environment. Now, any packages that you install will be housed here instead of your main interpreter folder (typically located at: C:\Users\[user name]\AppData\Local\Programs\Python\Python[version]).The advantage of this is that whenever you are finished with a project and no longer need any of the files, you simply delete the virtual environment (“temp-env” in my example) and all of the files are gone. No need to bloat your file system. Or you can delete them and reload a new virtual environment and start fresh.

Another thing to note when using a virtual environment is that none of your project files should be stored in this virtual environment folder. You should keep them separate because your project files are permanent but the virtual environment can be temporary.

Finally, in an attempt to make the project easier to run, I used pyinstaller to transform my project into an executable (.exe) file. This took a couple of hours to get working as there are some issues with using images and understanding that I still needed to create a couple of folders for my database and images after the installer was complete. But in the end, I was pleased with the results.





I thank you for your time and enjoyed the instruction and collaboration. I look forward to continuing with my python studies and increasing my knowledge.