Coding Challenge

One of our backend stacks has the following structure: two worker machines that process data as it is appears in an AWS S3-style bucket from an external provider. The purpose of this exercise is to simulate that process to make sure that files are processed properly.

An external provider puts files into our bucket with the following filename format: 2020_08_18_02_15.txt, (date, 2020/08/18, and time, 02:15 AM, based on a 24-hour clock). The provider puts these files into the bucket at roughly 15-minute intervals, though the writing sometimes lags. Within each file is a set of binary JSON objects that are, with some minor modification, turned into tab-delimited files for loading into a relational database.

Unfortunately, loading the data, decoding the JSON objects and writing the files is time consuming and requires two machines ("workers") to keep up with the data being produced. Also, because of the data generating process, the files can be uneven in size, meaning that one file can take a long time to process and the next one may take a very short amount of time. The hard part is keeping straight which files have been processed and which ones have not. Each worker should also be robust to the other worker failing.

The appendix of this document contains a list of files and the approximate time that it takes to process each one. To complete this assignment, please do the following:

1. Using Python 3, write a script that can process and load the data on two different processes (these are the "workers"). In particular, this script should take as its input (at a minimum) where the files will be found. It will wait until it sees files and then processes them, while dealing with the potential that another process may also be running.
2. When a worker begins processing a file it should simulate the processing time for that file based on the information in the file/appendix.
3. You should presume that the files appear in the directory at the time that is associated with their file name. In other words, your algorithm cannot presume that all files are there when you ask for them.
4. The file simulation.py (attached to this assignment) will populate a directory at the correct time with files which contain the runtime, in minutes, within each file. Note that this file has a parameter "secondsInMin" which can be used to speed up the entire process for simulation purposes. It is recommended to put a similar parameter in your code.
5. You can use any standard library.
6. Mentally you should think about these two workers as running forever on two different servers.
7. We would like the individual workers to do some console logging. Specifically, every 5 files that a worker processes we would like the worker to print to the terminal the following information:
    a. Number of files processed.
    b. Total time that worker spent processing (e.g. the sum of the values in the files)
    c. The rolling squared differences of time that worker spent processing. If, for a specific worker, the historical time spent processing files was equal to 5,6,7,8 and the next (5th) file coming in was 9 then this value would be: (5-9)^2 + (6-9)^2 + (7-9)^2 + (8-9)^2 + (9-9)^2= 30. In other words, you take the current processing time and subtract it from the historical processing times, square that number and then sum it.
    d. Given the numbers above, the format of the terminal logging should look something like:

```
Files processed: 5, Total Time: 35, Sum of Squares: 30
```

e. IMPORTANT: Each of these statistics is done within a worker and need not communicate any of the statistical numbers to the other worker.

Your code will be evaluated by having it called from the command line twice (like below)

```
% python3 your_code.py ARGUMENTS &
% python3 your_code.py ARGUMENTS &
% python simulation.py DIRECTORY &
```

Just to be clear: there will be *two* workers/processes running your code at the same time, both of which should be processing files from the same directory. These processes need to process all files which appear in the directory without conflicting with each other or processing the same file twice.

What we are looking for:

- Good clean code and comments.
- If you made assumptions, that is fine, just describe them!

