CS5010 Project: Charlottesville Weather Analytics

An Interactive Application in Python

Developed By:

Feiyin Wu (fw3fm)
Marcus Rosti (mer3ef)
Nathan Harmon (nth3xd)
Nikhil Mascarenhas (nm4gt)

1. Objective

The objective of this application is to collect weather data for the city of Charlottesville from free sources available on the internet and provide users an interactive application to query the data and find interesting information.

2. Data Source

In this release of our application, we collect weather data from OpenWeatherMap (http://openweathermap.org/). OpenWeatherMap provides weather data using an API (Application Program Interface). The free license for the API provides the full weather data, but limits API calls to 1200 calls per minute.

2.1 API Call for historical Data

The following is the syntax of the API Call used in our application. This call returns hourly weather data between the specified date range for a given City ID.

API call:

http://api.openweathermap.org/data/2.5/history/city?id={id}&type=hour&start={start}&end={end}

Parameters:

id	city ID (id = <> for Charlottesville)
type	= hour (for hourly data)
start	start date in unix time e.g. start=1369728000
end	end date in unix time e.g. end=1369789200

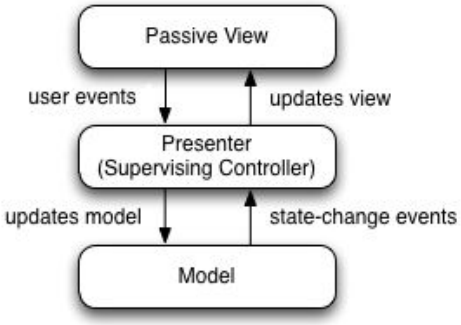
Response:

The API responds with weather data for the specified range and city encoded in JSON format. Each hour in the range is encoded as a separate element in the JSON. Sample JSON for a single hour's data is shown below:


```
{  "main":{"temp":300.32, "pressure":1015, "humidity":74,
           "temp_min":299.25, "temp_max":302.15 },
   "wind":{"speed":2.6, "deg":180 },
   "clouds":{"all":40 },
   "weather":[ { "id":802, "main":"Clouds",
             "description":"scattered clouds", "icon":"03n" }],
   "dt":1438217161 }
```

3. Software Architecture

The application follows the design pattern of Model-View-Controller MVC as much as possible. However in implementation, the software evolved into a series of “Layers of Interaction”. The software consists of four layers from high to low. User interface to interface controller to data controller to the data itself. At each layer of interaction, it only deals with what’s directly below it. That is the interface controller only deals with asking for the CSV of data and a dataframe with that data. The controller only deals with the CSV whereas the user-interface is ignorant of how the data is stored.



This allowed us to overcome the different problems of integrating with a foreign api and presenting useful information to the user. For each phase, we added a specific functionality that abstracts it to the next. Since the first phase dealt entirely with collecting data, the user interface could depend on it being correct.
[1]

4. Development Phases

The development of this application was completed in three phases.

- Phase 1: Write scripts to collect data, script should get data without duplicates and keep the data file up-to-date.
- Phase 2: Create a command line user-interface for users to query data.
- Phase 3: Represent data using graphs

4.1 Phase 1: Data Collection

Algorithm Step	**Implemented in:**
1. If data_file does not exist, create new data_file (CSV format)	*<DIR>/src/controller.py*
2. Find Range for which new data needs to be fetched. (Maximum datetime in file - Current time)	*<DIR>/src/common_lib.py* *Function: update()*
3. Perform API call for above date range, store results in temporary JSON File	*<DIR>/src/common_lib.py* *Function: extractor()*
4. Extract relevant data from JSON File and append output to data_file (CSV format)	*<DIR>/src/common_lib.py* *Function: parse_json_file()*

[1] "Model View Presenter GUI Design Pattern" by Google - http://www.gwtproject.org/articles/testing_methodologies_using_gwt.html. Licensed under CC BY-SA 3.0 via Wikimedia Commons - https://commons.wikimedia.org/wiki/File:Model_View_Presenter_GUI_Design_Pattern.png#/media/File:Model_View_Presenter_GUI_Design_Pattern.png

4.2 Phase 2: User Interface

The Second development phase consisted of the creation of a user interface that provides menu options for users to query the data collected in Phase 1. It abstracts the complications of updating and cleaning the data from the user.

