CSc 110 Assignment 9:
File I\O

Reminder: Your code is to be designed and written by only you and not to be shared with anyone else. See the Course Outline for details explaining the policies on Academic Integrity. Submissions that violate the Academic Integrity policy will be forwarded directly to the Computer Science Academic Integrity Committee. All materials provided to you for this work are copyrighted, these and all solutions you create for this work cannot be shared in any form (digital, printed or otherwise). Any violations of this will be investigated and reported to Academic Integrity.

Learning Outcomes:

When you have completed this assignment, you should understand:

- Reading from a file.
- Populating dictionaries/ looking up values in dictionaries
- Reading documentation
- Designing, calling and writing helper functions

Getting started

1. Download the starter file provided called `assignment9.py` and open it in your Wing editor
2. Download the provided input csv files to the same directory as your `assignment9.py`
3. Design the functions according to the specifications in the documentation provided in the starter file and using the further explanation given in the ***Background Information*** section below.

Submission

1. Double check your file before submission to avoid a **zero grade** for your submission for issues described in the Grading section of this document:
    a. Open and run `assignment9.py` in your Wing editor using the green arrow.
    You should see no errors and no output after the shell prompt.
    If there are errors, you must fix them before submitting.
    If there is output printed to the shell, find and remove any top-level print statements and/or top level function calls.
    b. At the shell prompt in Wing (>>>) type the following: `import assignment9`
    You should see no errors and no output after the shell prompt.
    If there are errors, you must fix them before submitting.
    If there is output printed to the shell, find and remove any top-level print statements and/or top level function calls.
    c. At the shell prompt in Wing (>>>), make calls to the required functions to ensure you have named them correctly. Ensure the function is exhibiting the expected behaviour when called and does not contain any unexpected output. You should be able to make calls to the following functions with expected input. Note: When making these calls, you must precede it with the module name ie. `assignment9.read_file('11lines_data.csv')`
        i. read_file
        ii. query
2. Upload your `assignment9.py` containing the completed function designs to BrightSpace

Grading:

- Late submissions will be given a **zero grade**.
- The files you submit must be named **`assignment9.py`**
  The filenames must be EXACT for them to work with our grading scripts. Errors in the filenames will result in a **zero grade**.
  Example mistakes often made by students include but are not limited to: spelling errors, different case letters, space characters and incorrect extension (not using .py)
- Your function names and the order of your function arguments must match EXACTLY as specified in this document or you will be given a **zero grade**. Use the example tests we give you to ensure your function header is correct.
- Your submission must not contain any print statements that are not required in the specification or any top-level calls to functions. This unexpected code can cause the automated tester to crash and will result in a **zero grade**.
- We will do **spot-check grading** in this course. That is, all submissions are graded BUT only a subset of your code might be graded. You will not know which portions of the code will be graded, so all of your code must be complete and adhere to specifications to receive marks.
- Your code must run without errors with Python 3. If you tested your configuration with setup.py file this would have verified you are using Python 3. Code that generates errors cannot be tested and will be given a **zero grade**.

Marks will be awarded for correctness, considering:

- the function signature matches the description given (has the name and arguments EXACTLY as specified)
- the function has the expected behaviour

and for code quality according to software engineering properties such as:

- documentation in docstring format: type hints, including return type, purpose, examples as demonstrated in lecture.
- Test coverage – examples within the function docstring cover all boundary cases of all conditions within your function
- readability
  - use of whitespace
  - splitting complex computation or long statements across multiple lines
- meaningful variable names
  - lower case, starting with an alpha-character
- proper use of constants (avoid magic numbers)
  - defined above all function definitions
  - in UPPERCASE
- use of code constructs (functions, variables) to:
  - use of helper functions where appropriate
  - eliminate redundant code and redundant computation
  - make complex code easier to read

Background Information:

In this assignment you will be writing functions that read data from comma separated data files and creating structured lists of tuples to represent this data. You will then be writing functions that work with these lists of tuples to answer questions about the data.
The data we are using has been taken from https://www.kaggle.com/shivamb/netflix-shows
The downloaded data was cleaned (removed additional commas and newlines) so that you can assume the only commas are between columns in the data and the only newlines are at the end of a row.

We have provided you with 3 sample input files:

- 0lines_data.csv – this file has just the header row
- 11lines_data.csv – this file has the header row followed by 11 rows of data
- large_data.csv – this file has the header row followed by 5583 rows of data

The smaller files are used in some of the testcases within the provided docstrings.

WINDOWS users In order to read the special characters from the file those on a Windows computer must pass an additional encoding argument to the open function as follows:

```
file_handle = open(filename, 'r', encoding="utf8")
```

Examples have been provided to allow you to test your functions and understand the problem.
`doctest` has its limitations so the length of the lines do wrap around the screen making it challenging to read but we felt it important to give you tests that would work with `doctest`.
We used the \ character to break lines to try to make the examples a little bit more readable.

NOTE: when grading we will be not be using these exact tests. That is, we will vary the inputs when grading. The tests provided do not necessarily provide full coverage but are given as examples to help you understand the problem.

TIPS for working on this assignment:

- open the smaller file and have a look at its contents in relation to the GLOBAL CONSTANTS you are given in the python file
- We have provided you with the `Date` type aliases that used last week. Feel free to replace it with your own as long as yours were correct.
- You will find that some of the functions that you wrote in previous assignments useful. Again, if yours were correct, you can copy them to your `assignment9.py` file so you can call them as helper functions.
- when working on each function,
    - take your time to read the documentation, including the examples so that you understand the problem you are trying to solve
    - write it incrementally, printing out intermediate results so you can see the data you are dealing with
- You are encouraged to design additional helper functions as you need. When you do, ensure you provide full documentation and ensure they are correct. Remember we grade not only based on correctness but on code quality!
- Always think about how other functions that you have written might be able to be called as helper functions instead of duplicating code you have already written.
- You are free to use Python list's `sort` method. This method will mutate the list into sorted order (it does not return a value. The values are sorted in lexicographical (dictionary) order.
  Example:
  ```
  my_list = ['bad', 'cab', 'car', 'Car']
  my_list.sort()
  ```
  after the call to `sort` then `my_list` will be: `['Car', 'bad', 'cab', 'car']`

Function Specifications:

The function specifications can be found within the documentation provided in `assignment9.py` starter file.