Filename	ProcessingTime
2018_06_28_00_00.txt	22.5050828
2018_06_28_00_15.txt	36.2682872
2018_06_28_00_30.txt	16.2480442
2018_06_28_00_45.txt	18.2287935
2018_06_28_01_00.txt	12.8346711
2018_06_28_01_15.txt	33.6345902
2018_06_28_01_30.txt	22.640068
2018_06_28_01_45.txt	26.1400089
2018_06_28_02_00.txt	17.5672472
2018_06_28_02_15.txt	28.004201
2018_06_28_02_30.txt	19.2419392
2018_06_28_02_45.txt	33.3947852
2018_06_28_03_00.txt	36.979999
2018_06_28_03_15.txt	24.2475979
2018_06_28_03_30.txt	20.0994091
2018_06_28_03_45.txt	20.5797647
2018_06_28_04_00.txt	18.1857038
2018_06_28_04_15.txt	35.2466807
2018_06_28_04_30.txt	36.7162491
2018_06_28_04_45.txt	31.164888
2018_06_28_05_00.txt	29.1469056
2018_06_28_05_15.txt	34.3604648
2018_06_28_05_30.txt	16.3212933
2018_06_28_05_45.txt	32.8551499
2018_06_28_06_00.txt	28.301116
2018_06_28_06_15.txt	36.2694804
2018_06_28_06_30.txt	34.3742056
2018_06_28_06_45.txt	25.143707
2018_06_28_07_00.txt	35.7792421
2018_06_28_07_15.txt	23.7838607
2018_06_28_07_30.txt	30.3121944
2018_06_28_07_45.txt	34.003378
2018_06_28_08_00.txt	28.4348137
2018_06_28_08_15.txt	30.4120305
2018_06_28_08_30.txt	20.290891
2018_06_28_08_45.txt	21.00611
2018_06_28_09_00.txt	35.4181588
2018_06_28_09_15.txt	25.2300617
2018_06_28_09_30.txt	16.2048092
2018_06_28_09_45.txt	27.5437933
2018_06_28_10_00.txt	28.8275469
2018_06_28_10_15.txt	27.8661811
2018_06_28_10_30.txt	26.5863073
2018_06_28_10_45.txt	25.2388229

2018_06_28_11_00.txt	35.0484145
2018_06_28_11_15.txt	32.2205758
2018_06_28_11_30.txt	37.0766723
2018_06_28_11_45.txt	26.079113
2018_06_28_12_00.txt	36.7521344
2018_06_28_12_15.txt	12.8775606
2018_06_28_12_30.txt	16.0389468
2018_06_28_12_45.txt	20.2897979
2018_06_28_13_00.txt	18.1674014
2018_06_28_13_15.txt	28.0353312
2018_06_28_13_30.txt	14.539335
2018_06_28_13_45.txt	31.4626478
2018_06_28_14_00.txt	27.1601525
2018_06_28_14_15.txt	17.1779466
2018_06_28_14_30.txt	29.3624968
2018_06_28_14_45.txt	32.0042649
2018_06_28_15_00.txt	36.86204
2018_06_28_15_15.txt	20.6448372
2018_06_28_15_30.txt	22.9457731
2018_06_28_15_45.txt	22.5354117
2018_06_28_16_00.txt	37.0728314
2018_06_28_16_15.txt	21.7150664
2018_06_28_16_30.txt	20.1183597
2018_06_28_16_45.txt	21.4763721
2018_06_28_17_00.txt	21.4287094
2018_06_28_17_15.txt	29.8167205
2018_06_28_17_30.txt	28.3136726
2018_06_28_17_45.txt	35.687019
2018_06_28_18_00.txt	35.3763017
2018_06_28_18_15.txt	35.9962251
2018_06_28_18_30.txt	30.486624
2018_06_28_18_45.txt	14.6953196
2018_06_28_19_00.txt	37.3213581
2018_06_28_19_15.txt	28.8380489
2018_06_28_19_30.txt	28.3122565
2018_06_28_19_45.txt	29.7442816
2018_06_28_20_00.txt	25.931363
2018_06_28_20_15.txt	35.6121706
2018_06_28_20_30.txt	17.3681563
2018_06_28_20_45.txt	14.6148947
2018_06_28_21_00.txt	25.4629607
2018_06_28_21_15.txt	24.5448418
2018_06_28_21_30.txt	30.9103012
2018_06_28_21_45.txt	32.9414083
2018_06_28_22_00.txt	23.1579187

2018_06_28_22_15.txt	19.7576042
2018_06_28_22_30.txt	19.540461
2018_06_28_22_45.txt	21.525315
2018_06_28_23_00.txt	18.2054742
2018_06_28_23_15.txt	15.6001769
2018_06_28_23_30.txt	15.5330086
2018_06_28_23_45.txt	25.975718