The application is best run from the command line by executing *<DIR>/src/user_interface.py*. Methods for each option reside in a library named: *<DIR>/src/user_interface_lib.py*.

Algorithm

1. Before the user is presented with menu options, the application performs the Data Collection step to ensure that the weather data is up-to-date.
2. The user is presented with options to query the data. The options are discussed in the next section.

3.2.1 Option 1: Today's Weather (*Function: today_weather())*

Today's Weather returns a string containing all the weather information from current date, including minimum and maximum temperatures of the day in Fahrenheit, the average pressure, average humidity, average wind speed, and the average cloud coverage for the day.

3.2.2 Option 2: Average Temperatures (*Function: ave_temps())*

Average Temperature returns two values in Fahrenheit. The first is the average temperature of the previous week. This is accomplished by getting the current date from python's time library, and subtracting off the amount equal to a week. All entries which have greater timestamps are averaged.

The second value is the average temperature of the current month. The current month is determined today's date, and all entries which have the matching month are averaged.

3.2.3 Option 5: Return Data Range (*Function: date_range())*

This function returns the range for which weather data is present. All the times are read in and sorted, with the first and last values returned.

3.2.4 Option 0: Exit

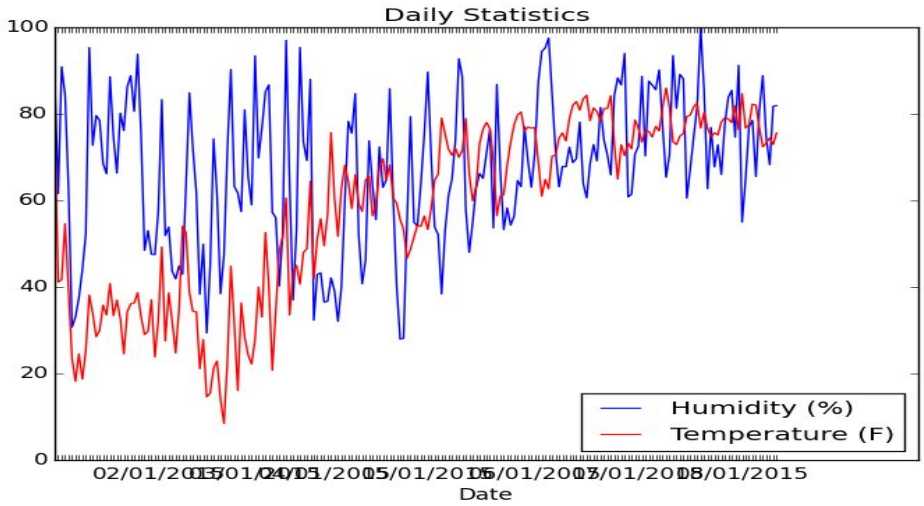
The program prints a message and exits the system.

4.3 Phase 3: Representing Information using Graphs

A pair of graphical options were included in the final phase (Options 3 & 4). Each graph is saved as a png every time their option is called.

4.3.1 Temperature and Humidity Line Graph(*Function: temp_graph())*

The average temperature and humidity is calculated for each day and plotted as a line graph. Temperature and humidity have different units, fahrenheit and percent, respectively, but they cover the same range. The x labels (Dates) are limited to the first of every month, in an effort to make the graph more readable.



4.3.2 Weather Bar Chart(*Function: weather_barchart())*

The weather bar chart shows the frequency of each type of the weather occurred over the past 7 days.