Department of Computer Science and Software Engineering
The University of Western Australia
CITS1401
Computational Thinking with Python
Project 2 Semester 1 2023
(Individual project)

Submission deadline: Friday 26th May 2023 @ 6:00 PM
Value: **20%** of CITS1401.

Project description:
You should construct a Python 3 program containing your solution to the following problem and submit your program electronically on Moodle. The name of the file containing your code should be your student ID e.g. 12345678.py. No other method of submission is allowed. Please note that this is an individual project. Your program will be automatically run on Moodle for sample test cases provided in the project sheet if you click the "check" link. However, your submission will be tested thoroughly for grading purposes after the due date. Remember you need to submit the program as a single file and copy-paste the same program in the provided text box. You have only one attempt to submit so don't submit if you are not satisfied with your attempt. All open submissions at the time of the deadline will be automatically submitted. There is no way in the system to open a closed submission or reverse your submission.

You are expected to have read and understood the University's guidelines on academic conduct. In accordance with this policy, you may discuss with other students the general principles required to understand this project, but the work you submit must be the result of your own effort. Plagiarism detection, and other systems for detecting potential malpractice, will therefore be used. Besides, if what you submit is not your own work then you will have learnt little and will therefore, likely, fail the final exam.

You must submit your project before the deadline listed above. Following UWA policy, a late penalty of 5% will be deducted for each day (or part day), after the deadline, that the assignment is submitted. No submissions will be allowed after 7 days following the deadline except approved special consideration cases.

Project Overview:
The XYZ organisation collected world population data for all countries over the world using different parameters such as population in 2020, annual change, net change, per kilometre density, land area, median age, etc. The data also shows the regions for each country as shown in Figure 1. It can be observed that each region has many countries and has its own population.

You are required to write a Python 3 program that will read a CSV file. After reading the file, your program is required to complete the following statistical tasks:

1) Create a dictionary and calculate the following statistical information to store in it:
    a. Calculate standard error of population for each region.
    b. Calculate the cosine similarity between population and land area for each region.
2) Create a dictionary that contains information for each region and for each country. The information required is: population, net change, percentage of population with respect to a region, density of population, and rank of population in the region.

Requirements:

1) You are not allowed to import any external or internal module in python. While use of many of these modules, e.g. csv or math is a perfectly sensible thing to do in production setting, it takes away much of the point of different aspects of the project, which is about getting practice opening text files, processing text file data, and use of basic Python programming skills.

2) Ensure your program does NOT call the input() function at any time. Calling the input() function will cause your program to hang, waiting for input that automated testing system will not provide (in fact, what will happen is that if the marking program detects the call(s), it will not test your code at all which may result in zero grade).

3) Your program should also not call print() function at any time except for the case of graceful termination (if needed). If your program has encountered an error state and is exiting gracefully then your program needs to empty dictionaries and print an appropriate message. At no point should you print the program's outputs instead of (or in addition to) returning them or provide a printout of the program's progress in calculating such outputs.

Input:
Your program must define the function `main` with the following syntax:
```
def main(csvfile):
```

The input arguments for this function are:

- `csvfile`: The name of the CSV file (as string) containing the record of the population of all countries in the world. Below is the first row of the sample file:

Country	Population	Yearly Change	Net Change	Urban	Land Area	Med Age	Regions
China	1439323776	0.0039	5540090	0.61	9388211	38	Asia

Output:
Two outputs are expected:

1) A dictionary which will have region as key, and a list as a value containing the standard error of population and cosine similarity between population and land area, in the following format:
   `'Region': [Standard Error value, Cosine Similarity value]`

2) A nested dictionary which will store region as key and its value is a dictionary item, where country in that region is key and information about that country in a list as its value. The list will contain the following information for each country: population, net change, percentage of population with respect to total population of the region, density of population, and rank of population in the region. Below is the format:

   ```
   'Region1': {'Country1':[population, net change,
   percentage of population with respect to Region1, density
   of population, rank of population in Region1],
   'Country2': [population, net change, percentage of
   population with respect to Region1, density of
   population, rank of population in Region1]}, 'Region2':
   {'Country1':[population, net change, percentage of
   population with respect to Region2, density of
   population, rank of population in Region2], 'Country2':
   [population, net change, percentage of population with
   respect to Region2, density of population, rank of
   population in Region2]}
   ```

   If two or more countries have the same population (rank), then their ranks need to be sorted by their population density in descending order. If both population and population density are same for multiple countries then they need to be ranked by alphabetical order (ascending). For instance, country "A" and "B" have same population in the region and country "A" has higher population density than "B", then country "A" and "B" will be ranked "N" (higher) and "N+1"(lower) respectively.

Note: All the float values should be rounded by 4 decimal places.

