Intermediate Python **Homework 5**
Due: 11:59 PM Friday, December 11
This is an pairs assignment: find a partner to work with.

Download the file homework5.zip from Canvas and unzip it. It should contain four files: weather5.csv, alarm.py, sensor.py, and hw52.py. To submit, zip up hw51.py, alarm.py, sensor.py (unchanged), and hw52.py as <your_andrew_id>_hw5.zip.

Part 1: Create hw51.py to do the following.
1. Use the file 'weather5.csv'. Create a DataFrame by loading this data into ***weather***. Print weather. For these next few, display with an appropriate label, like 'Weather table columns :'. Display the column names of weather. Display the number of columns of weather (use the first row). Display the number of rows of weather (use the Temperature column).

Clean weather by removing any NaN values from weather. Display the number of rows now.

Write a function named `filter` that takes one parameter: if that value is greater than 100, return 100, but if it's less than 0, return 0, otherwise just return the value. Use filter to change the bad humidity values (see the Data Cleaning notes). Display weather.

2. Write a function called toTime( ) that takes a str parameter – the time field from weather – and returns two str values: the original time converted from the 24-hour clock to the 12-hour clock, and whether the time is AM (before noon) or PM (noon or later). For example, the value 13:15 should return '1:15' and 'PM'.

Create two new columns in ***weather***, named '12Hour' and 'AM-PM', to store these values.

Then create a nicely formatted table to display this data. Print the Time column first, then the 12Hour column, then the AM-PM column, then the other data.

3. Write a function called toMinutes( ) that takes a str parameter – the time field from weather – and returns the time in minutes since midnight. For example, the value 13:15 should return 795 as an int.

Use toMinutes( ) to add another new column to ***weather*** named 'Elapsed'. Display this new version of weather by changing the table code in #2; print Elapsed after the AM-PM column.

For the plotting problems, you may use matplotlib, Seaborn, plotly, or some mix, unless noted otherwise.

4. Plot Elapsed against temperature data as a scatter plot with the label 'Temperature Data', with the axes labeled with the column names.

5. Plot Elapsed against the humidity data as a line graph with the label 'Humidity Data', and the axes labeled with the column names.

6. Use weather and matplotlib to plot a histogram of the wind values, using 5 buckets, with the label 'Wind'.

7. Re-do #4, but use the matplotlib function `ylim(low, high)` to make the y-axis start at 0 and end at the maximum temperature + 1.

8. Plot the linear regression line for temperature versus humidity. Display the correlation coefficient as part of the label `'Temperature x Humidity, Correlation = x'` where x is the correlation coefficient.

9. Create a Seaborn `relplot` for Elapsed against temperature, with separate graphs for AM and PM; use the parameters `size='Temperature'` and `sizes=(20,200)` for the points. Repeat this, but with separate graphs for Cloud types.

10. Create a Seaborn `factorplot` bar graph for Clouds against temperature, with separate graphs for AM and PM. Repeat using a box chart.

Part 2: Make changes to hw52.py and alarm.py
Use the files sensor.py, alarm.py, and hw52.py.

The first file, sensor.py, defines a simple Sensor class; it models a heat sensor, with a location, value (the current temperature at that location), and a trip value (the value at which the sensor signals a problem). It doesn't need to be modified – use it as is.

The second file, alarm.py, is mostly empty; it will define a simple Alarm class, to be coded by you – see below.

The third file, hw52.py, contains a main program that initializes a list of Sensors and a list of Alarms, then displays a menu of choices to display the sensors, display the alarms, test a sensor, and quit; some of this will be coded by you, too; the rest should work as-is.

Run the program either at the command line by typing 'python3 hw52.py' or from within Sypder. The program should run, but some of the output relating to alarms will be blank. The keyword '`pass`' is used several times in the code as a placeholder to make things work; remove this when you fill in your code.

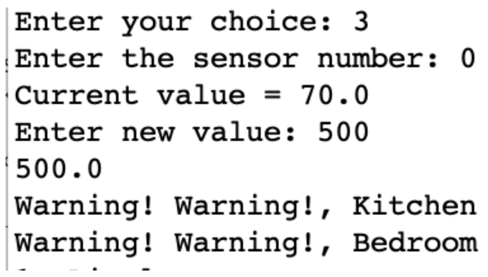
1. Code the ***Alarm*** class. It needs the following methods:
- constructor with one parameter (besides self), location; it sets the location and message, both strings; the message should be hard-coded as 'Warning! Warning!'
- getter for the location
- soundAlarm( ) that displays the location and the message
- the __str__( ) method, because that's always a good idea

2. Finish the ***init( )*** method by filling in the list alarms with two alarms, one for the kitchen and one for the bedroom.

3. Code ***soundAlarms( )***. This should loop over alarms (the parameter, a list of Alarm objects) and sound the alarm for each one.

4. Code choice 2 in the main menu. It should display the list of alarms, similar to the list of sensors in choice 1.

5. Finish choice 3. The missing part should loop over each Sensor object in sensorList; if it has been tripped, call the soundAlarms( ) function, then reset the sensor. This is how it should look when sensor #0 is tested with the new value 500, which causes it to trip. If you then do choice 1, sensor #0's value should be 70.0 again.



```
Enter your choice: 3
Enter the sensor number: 0
Current value = 70.0
Enter new value: 500
500.0
Warning! Warning!, Kitchen
Warning! Warning!, Bedroom
```