CITS1401 Computational Thinking with Python

Project 2, Semester 2, 2024

Submission deadline: 18th Oct. 2024, 11:59 PM

Total Marks: **30 (Value: 20%)**

Project Submission Guidelines

You should construct a Python 3 program containing your solution to the given 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, this does not test all required criteria and your submission will be thoroughly tested manually 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 the closed submission and reverse your submission.
- 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, including special consideration cases.

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.

Project Overview

This project focuses on analysing hospital data from multiple countries, examining various key factors such as patient mortality rates, disease-specific admissions, staff numbers and patient demographics etc. The dataset includes information about each hospital's ID, country, the number of staff and patients, male and female patient distribution, hospital category, bed availability, and mortality figures for 2022 and 2023. The dataset includes a diverse range of hospital categories, such as *research, children, general, rehabilitation, teaching, speciality, psychiatric, day surgery, standalone emergency, and outpatient* hospitals, with each category including a varying number of hospitals.

Dataset

The dataset for this project comprises two key files: a CSV and a TXT file. The CSV file includes information on countries, hospital IDs, hospital categories, and the number of deaths recorded in 2022 and 2023, among other details. Meanwhile, the TXT file provides a detailed account of patient admissions for each country and corresponding hospitals, specifying the number of cases for Covid, Stroke, and Cancer in 2022.

You are required to write a Python 3 program that will read two different files: a CSV file and a TXT file. After reading the data file(s), your program will perform four different tasks outlined below.

Tasks

i. Task 1: Generate Country-Specific Hospital Data

Create a list of three dictionaries `[Country_to_hospitals, Country_to_death, Country_to_covid_stroke]` to map each country to its hospital IDs, deaths in 2022, and the total number of patients admitted for covid and stroke in 2022, respectively. The dictionaries will have country names as keys and lists as values, where each entry in the lists corresponds to a specific hospital. The order of the entries in all lists across the dictionaries must be perfectly aligned to ensure consistency across the data. For example:

- `Country_to_hospitals['argentina'] = ['H1', 'H2', 'H3']`
- `Country_to_death['argentina'] = [10, 15, 8]` (number of deaths for H1, H2, H3 in 2022)
- `Country_to_covid_stroke['argentina'] = [50, 60, 45]` (covid and stroke admissions for H1, H2, H3 in 2022)

ii. Task 2: Calculate Cosine Similarity

For each country, compute the *Cosine Similarity* between the number of deaths in 2022 and the total number of patients admitted for covid and stroke. Store these results in a dictionary (`Cosine_dict`) with the country names as the keys and the *Cosine Similarity* as the values.

iii. Task 3: Analyze Variance in Cancer Admissions

Create a dictionary (`Variance_dict`) that stores the variance in cancer patient admissions across hospitals in each country for a specified hospital category (e.g., `'children'`). The key is the country name, and the value is the variance in the number of cancer admissions for the specified hospital category.

iv. Task 4: Generate Hospital Category Statistics

Create a nested dictionary (`Category_Country_dict`) to store data for each hospital category. The outer dictionary will use the hospital category as the key, while the value will be another dictionary that maps country names to lists of key statistics, including:

- The average number of female patients treated.
- The maximum number of staff in hospitals.
- The percentage change between the average number of deaths from 2022 to 2023

Input

Your program must define the function main with the following syntax:

```
def main (CSVfile, TXTfile, category):
```

The input arguments for this function are:

- `CSVfile:` The name of the CSV file (as string) containing records of hospital data across different countries. The first row of the CSV file contains the column headings.
- `TXTfile:` The name of the TXT file (as string) containing records of patients admitted for covid, stroke, and cancer across various hospitals in different countries.
- `category:` A string representing the category of hospitals to be analysed. The `CSV file` contains information about hospitals from multiple categories.

Output

The following four outputs are expected:

i. **OP1=** list of dictionary items: `[Country_to_hospitals, Country_to_death, Country_to_covid_stroke]`.

   a) `Country_to_hospitals` maps each country name to a list of hospitals within that country. In this dictionary, the key is the name of the country, and the value is a list of hospital IDs in that country.

   b) `Country_to_death` maps each country name to a list of the number of deaths in 2022 for each hospital in that country. The key is the name of the country, and the value is a list where each element represents the death count for a corresponding hospital in that country.

   c) `Country_to_covid_stroke` maps each country name to a list of the total number of patients admitted for covid and stroke in each hospital in that country. The key is the name of the country, and the value is a list where each element represents the total number of covid and stroke admissions for a hospital in that country.

