Lab Exercise #5

Assignment Overview

This lab exercise provides practice with file processing.
You will work with a partner on this exercise during your lab session. Two people should work at one computer. Occasionally switch the person who is typing. Talk to each other about what you are doing and why so that both of you understand each step.

PART A: DEBUGGING

The art of debugging is an integral part of programming. The following exercise is designed to help you understand file access. The debugger is described in Appendix C of the text.

Open up the program `debug5.py` in Spyder. You also need to download the file `data.txt` in the same folder as your `debug5.py` file. In the upper-right window click on Variable Explorer to open that window. Click 'Remove all variables' button () in Variable Explorer to clear all variables.

Set a breakpoint at the first executable line of the program (`call to the open function`), and start the debugger by pressing the "Debug File" button (`CTRL+F5`). Answer the following questions:

1) The program fails to run in its current state. Debug the issue.
   HINT: Set a breakpoint somewhere before the line that you suspect is problematic. Step through the program in the debugger and keep track of the value stored in the variable `line`

2) The program fails to store the value of `total_height` in `data2.txt`. Debug the issue.
   HINT: There is a missing statement. Contents written to file are buffered in memory and not *saved* until this statement is executed (you don't need the debugger for this one.)

Note: Please *do not* use the `print()` statements during the debug exercise.
Additional Resources
https://docs.spyder-ide.org/debugging.html

PART B: DATA PROCESSING

Given a data file, find the max, min, and average values of columns. Also, create an additional column that is based on the other columns. There will be no error checking required for this lab.
You are provided with a data file named (surprise!) `data.txt`. Data is arranged in columns where each item of data is twelve columns wide (so it is easy to extract data items using slicing):

```
Name        Height(m)   Weight(kg)
Joe         1.82        72.57
Mary        1.60        63.50
Dion        1.90        90.71
Kayla       1.72        66.31
Jose        1.78        70.23
Sofia       1.63        65.12
Erik        1.98        92.21
Sara        1.57        65.77
```

Your task is read the file `data.txt` and calculate the max, min, and average of the height and weight columns and to add an additional column that has the BMI calculated based on each height and weight. Your output will look like this (and needs to be formatted like this). To match the testing on Mimir here is the format string: `"{:<12s}{:<12.2f}{:<12.2f}{:<12.2f}"`

```
Name        Height(m)   Weight(kg)  BMI
Joe         1.82        72.57       21.91
Mary        1.60        63.50       24.80
Dion        1.90        90.71       25.13
Kayla       1.72        66.31       22.41
Jose        1.78        70.23       22.17
Sofia       1.63        65.12       24.51
Erik        1.98        92.21       23.52
Sara        1.57        65.77       26.68

Average     1.75        73.30       23.89
Max         1.98        92.21       26.68
Min         1.57        63.50       21.91
```

The formula for BMI is simple in the metric system: `BMI = weight/height**2`

How do we find the max and min?
The basic concept is similar to counting that you have done multiple times: initialize a value before a loop and update the value every time through the loop.

Based on that concept here is the algorithm to find the minimum of a set of values:

1. Initialize `minimum` to be much larger than any data value in your data, e.g. `10**6`.
2. When you consider each data item, if the data item is smaller than the current `minimum`, you have a new `minimum` (so update it).
3. After considering all data items your `minimum` will contain the smallest.

The algorithm to find the maximum is a tiny variation.

To find the average, add up all the values and count how many values there are. Divide those two to get the average.

Hint: build this program incrementally

1. Begin by simply opening the file and printing every line – now you know that you are correctly working with the correct file.
2. Next ignore the header line—use either `readline()` before the loop or `continue` in the loop.
3. Next find the average of a column: collect a total and count, remembering to initialize before the loop.
4. Next find the minimum of a column using the algorithm above.
5. Continue in such small increments to do the remainder of the program.

★ **Demonstrate your completed program to your TA. On-line students should submit the completed program (named “`lab05a.py`”) for grading via the Mimir system.**

PART C: CREATE A DATA FILE

The second task is to write that output to a file. The steps are

1. Open a file for writing, e.g. `outfile = open("output.txt","w")`
2. Whenever you print, include the argument `file = outfile`
   For example, if you previously had `print(x)`
   you will now have `print(x, file = outfile)`
3. Close the file, e.g. `outfile.close()`
   If you forget this step, nothing will be written to the file. Also, don't forget the parentheses.

Copy your `lab05a.py` to `lab05b.py` and make those modifications to your `lab05b.py` file; test that the output file created is correct.

★ **Demonstrate your completed program to your TA. On-line students should submit the completed program (named "`lab05b.py`") for grading via Mimir system.**

Mimir testing note: to get both files tested at the same time you need to select both of them then upload and submit. Also, for Part C, the output file that your file is compared against uses the same formatting for the header as for the rest of the file, that is, a field of 12 for each element so the `"BMI"` in the header has nine trailing spaces.