Department of Computer Science and Software Engineering

The University of Western Australia

CITS1401

Computational Thinking with Python

Project 1, Semester 2, 2024

(Individual project)

Submission deadline: 23:59 PM, 13 September 2024.

Total Marks: 30

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 until you are 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/modify/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) i.e., 24 hours after the deadline, that the assignment is submitted.
- No submissions will be allowed after 7 days following the deadline except approved 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:

In the rapidly expanding world of e-commerce, platforms like Amazon provide vast amounts of data that can offer valuable insights into various aspects of product performance. This project aims to analyze Amazon data for different products within specific categories, utilizing key parameters such as *product ID, product name, category, discounted price, actual price, ratings, rating coun*t etc., The data set includes a diverse range of categories, each with multiple products, allowing us to identify trends and patterns specific to each category.

You are required to write a Python 3 program that will read two different files: a CSV file and a TXT file. Your program will perform four different tasks outlined below. While the CSV file is required to solve all the tasks (Tasks1-4), the TXT file is only required for the last task (Task 4).

After reading the CSV file, your program is required to complete the following:

- **Task 1: Identify Extreme Discount Prices**
  Find the product ID with the highest *discounted price* and the product ID with the lowest discounted price for a specific category.
- **Task 2**: **Summarize Price Distribution**
  Provide a summary of the '*actual price*' distribution i.e., mean, median and mean absolute deviation of products for a specific category, considering only the products with a *rating count* higher than 1000.
- **Task 3**: **Calculate Standard Deviation of Discounted Percentages**
  Calculate the standard deviation of the *discounted percentages* for products with *rating* in the range 3.3≤*rating*≤4.3, for each category.
- **Task 4**: **Correlate Sales Data**
  Find the correlations between the sales of the products identified in Task 1 (products with highest and lowest *discounted prices* for a specific category).
  <u>Steps:</u>
  - Read the TXT file which contains the sales data for several years, such as 1998-2021. Each line lists product IDs and the units sold for that year. If a product ID is not mentioned in a line, it means zero units sold for that year.

- Create two lists, one for the sales of the product with the highest discounted price and another for the sales of the product with the lowest discounted price identified in Task 1.
- Process each line of the TXT file to determine the number of units sold each year.
- Each list should have one entry per year, with the total number of entries matching the number of lines in the TXT file.

Finally, calculate the correlation coefficient between the two sales lists.

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 structures, in this case *lists* and *loops*.

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 encounters an error state and exits gracefully, it should return a correlation/standard deviation/mean/median 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.

4) 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.

Input:

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

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

The input arguments for this function are:

1. `CSVfile`: The name of the CSV file (as string) containing the record of the Amazon's product data.
2. `TXTfile`: The name of the TXT file (as string) containing the record of Amazon's product sales.
3. `category`: A string representing the *category* to be analysed. The Amazon's product data contains multiple categories.

Output:

The following four outputs are expected:

i) **`OP1`**`= [Product ID1, Product ID2]`: A list that contains two items, ID of the product with the highest discounted price, ID of the product with the lowest discounted price. Your output should be stored in a list in the following order:
[highest discounted price product ID, lowest discounted price product ID]
For example: `['b07vtfn6hm', 'b08y5kxr6z']`
Note: *If multiple products have the same highest discounted price, select the product ID that comes first when the product IDs are sorted in ascending order. Apply the same rule for the lowest discounted price.*

ii) **`OP2`**`= [mean, median, mean absolute deviation]`: A list containing three statistical measures i.e., mean, median, and mean absolute deviation of the *actual price* for products within a given category, considering only those products with a *rating count* higher than 1000. The output should be stored in a list in the following order:
`[mean, median, mean absolute deviation]`
For example: `[2018.8, 800, 2132.48]`

iii) **`OP3`**`= [STD1, STD2, ..., STDN]`: A list containing the standard deviation of the *discounted percentages* for products within the *rating* in the range 3.3 to 4.3 (3.3 ≤ *rating* ≤ 4.3) of each category. The output should be sorted in the descending order. The expected output is a list with values sorted in the descending order.
For example: `[0.297, 0.2654, 0.2311, 0.198, 0.1701, 0.1596, 0.0071]`

iv) **`OP4`**`= Correlation`: A numeric value representing the correlation between the sales of a product with the highest discounted price and the lowest discounted price found in the task 1 above. The expected output is a single float value.
For example: `-0.0232`

All returned numeric outputs (both in lists and individual) 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.

Examples:

Download `Amazon_products.csv` and `Amazon_sales.txt` 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) is provided below:

```
>>>OP1,    OP2,    OP3,    OP4=    main('Amazon_products.csv',
'Amazon_sales.txt', 'Computers&Accessories')

>>>OP1
['b07vtfn6hm', 'b08y5kxr6z']

>>> OP2
[2018.8, 800, 2132.48]

>>> OP3
[0.297, 0.2654, 0.2311, 0.198, 0.1701, 0.1596, 0.0071]

>>> OP4
-0.0232
```

Assumptions:

Your program can assume the following:

1. Anything that is meant to be string (e.g., header) will be a string, and anything that is meant to be numeric will be numeric.
2. All string data in the CSV file and TXT file is case-insensitive, which means "Computers&accessories" is same as "Computers&Accessories" or "B08Y5KXR6Z" is same as "b08y5kxr6z". Your program needs to handle the situation to consider both strings to be the same.
3. In the CSV file, the order of columns in each row will follow the order of the headings provided in the first row. However, rows can be in random order except the first row which contains the headings.
4. No data will be missing in the CSV file; however, values can be zero and must be accounted for when calculating averages and standard deviations.
   [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.]
5. Each line in the TXT file will correspond to a unique year, with no repetition of years. The number of years may vary, so avoid hard coding.
6. All the *product IDs* in the CSV file will be unique.
7. The `main()` will always be provided with valid input parameters.
8. 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:

Your program will be marked out of 30 (later scaled to be out of 15% of the final mark).
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 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:

i) Median

Mathematically, median is represented as:

$$Median(X) = \begin{cases} X\left[\dfrac{n+1}{2}\right] & if\ n\ is\ odd \\ \dfrac{X\left[\frac{n}{2}\right] + X\left[\frac{n}{2}+1\right]}{2} & if\ n\ is\ even \end{cases}$$

X= ordered list of values in the data set.

n= number of values in the data set.

ii) Mean absolute Deviation

$$\text{MD} = \frac{1}{n}\sum_{i=1}^{n}|x_i - m(X)|$$

$m(X)$ = average value of X

n = number of data values

x_i = data values in X

iii) Standard deviation:

Mathematically, standard deviation is represented as:

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

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.

iv) Correlation coefficient:

Mathematical formula to calculate correlation is as follows:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \Sigma(y_i - \bar{y})^2}}$$

where x_i and y_i are the values of sales in each year (mentioned in the sales.txt file) for the product with the highest and the lowest discounted price respectively. $\bar{x}$ is the mean of sales of product with the highest discounted price and $\bar{y}$ is the mean of the sales of the product with the lowest discounted price.

Note: Any updates regarding the project will be posted on Moodle help forum.