Important Note: It is essential that the order of the entries in the values (lists) for all three dictionaries (`Country_to_hospitals`, `Country_to_death`, and `Country_to_covid_stroke`) is perfectly aligned. For example, the first entry in `Country_to_death['brazil']` and `Country_to_covid_stroke['brazil']` should relate to the first hospital ID in `Country_to_hospitals['brazil']`, and so on for the remaining entries.

ii. **OP2=** `Cosine_dict`: A dictionary where the key is the country name, and the value is the *Cosine Similarity* between the number of deaths in 2022 and the total number of patients admitted for covid and stroke in that country.

iii. **OP3=** `Variance_dict`: A dictionary where the key is the country name, and the value is the variance in the number of `Cancer` patient admissions for a specified `category`, such as `'children'` across hospitals within that country.

iv. **OP4=** `Category_Country_dict`: A nested dictionary that stores information for each hospital category 'C'. The outer dictionary uses the hospital category as the key, and the value is another dictionary 'D'.

   a) The dictionary 'D' uses the country name as the key, and the value is a list containing the following data for hospitals in category 'C' within that country:

- The average number of female patients treated in hospitals under category ‘C’.
- The maximum number of staff working in hospitals within category ‘C’.
- The percentage change between the average number of deaths from 2022 to 2023 for hospitals in category ‘C’.

All returned numeric outputs must contain values rounded to four decimal places (if required to be rounded off). Do not round the values during calculations. Instead, round them only at the time when you save them into the final output variables.

Requirements

i. You are not allowed to import any external or internal module in python.

ii. 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).

iii. Your program should also not call `print()` function at any time except for the case of graceful termination (if needed). If your program encounters an error state and exits gracefully, it should return a cosine-similarity/variance/mean/percentage-change value of zero and print an appropriate error message. At no point should you print the program’s outputs or provide a printout of the program’s progress in calculating such outputs. Outputs should be returned by the program instead.

iv. Do not assume that the input file names will end in `.csv` or `.txt`. File name suffixes such as `.csv` and `.txt` are not mandatory in systems other than Microsoft Windows. Do not enforce within your program that the file must end with a specific extension, nor should you attempt to add an extension to the provided file name. Doing so can result in loss of marks.

Examples

Download `hospital_data.csv` and `disease.txt` files from the folder of Project 2 on LMS or Moodle. An example of how you can call your program from the Python shell and examine the results it returns, is provided below:

```
>> OP1, OP2, OP3, OP4 = main('hospital_data.csv', 'disease.txt', 'children')
```

Few details of the returned output variables are:

```
>> len(OP1)

3

>> OP1[0]['afghanistan']

['4eb9d3e5cf79b91', 'bba52b87bb6a32f','8a9190a50adf241']

>> OP1[1]['afghanistan']

[20, 2, 12]

>> OP1[2]['afghanistan']

[830, 3898, 6854]

>> len (OP2)

32

>> OP2['afghanistan']

0.5746

>> OP2['albania']

0.9257

>> OP3['afghanistan']

785004.5

>> OP3['brunei darussalam']

24420.5

>> len(OP4['children'])

32

>> OP4['children']['canada']

[3925.4, 4448, 22.0588]
```

Assumptions

Your program can assume the following:

1. All string data in the CSV file and TXT file is case-insensitive, which means "albania" is same as "AlbAnia" or "covid" is same as "Covid". Your program needs to handle the situation to consider both strings to be the same.
2. In the CSV file, the order of columns can be different than the order provided in the sample file. Also, there can be extra or less columns in the testing files. Moreover, rows can be in random order except the first row which contains the headings.
3. There can be missing or invalid data in the CSV file; and in such an instance, the entire row should be ignored. Some examples of invalid data can be negative or zero number of staff, null/empty values in the required columns. You need to think of other invalid cases yourself. In case any part of the calculation cannot be performed due to zero values or other boundary conditions, do a graceful termination by printing an error message and returning a zero value (for numbers), None for (string) or empty list depending on the expected outcome. Your program must not crash.
4. All hospital IDs will be unique. There will be no missing or invalid data in the TXT file.
5. 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, TXTfile, category)` (commonly 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 three input arguments). 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

24 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 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 investigate 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.

6 out of 30 marks will be awarded on style (3/6) "the code is clear to read" and efficiency (3/6) "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, student ID, etc. (Please watch the lectures where this is discussed).

Style Rubric:

0	Gibberish, impossible to understand
1	Style is really poor or fair.
2	Style is good or very good, with small lapses.
3	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 can 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.
- are 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, $\boldsymbol{cos(\theta)}$

$$cos(\theta) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

where,

x_i = first set of data

y_i = second set of data

n = the number of samples

2) Variance, S^2

$$S^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$$

where,

S^2 = sample variance
x_i = the value of the one observation
$\bar{x}$ = the mean value of all observations
n = the number of observations

3) Average Percentage Change:

Percentage change between the average number of deaths from 2022 to 2023, for all hospitals of specific hospital type within a country.

$$Percentage\ Change\ in\ Average\ Deaths\ (PCAD)$$

$$PCAD = \frac{(Average\ deaths\ in\ 2023) - (Average\ deaths\ in\ 2022)}{Average\ deaths\ in\ 2022} \times 100\%$$