Timezone issue in departure_alarm.py

My specifications (though this should not affect the results in this case):

1. Python version 3.7.1
2. macOS Mojave Version 10.14.1

- The time we specify in human readable format (datetime object) corresponds to the local time based on our local system time and the timezone you are located in (more specifically your system timezone).
- Let us suppose we are looking to download trip information from the MBTA database for trips occurring between 7 AM and 10 AM on May 1, 2018.
- When we convert this to time-after-epoch using the timestamp() function (to supply the **&from_datetime** and **&to_datetime** in the query to the MBTA database), the from_time 7 AM on May 1, 2018 will yield **different results** for epoch-timestamp based on the timezone **you are currently located in**.
- What we really want is the time from epoch for 7 AM on May 1, 2018 corresponding to Boston (US/Eastern) time, since this is the time from epoch that will be stored in the MBTA database.
- So if you don’t do anything, you will be pulling trips that lie between 7 AM and 10 AM on May 1, 2018 **corresponding to your local timezone**, **which is not correct**, as the program is being written for someone who as to be at Harvard Book Store by 9 AM, Boston time. Though this is not a fatal error (which will cause your program to crash), you will nevertheless be solving a different problem than what is asked, and your model will be trained with wrong data, even though it will give a spurious result, that might even look very plausible based on how far removed your timezone is from US/Eastern and the variation in the trip frequency and length through the day. This is still a wrong result.
- Since Brandon is running this program presumably from Boston, he would get the correct results as his Python environment is by default set to US/Eastern. You would too if your computer is set to US/Eastern.
- However, this is not a robust solution, as if you were to travel to China for a conference and your computer automatically switched timezones, you would get different results in your predictions, which should not happen (as the customer of this product departure_alarm still cares about being at work at 9 AM in Boston)
- To overcome this problem, you can do the following
    1. Add the line **import os, time** to the import statements on the top of your code
    2. Within the main function (right at the beginning) add the lines
        - **os.environ[‘TZ’] = ‘US/Eastern’**
        - **time.tzset()**
- Doing this will ensure that within your program’s execution, all times you provide will be treated as US/Eastern time. When you evaluate timestamp() for a given datetime, it will assume that that datetime is US/Eastern (with automatic handling of daylight savings). In the same way, when you convert from timestamp to a datetime, it will convert to US/Eastern datetime. **In other words, the rest of the program can stay as it is.**

- Note that python automatically accounts for daylight savings so you don't have to bother about this. That is why we specify timezone as US/Eastern, and not EDT or EST explicitly. For example if you want 7 AM on May 1, 2018, you **DO NOT** have to specify this time with respect to Eastern Standard Time, i.e. as 6 AM. Python knows that May 1, 2018 is under Eastern Daylight Time, and the time in your from_time object should correspond to 7 AM to get the right epoch timestamp.

