Project 1:

Submission deadlines: ~~8:00am, Monday 27 April 2020~~
5:00pm, Friday 1st May 2020

Value: **15%** of CITS1401.

To be done individually.

You should construct a Python 3 program containing your solution to the following problem and submit your program electronically on LMS. No other method of submission is allowed.

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 submission 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.

Overview

UWA, like every university around the country needs to deal with marks of the students in different courses. For the analysis of university's performance, management needs to have a computer program which can read the data from a csv (comma separated values) file and return different statistical aspects of the marks of the students and their ranking. Your task is to write a program which follow the following specifications.

Specification: What your program will need to do

Input:

Your program must define the function **`main`** with the following signature:

```
def main(csvfile):
```

The input argument is the name of the csv file containing information and marks of the students which needs to be analysed. The first row of the csv file will contain the headers. From the second row, the first and second values of each row contain the names and any other information of the students. From third value onwards, it contains the marks student obtained in each subject. We do not have prior knowledge about the number of students and subjects available in the csv file.

Output:

The function is required to return the following outputs in the order provided below:

- List containing the **minimum** marks for each subject followed by minimum total marks obtained by a student. The order of the subjects should be same as the order of the subjects in the header row.
- List similar to above containing the **maximum** marks.
- List similar to above containing the **average** marks.
- List similar to above containing the **standard deviations** in marks.
- List similar to above containing the **correlation** of the ranks of the subject marks and total marks. The highest mark will be ranked as '1', the second highest mark will be ranked '2' and so on. If two or more students get the same marks then they should be ranked according to their names in alphabetical order. Two students cannot have the same rank for a course or total marks which means two student cannot be with the same name and marks. Use Spearman's rank correlation coefficient for calculating the correlation of rankings.

All returned lists must contain numerical values rounded to four decimal places. Remember not to round the values during calculations and round them only at the time of saving them in the output lists.

Example:

Download the sample_student_marks.csv file from the folder of Project 1 on LMS. An example interaction is:

```
>>> mn,mx,avg,std,cor = main('sample_student_marks.csv')
```

The output returned in the variables are:

```
>>> mn

[0.0, 6.0, 3.0, 5.0, 1.0, 7.0, 9.0, 2.0, 262.0]

>>> mx

[98.0, 96.0, 92.0, 92.0, 96.0, 97.0, 94.0, 100.0, 587.0]

>>> avg

[49.7667, 53.2333, 61.7333, 54.6333, 57.2333, 55.0, 50.5, 43.0, 425.1]

>>> std

[31.3706,  24.5678,  25.737,  26.1731,  31.6014,  27.3496,  23.8841,
30.7864, 83.7661]

>>> cor

[0.2654, 0.2908, 0.3286, 0.3428, 0.4051, 0.2863, 0.4309, 0.495, 1.0]
```

Assumptions:

Your program can assume a number of things:

- Anything that is meant to be a string (i.e. a name) will be a string, and anything that is meant to be a number (i.e. a score or mark) will be a number.
- The order of columns in each row will follow the order of the headings provided in the first row, though number of columns and rows are not constant across different csv files.
- No data will be missing in the csv file.
- The **`main()`** function will always be provided with valid file name.

Important grading instruction:

You will have noticed that you have not been asked to write specific functions. That has been left to you. However, it is important that your program defines the top-level function `main()`. The idea is that within `main()`, the program calls the other functions. (Of course, these may call further functions.) The reason this is important is that when I test your program, my testing program will call your `main()` function. So, if you fail to define `main()`, my program will not be able to test your program and your submission may be graded zero.

Things to avoid:

There are a couple things for your program to avoid.

- You are not allowed to import any Python module. While use of the many of these modules, e.g. csv or scipy is a perfectly sensible thing to do in a 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.
- Do not assume that the input file names will end in .csv. File name suffixes such as .csv and .txt are not mandatory in systems other than Microsoft Windows.
- Ensure your program does NOT call the `input()` or `print()` functions at any time. That will cause your program to hang, waiting for input that my automated testing system will not provide. In fact, what will happen is that the marking program detects the call(s), and will not test your code at all which may result in zero grade.

Submission:

Submit a single Python (.py) file containing all of your functions via LMS before **8:00 am 27 April 2020** on LMS

You need to contact unit coordinator if you have special considerations or making late submission.

Marking Rubric:

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

22 out of 30 marks will be awarded 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 error states such as different number of columns and rows in the input file. You need to think creatively what your program may face.

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

Style Rubric:

0	Gibberish, impossible to understand
1-2	Style is really poor or 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 try to minimise the number of times your program looks at the same data items. You may think to use different data structure such as tuples, lists, or dictionaries.

Efficiency Rubric:

0	Code too incomplete 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 too hard to fix, the marker needs to move on to other submissions.