CITS1401 Project 1 grading guide, Semester 2, 2023

There are 22 marks for functionality which are divided into 4 outputs test cases and a few error states. Each test case checks each output with several files. Marks for each test case are mentioned below.

Please provide feedback explaining the reasons for the student's loss of marks. Feel free to include any additional comments you find relevant. You can also include a note in the comments section using a format like x/22 (test) + x/5 (style) + z/3 (efficiency) to further specify the breakdown of marks. **Please note that** you can't leave the comment section empty in Moodle. You can add their awarded marks for each criterion (x/22 (test) + x/5 (style) + z/3 (efficiency)) in the comment section if they get full marks.

If the output is not returned in a proper format as required and simply printed, then deduct 4 extra marks as mentioned at the end and grade it accordingly. This is considered as marker's intervention to fix student's code.

Following table shows the marking criteria for each calculation they complete in Python:

Maximum awarded marks	**Criteria**
6	Correlation calculation.
4	Standard deviation calculation.
4	Ratio Calculation
4	Maximum and minimum calculation.
4	Error state
8	Style and efficiency
Total: 30	

Output test cases:

1. Correlation (6 Marks):

If the outputs of both test files match then assign 6 marks, else
If the output of only one test file fully matches then assign 3 marks, else
If the output didn't match at all but match for the sample case provided in the lab sheet as a test case in Moodle, then assign 1 mark.

2. Standard deviation (4 Marks):

If the outputs of both test files match then assign 4 marks, else
If the output of only one test file fully matches then assign 2 marks, else
If one of the list values of the output for a test file is correct, then assign 0.5 mark/test file.

3. Maximum and Minimum (4 Marks):

If the outputs of both test files match then assign 4 marks, else
If the output of only one test file fully matches then assign 2 marks, else
If one of the list values of the output for a test file is correct, then assign 0.5 mark/test file.

4. Ratio (4 Marks):

If the outputs of both test files match then assign 4 marks, else
If the output of only one test file fully matches then assign 2 marks, else

5. Error State (4 Marks):

   For successfully passing each error state assign 1 mark. We are checking following error states:

   i) Zero division
   ii) Single row
   iii) All data zero for a specified country
   iv) Case sensitivity

   For cases i, ii, and iii, avoiding triggering a traceback and returning the outputs in the correct format are sufficient to allocate the mark.

Note:

a) Please deduct 2 marks if the values are not rounded to 4 decimal places in any required calculation.
b) If student has imported any module, then comment it out and run the code, grade it accordingly and assign 0/3 in efficiency.

<u>Style and efficiency: (be lenient but mention in comments to improve for next time)</u>

- Style (5/8) which involves intuitive variable and function names, consistent indentation, comments, etc.
  - Default is 5.
  - Deduct 1 mark if person's name or student id is not on the file to identify author of the code.
  - Deduct 2 marks if scant comments are provided.
  - Deduct 2 marks if no or one function other than the main is created.
  - Minimum can be zero.

- Efficiency (3/8):
  - Default is 3 marks.
  - Deduct 2 mark if readline() function is used in a loop or file is opened multiple times.
  - Deduct 1 marks if the code has more loops than necessary.
  - Deduct 1 mark if the code is taking long time to compile the big testing file.

For any intervention to fix the code, deduct 4 marks extra. However, it should be quick and simple. Normally missing a colon or wrong indentation or missing a character of some instruction is considered as a simple debug. Re-writing or adding or removing the code is not part of the grading and code should be graded accordingly as mentioned above.

Sample Outcomes:

1. Testing general datasets:

```
>>> MaxMin, stdv, ratio, corr = main('Organisations.csv',
'Belgium')
>>> MaxMin
['weaver-barnett', 'bolton diaz and kent']
>>> stdv
[936.9468, 875.2285]
>>> ratio
```

```
2.2186
>>> corr
-0.1013
```

2. Testing random rows:

```
>>> MaxMin, stdv, ratio, corr = main('Organisations_shuffled.csv',
'Belgium')

>>> MaxMin

['weaver-barnett', 'bolton diaz and kent']

>>> stdv

[936.9468, 875.2285]

>>> ratio

2.2186

>>> corr

-0.1013
```

3. Testing single row, zero data, and zero division:

```
>>> MaxMin, stdv, ratio, corr = main('Organisations_errors.csv',
'Belgium')

>>> MaxMin

[]

>>> stdv

[]

>>> ratio

0

>>> corr

0
```