Assignment #3

420-SF2-RE: Data Structures and Object Oriented Programming
April 7, 2025

Late submissions are not accepted! Submissions via MIO are not accepted!

Constraints:

- For this assignment, you may only use what has been covered in class up to and including exceptions.
- You will be graded on the efficiency of your code as well as the design and simplicity. Although the provided data may be small, you should write your code so that if the size of the data were to increase, your code will run efficiently. Make sure you adhere to the assignment instructions.
- Make sure your function names and arguments passed to them are exactly as stated in this assignment so that the automated test cases will not fail because of naming functions incorrectly.

- For this assignment you **may not use** any **`comprehensions`**, the functions: **`filter`**, **`enumerate`**, **`sort`**, **`sorted`**, opening files using **`with`**, **`global`** variables. Doing so will result in a grade of 0 for Assignment#3.
- All `import` statements must be at the top of your `.py` file, then followed by function definitions, then the main program.
- For this assignment you are not allowed to define any extra/helper functions.

Violating any of the above constraints will result in sever deduction of marks and/or a grade of 0 for Assignment#3.

Part 1: due at the end of class on April 7

- 70% of Assignment#3 grade
- On Git in your Assignment#3 folder you are to add, commit, push your `.py` and `.txt` files called `temperature.py` and `data.txt`
- On Moodle under Assignment#3 Part-1 Submission you are to upload your `.py` file called `temperature.py`
- Make sure you add your name and student ID at the top of your `temperature.py` file as comments.
- **Failure to follow submission instructions properly will result in a grade of 0 for Part-1 of Assignment#3.**

Functions to Implement for Part 1 (due end of class today):

If a portion of your code is not working comment it out in your `.py` file before submission. For any submitted code that does not compile will receive an automatic grade of 0.

1. **(time estimate: 5 minutes)** Write a function called `toCelsius` that takes a temperature in Fahrenheit and returns the corresponding temperature in Celsius rounded to 2 decimal places. Here is the formula that you can use: `C = (F - 32)*(5/9).`

   For example:
   If the temperature in Fahrenheit is `77`, then your function returns `25.0`.

2. **(time estimate: 15 minutes)** Download the file `data.txt` from Moodle. In your main program, open the file `data.txt` for reading only. From this file you are to read all the lines from the year `1964` to `1975` into a dictionary `temp_dict`. That is, all the lines in your file before `1964` should not appear in `temp_dict` dictionary. In `temp_dict` the keys are the years (as an integer), and the values are the list of temperatures from JAN to DEC (as floats and changed to Celsius).
   HINT: the map function may be useful to do fast change of floats from strings and from Fahrenheit to Celsius.

3. **(time estimate: 10 minutes)** Write a function called `avgTempYear` that takes two arguments: a dictionary (of same format as you read from the `data.txt` file) and year. The function returns the average temperature for the given year in the dictionary rounded to 2 decimal places. If the provided year is not in the dictionary, your code should use exceptions to handle it as we have done in class by using the `try-except-else` block as needed. In the case of invalid year, you should print a friendly message and your function in such a case should return nothing.

4. **(time estimate: 10 minutes)** Write a function `topThreeYears` that takes a dictionary (of same format as you read from the `data.txt` file) and returns a list of the three largest averages in descending order among all the years. Pay attention to what data structures you may use for efficiency.

5. **(time estimate: 15 minutes)** Write a function called `avgTempMonth` that takes a dictionary (of same format as you read from the `data.txt` file) and a month (as a string of 3 characters) and returns the average temperature for the given month across all years rounded to 2 decimal places.

Note that you may create a dictionary in your function to help you go back and forth between string and integer representation of months.
For example: `month_dict = {'JAN': 1, 'FEB': 2, 'MAR': 3, etc.}`

Part 2: on April 9 by 23:59

- 30% of Assignment#3 grade
- On Git to your `temperature.py` file commit and push your changes of Part-2. Also, make sure you add, commit, push any new `.txt` files that were created.
- On Moodle under Assignment#3 Part-2 Submission upload your final `.py` and `.txt` files called `temperature_part2.py` and `data_celsius.txt`. You must also upload a .txt file with a link to your final git submission.
- Make sure you add your name and student ID at the top of the `temperature_part2.py` file as comments.
- **Failure to follow submission instructions properly will result in a grade of 0 for Part-2 of Assignment#3.**

Functions to Implement for Part 2:

For any of the functions below or the Main Program that does not compile will receive an automatic grade of 0. If a portion of your code is not working comment it out in your `.py` file before submission.

6. Write a function `belowFreezing` that takes a dictionary (of same format as you read from the `data.txt` file) and returns all months that have had a temperature value below freezing in some year. On the given data, the output should be the months of January, February, March and December.

7. In your main program you are to write to a file called `data_celsius.txt`. Create this file for writing using Python (don't manually create the file in the folder).
   - In `data_celsius.txt` the first 4 lines are the same as the 4 lines in `data.txt`. Do not copy and paste these 4 lines. Write Python code that automates this process for you.
   - The next 12 lines are the 12 key-value pairs from your `temp_dict`. When you write this data to your file, make sure the display format and spacing in `data_celsius.txt` is the same as that of `data.txt`.