CSC120H Lab 7

1 Objectives

- Working with Files
- Practice creating and modifying dictionaries

2 Files Warmup

From the Labs page, download `warmup.txt`. *With your partner*, answer the following questions *without running the code.*

1. What does this code print?

```
f = open('warmup.txt')
for line in f:
    print(line.strip())
f.close()
```

2. What does this code print?

```
f = open('warmup.txt')
f.readline()
f.readline()
for line in f:
    print(line.strip())
f.close()
```

3. What does this code print?

```
f = open('warmup.txt')
for line in f:
    print(line.strip())
    line = f.readline()
f.close()
```

4. What does this code print?

```
f = open('warmup.txt')
for line in f:
    print(line.strip())
for line in f:
    print(line.strip())
f.close()
```

5. What does this code print?

```
f = open('warmup.txt')
print(f.readlines())
f.close()
```

Download `warmup.py` and save it in the same directory as `warmup.txt`. Run `warmup.py` and check your answers.

3 File Processing and Dictionaries

Recorded data is often stored in files for processing and analysis. Reading files in Python allows us to process data and calculate information. In this section, we're going to open files containing real data on temperature. Download `temps.txt` and `files.py` from the Labs page of the course website. This section of the lab involves processing temperature data from `temps.txt`. The starter code, `files.py`, contains docstrings for functions that you need to implement. These functions analyze temperature data in a particular format.
Choose a driver and navigator.

1. Write the body of the helper function `open_temperature_file`. It receives a filename as a parameter, opens the file for reading, reads and discards the header (the first three lines of the file), and returns the open file.
   Make sure you correctly write this function first. You will need it to open files later. Ask your TA for help if you are unsure that you wrote it correctly.

2. The data in `temps.txt` is organized so that each row represents data for one year. Each row therefore has 12 numbers corresponding to the months of that year (January, February, March, etc.) Write the body of the function `average_temp_march` and call it in the Python shell to make sure that it produces the correct result for the data from the file. Of course, given how the data is organized, this means looking at the values in the column corresponding to March. **Hint:** Convert the result to a float along the way.

   Switch driver/navigator

3. Next, implement function `average_temp_month`. It will do the same thing as `average_temp_march`, but it is more general: it finds the average temp for *any* month, not just for March. Copy your `average_temp_march` code to the `average_temp_month` function and make a small change to it, so that it works with any month. Test your function by calling it in the Python shell.

4. Implement the function `higher_avg_temp`. Be careful: this function takes a filename (a `str`) as a parameter (not a file open for reading). If you don't understand the difference, ask your TA. Here your first step should be opening the file (using a function you wrote earlier of course!) Test your function by calling it in the Python shell.

   Switch roles

5. Implement the function `below_freezing`. Test your function by calling it in the Python shell. Show your TA that it works.

Dictionary functions

Fill out the function body for the three dictionary functions. Note which ones have an open file as their parameter, and which have just a filename, requiring you to call your `open_temperature_file` function. Remember to close any files that you open.