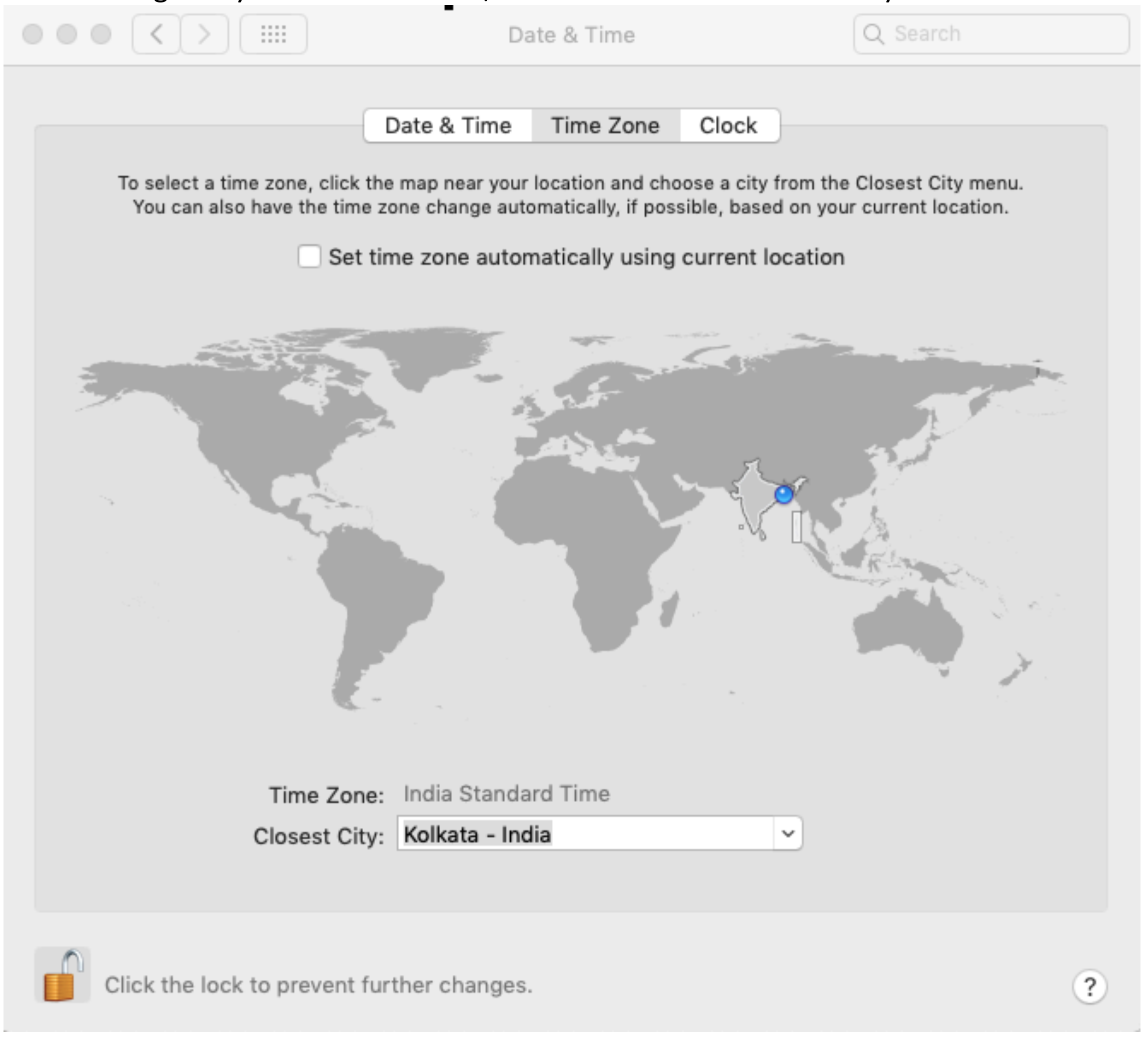
Below shows the output of a smaller version of the program (departure_alarm_debug.py), where we are just pulling the number of trips for every day between May 1 2018 and May 5 2018. I have added print statements to show current timezone. Note that since I'm in Indianapolis, my current timezone is US/Eastern (same as Boston).

```
Current timezone  ('EST', 'EDT')

Number of trips between 7 AM and 10 AM
pulled from MBTA database for
May 1, 2018 through May 5, 2018

2018-05-01 : 18
2018-05-02 : 20
2018-05-03 : 21
2018-05-04 : 21
2018-05-05 : 13
```

Then I changed my timezone to India/Kolkata as shown below on my Mac



To select a time zone, click the map near your location and choose a city from the Closest City menu.
You can also have the time zone change automatically, if possible, based on your current location.

[ ] Set time zone automatically using current location

Time Zone: India Standard Time
Closest City: Kolkata - India

Click the lock to prevent further changes.

Upon rerunning the same python program departure_alarm_debug.py (within this repo), I get the following output

```
Current timezone  ('IST', 'IST')

Number of trips between 7 AM and 10 AM
pulled from MBTA database for
May 1, 2018 through May 5, 2018

2018-05-01 : 12
2018-05-02 : 14
2018-05-03 : 15
2018-05-04 : 13
2018-05-05 : 14
```

Note that now Python thinks I am in IST (Indian Standard Time) based on my changing the timezone on my computer, and even though nothing is changed in the code itself, it pulls different trips, corresponding to 7 AM to 10 AM in IST on those days. This is clearly incorrect. Note how the number of trips that fall within the window of interest on 2018-05-01 **is now 12 instead of 18.**

In the modified version of this code, departure_alarm_debug_fixed.py (also within this repo), I now include the statements specified earlier to force the timezone for this instance of Python to be US/Eastern. Here is the output of that program (even though my system timezone is still set to IST)

```
Current timezone  ('EST', 'EDT')

Number of trips between 7 AM and 10 AM
pulled from MBTA database for
May 1, 2018 through May 5, 2018

2018-05-01 : 18
2018-05-02 : 20
2018-05-03 : 21
2018-05-04 : 21
2018-05-05 : 13
```