Examples:
Download `countries.csv` file from the folder of Project 1 on LMS or Moodle. An example of how you can call your program from the Python shell (and examine the results it returns) are:

```
>>> dict1, dict2 = main('countries.csv')
```

The output variables returned are:

```
>>> dict1['northern america']
[80089583.5645, 0.7841]

>>> dict2['northern america']
{'united states':[331002651, 1937734, 89.7357,36.1854 ,1],
'canada': [37742154, 331107, 10.232, 4.1504, 2], 'bermuda':[
62278, -228, 0.0169, 1245.56, 3], 'greenland':[56770, 98,
0.0154, 0.1383, 4]}
```

Assumptions:
Your program can assume the following:

- The order of columns can be different than the order provided in the sample file. Also there can be extra columns in the CSV file. Moreover, rows can be in random order except the first row containing the headings.
- All string data in the file is case-insensitive, which means “Asia” is same as “ASIA”. You program needs to handle the situation to consider both to be the same.
- There can be missing or invalid data, for instance, population or area cannot be negative, there cannot be two identical countries in a region, there should not be zero population and land area, name of a region and country cannot be empty or null, etc. You need to think of other cases yourself. If there is any invalid data then entire row(s) should be ignored.
- The necessary formulas are provided at the end of this document.

Important grading instruction:
Note that you have not been asked to write specific functions. The task has been left to you. However, it is essential that your program defines the top-level function *main(csvfile)* (hereafter referred to as *“main()*" in the project documents to save space when writing it. Note that when *main()* is written it still implies that it is defined with its input argument). The idea is that within *main(),* the program calls the other functions. (Of course, these functions may then call further functions.) This is important because when your code is tested on Moodle, the testing program will call your *main()* function. So if you fail to define *main(),* the testing program will not be able to test your code and your submission will be graded zero. Don’t forget the submission guidelines provided at the start of this document.

Marking rubric:
Your program will be marked out of 30.

22 out of 30 marks will be awarded automatically based on how well your program completes a number of tests, reflecting normal use of the program, and also how the program handles various states including, but not limited to, different numbers of rows in the input file and / or any error states. You need to think creatively what your program may face. Your submission will be graded by data files other than the provided data file. Therefore, you need to be creative to look into corner or worst cases. I have provided few guidelines from ACS Accreditation manual at the end of the project sheet which will help you to understand the expectations.

8 out of 30 marks will be awarded on style (5/8) "the code is clear to read" and efficiency (3/8) "your program is well constructed and run efficiently". For style, think about use of comments, sensible variable names, your name at the top of the program, etc. (Please watch the lectures where this is discussed)

Style Rubric:

0	Gibberish, impossible to understand or style is poor
1-2	Style is fair
3-4	Style is good or very good, with small lapses
5	Excellent style, really easy to read and follow

Your program will be traversing text files of various sizes (possibly including large csv files) so you need to minimise the number of times your program looks at the same data items.

Efficiency rubric:

0	Code too complicated to judge efficiency or wrong problem tackled
1	Very poor efficiency, additional loops, inappropriate use of readline()
2	Acceptable or good efficiency with some lapses
3	Excellent efficiency, should have no problem on large files, etc

Automated testing is being used so that all submitted programs are being tested the same way. Sometimes it happens that there is one mistake in the program that means that no tests are passed. If the marker is able to spot the cause and fix it readily, then they are allowed to do that and your - now fixed - program will score whatever it scores from the tests, minus 4 marks, because other students will not have had the benefit of marker intervention. Still, that's way better than getting zero. On the other hand, if the bug is hard to fix, the marker needs to move on to other submissions.

Extract from Australian Computing Society Accreditation manual 2019:
As per Seoul Accord section D, a complex computing problem will normally have some or all of the following criteria:

- involves wide-ranging or conflicting technical, computing, and other issues;
- has no obvious solution, and requires conceptual thinking and innovative analysis to formulate suitable abstract models;
- a solution requires the use of in-depth computing or domain knowledge and an analytical approach that is based on well-founded principles;
- involves infrequently-encountered issues;
- is outside problems encompassed by standards and standard practice for professional computing;
- involves diverse groups of stakeholders with widely varying needs;
- has significant consequences in a range of contexts;
- is a high-level problem possibly including many component parts or sub-problems;
- identification of a requirement or the cause of a problem is ill defined or unknown.

Necessary formulas:

1. **Cosine similarity**

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

You can find more details at https://en.wikipedia.org/wiki/Cosine_similarity

2. **Standard Error**

Step 1: Calculate the standard deviation:
Standard deviation:

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}}$$

where $x_1, x_2, x_3 \ldots\ldots x_n$ are observed value in sample data. $\bar{x}$ is the mean value of observations and N is the number of sample observations.
Step 2: Calculate Standard Error
Standard Error:

$$SE_{\bar{x}} = \frac{S}{\sqrt{n}}$$

where, *S* is the standard deviation and *n* is the number of observations.

3. Density

Density of a country can be calculated using the following formula:

$$D = P/L$$

where P is the population of a country and L is the land area of a country.

4. Percentage

Percentage = (P*100)/N

where, P is the population of a specific country, N is the total population of a region.