SmartFin Data

- Smartfin session data is classified by ride id#
    - For each ride there are multiple csv files with different smartfin ride data
        - Each measurement has an absolute reading, a calibrated reading, and a stability reading
            - Absolute reading: in smartfin units
            - Calibrated reading: in familiar units (celcius, ph, etc)
            - Stability reading: boolean that indicates whether the reading is stable (what does stable mean?)

- All smartfin data is classified as time-series data
    - Each sample represents the same entity
    - Each sample is taken at a different time
    - Used to measure change over time
- Opposed to cross sectional data
    - Each sample represents a different entity
    - All samples are taken at the same time
    - Used to measure differences amongst different entities

- **CSV FILES**
    - **Ocean.csv**
        - UTC (time of reading, UTC format)
        - Time (another time reading probably not important)
        - Temperature 1 (smartfin temperature)
        - Calibrated temperature 1 (celcius)
        - Temperature 1 stable (boolean)
        - Temperature 2
        - Calibrated temperature 2
        - Temperature 2 stable
        - *I believe only nextgen smartfins have the readings for the rest of these columns down vvv*
        - Salinity
        - Calibrated salinity
        - Salinity stable
        - Ph
        - Calibrated ph
        - Ph stable

    - Motion.csv (we're probably gonna spend most of our time with this)

        - UTC
        - Time
        - IMU A1
        - IMU A2
        - IMU A3
            - All A labels are acceleration
        - IMU G1
        - IMU G2
        - IMU G3
            - All G labels are angular velocity
        - IMU M1
        - IMU M2
        - IMU M3
            - All M labels are magnetometer
        - Latitude
        - Longitude
            - Latitude and longitude are frequently na

API Web Scraper Breakdown

For each ride id specified:

1. Query the smartfin API using requests library:
    a. requests.get('https://surf.smartfin.org/ride/{rideID}').text
        i. Returns html of website into variable named HTML_contents
2. Shift through the returned HTML to find csv file of smartfin data using HTML_contents.find()
    a. Build query to find csv id
        i. str_id_csv = 'img id="temperatureChart" class="chart" src="'
    b. Get csv id from query
        i. loc_csv_id = html_contents.find(str_id_csv)
3. Log into smartfin website to get keys to build csv file request
    a. Use off0, off1, and loc_csv_id
4. Build csv file string to find csv file id in the website HTML
    a. csv_id_longstr = html_contents[loc_csv_id+off0:loc_csv_id+off1]
        i. csv_id_longstr will be needed to build the URL filepath of the csv files
5. Build URL to download individual csv files
    a. ocean_csv_url = 'https://surf.smartfin.org/'+csv_id_longstr+{csv file name}'
6. Build dataframes from csv download URLs
    a. ex:
        i. motion_df_small = pd.read_csv(motion_csv_url, parse_dates = [0])

        1. parse_dates: argument that specifies the column we want to try parsing in date form, to make the dates in the dataframe easier to understand
7. Data cleaning
    a. Add elapsed time column to show how much time as elapsed since the first reading
    b. Replace the row number with the timestamp as the index of each row/measurement (only for ocean_df in the example)
    c. Drop na values from dataframe