Reference Manual

Vu Q. Le

July 19, 2018

Contents

0.1 Intro

This documentation will serve as a user guide to the Python code and how to extract weather data from Wunderground. For further details, please refer to the code and related comments. There is an API for Wunderground, however, it requires a premium license to allow large number of requests or calls per day. Therefore it is preferred to use Python to scrape the weather data as needed. The process is two step. With a given CSV file that contains a list of pitches, all the necessary HTML files are pulled first. Once all files are saved locally, another script can scrape the necessary weather data from the HTML files and save the results to a new CSV file.

0.2 Useful Stuff to Know

Knowing Python and Python libraries such as datetime, requests, csv, json, and beautifulsoup will be helpful in adding or making changes to the code. Wunderground weather data is not completely consistent across the board. Sometimes, there are missing weather data and/or not all weather data is available for each measurement time. Keeping this in mind, parsing requires multiple conditions to properly parse the HTML files and scrape the weather data. All python scripts are written in Python 3.6.5. To run the scripts, please ensure you have requests and beautifulsoup libraries.

0.2.1 Websites

https://www.crummy.com/software/BeautifulSoup/
http://airnav.com/cgi-bin/airport-search
https://weather.gladstonefamily.net

0.3 Preparation

The pitch CSV file must be sorted first to optimize parsing. It should be sorted by **gameName**, then **at_bat_num**, then **pitch_number**. **sv_id** should be in order after sorting those three fields. This allows the script to parse all similar **gameName** pitches in sequence. The pitch file must be a comma delimited CSV format. If it is not, use Excel to do 'Save As' and select 'CSV (Comma Delimited)' option.

0.4 Pulling HTML Files

Use **get_html.py** script to pull all necessary HTML files one by one automatically. Open the script and specify the filename for the pitch CSV file and the total nunber of rows including the fieldnames. There is a delay between requests to prevent Wunderground from timing out and this value can be changed using **requestDelay**. This script relies on the venue database **ballparks.json** that is exported from **ballparks_closest_airports_vu_new.csv**. There is an optional variable **startRow** to start pulling HTML files from a specific row in the pitch CSV file. This is helpful for resuming the operation at a specific point.

0.4.1 Running the Script

The script can launch by simply opening it or running it within a Python shell. Ensure all parameters are set properly in the script before running. Please see above.

0.4.2 Ballpark JSON File

The **ballparks.json** file contain all ballparks of interest and corresponding three closest weather stations including the latitude/longitude coordinates and distance of each station. The JSON file was originally a CSV file that was converted by using the following website:

https://www.csvjson.com/csv2json

For option '**Output**', please select '**Hash**' instead of '**Array**'.

0.5 Parsing

All scraping is done using **parse.py** script and the library **parse_h**. **parse.py** contains the main algorithm while **parse_h.py** is a library that has supporting functions such as **getAirDensity** and **getClosestMeas**. BeautifulSoup is a library that can parse and serialize HTML data. It is the core of scraping any useful websites such as Wunderground. The function **getClosestMeas** will find the closest time match for a desired measurement such as temp, relative humidity, or pressure.

0.5.1 Running the Script

The script can only parse one station each time. To run the script, open a Python shell starting in the directory that contains **parse.py**. Then run the following commands. For example, we have a pitch CSV file name **sample.csv** and we would like to use the first weather station from **ballparks.json** to extract weather data.

```
from parse import parse
parse(pitchesfilename='sample.csv', stationNum=1)
```

Repeat the second command for the second and third weather stations as defined in **ballparks.json**.

0.5.2 Multiprocessing

Multiprocessing can be implemented using **parse.py**. Please see the sample script **parse_multi.py** for details. One can also run multiple scripts in separate Python shell if desired. The included library **csvtool** will be useful for splitting the pitch CSV file into multiple parts for processing in multiple parts at the same time. Generally, the time can be reduced by 2 to 3 times. After processing, **csvtool** can also join the files back together.

0.5.3 CSV Tools

Within **csvtool** library, there are **csvsplit** and **csvjoin**. **csvsplit** will split a given CSV file by the rowsize. The rowsize argument is the maximum number of rows each partial CSV file will have. The argument **headersall** will put fieldnames on all CSV files if set to True (default). **csvjoin** will join a given list of files and saving it to an output file.

0.6 Other Scripts

0.6.1 get_localtime.py

This script will read the pitch CSV file and use the **sv_id** field to retrieve the local time. This is useful for manually checking the measurement time with Wunderground weather data.

0.6.2 get_airports.py

This script will query **www.airnav.com** for nearby airports given a lat/long coordinate. It will first open **'ballparks_closest_airports_vu_new.csv'** file to retrieve the ballpark of interest, query airnav for nearby airports and save the results in a new file with the suffix '_new'.