Assessment 2 - Program implementation

Aim

The goal of this assignment is to design and implement a program for basic analysis in Python. You will need to use the Python skills covered in weeks 3 and 4, as well as the problem solving skills from weeks 1 and 2. Parts of the functionality for this program you will have designed in Assessment 1, but additional functionality is also required.

Task

You are required to develop a basic Python program as detailed in the instructions below. 80% of your marks will be based on the completeness and correctness of the basic program, and 20% based on the functionality in the challenge-version of the program.

Important note

For pedagogical reasons, we require you to implement the program using **only** basic Python constructs and the `csv` module. Do **not** import any modules aside from `csv`. In Assessment 3, you will be free (and expected) to use additional libraries to calculate statistics, etc.

Detailed instructions

We have been asked to create a program that will allow users to load sets of data from CSV files and then view, manipulate and perform simple statistical analysis on the data.

When the program is loaded the user will see an introductory message (you are welcome to determine this as you see you fit, but make sure to include your name). For example:

```
Welcome to The Smart Statistician!
Programmed by Ada Lovelace
```

After the welcome message, the user will be presented with the following menu:

```
Please choose from the following options:
    1 – Load data from a file
    2 – Display the data to the screen
    3 – Rename a set
    4 – Sort a set
    5 – Analyse a set
    6 - Quit
```

Option 6 will exit the program, every other option will do some task and then display the menu again until the user chooses 6 at this menu.

If the user enters anything other than a value between 1 and 6, display an appropriate error message (e.g. `Invalid selection!`), then get the user to enter another choice.

Option 1 - load data from a file

When the user chooses option 1, they will be asked for a filename, which is expected to be in the same directory as the program (no need for path information). Your program should use the exact filename as stated, **do not append .csv or any other extension**.

Your program should be able to handle data in basic CSV format, for example:

```
Rainfall,35,23,12,65,34,111,54,23,68,97
Age,35,23,14,76
Odometer Reading,35065,67443,23545,12323,72335
```

Your program should treat the first value in each line of the file as the name of a dataset, and the subsequent values as the data in the set. Your program needs to deal only with integer data (you may choose to allow floating-point numbers, but we won't test for this case, and it's fine for your program to treat them as unexpected input).

Your program should store the data either as a list of lists or a dictionary of lists. Each dataset may have any length >= 1.

There are three problems your program may encounter here.

1. the file does not exist
2. the file is in an unexpected format
3. the file contains a dataset with no data

In all of these cases your program should display an appropriate error message (e.g. `File not found`, `Data is in an unexpected format`) then return control to the main menu. Hint: you do not have to write any code to analyse whether the file is in the correct format, instead, make good use of exceptions to gracefully handle any conversion errors that your program encounters.

If the user has previously loaded data from a file in the same session, none of the previous data should appear in the program after a new load. E.g. if the user loaded datasets W, X, Y and Z from `A.csv`, then subsequently loads `B.csv`, none of the datasets from `A.csv` should remain in the system.

Option 2 - display the data to the screen

Choosing this menu option will display each set of data, one after the other, with each set being displayed on a single line as shown below:

```
Rainfall
--------
35, 23, 12, 65, 34, 111, 54, 23, 68, 97

Age
---
35, 23, 14, 76

Odometer Reading
----------------
35065, 67443, 23545, 12323, 72335
```

For full marks, you'll need to match the above formatting perfectly. Consider how you can build a string of values from a list. You can do this manually, or look into online Python documentation for an appropriate built-in string method.

If no data has been loaded yet, display an appropriate message (e.g. `No datasets to display`).

Option 3 - rename a set

When the user wants to rename a set we first have to find out which set. To do this we will display the sets in order with a number, and we ask the user to enter a number, as below:

```
Which set do you want to rename?
    1 – Rainfall
    2 – Age
    3 – Odometer Reading
```

The user will then need to enter the appropriate number (in this case between 1 and 3). The range will change depending on the number of datasets in the system.

If the user enters an invalid number, then display an error message (e.g. `Invalid selection`) and ask again until they enter a valid number.

Once they have entered a valid number we then ask for the new name for that set. There are two requirements for the new name:

1. It cannot be blank
2. It cannot already be in the system.

You need to check both of these (in Python, the empty string is "", and you can always compare the new name to the names already in the system). If the name is invalid then display an error message (e.g. `Name cannot be blank`, `Name must be unique`), and ask for a new name until a valid name is entered.

Once we have a valid name we will replace the first element of the appropriate list with the new name, and then display a message to the user:

```
Rainfall renamed to Monthly Rainfall.
```

Option 4 - sort a set

As per option 3, the program must first provide the user with a menu to choose the set they want to sort:

```
Which set do you want to sort?
    1 – Rainfall
    2 – Age
    3 – Odometer Reading
```

Once we have a valid choice of set, we need to sort its data (in **ascending** order). If you're not sure how to sort a list, revisit the material on lists in week 4.

Option 5 - analyse a set

Again, the program should first determine which set the user wants to analyse:

```
Which set do you want to analyse?
    1 – Rainfall
    2 – Age
    3 – Odometer Reading
```

The program will then produce a statistical report for the dataset, with the format below:

```
Rainfall
--------
number of values (n): 10
             minimum: 12
             maximum: 111
                mean: 52.20
              median: 44.50
                mode: 23
  standard deviation: 31.35
```

Again, for full marks you must match the format above perfectly. Notice the alignment of the labels and the values, and two-decimal places for statistics that might not be whole numbers.

We encourage and expect you to use the built-in Python functions `len()`, `min()`, and `max()` as appropriate, however you will need to write your own logic to determine mean, median, mode, and standard deviation. Note: for your report, calculate the population standard deviation, not the sample standard deviation.

How to handle mode for this assignment:

- if there is one mode, report that (e.g. `mode: 23`)
- if all values appear only once, report `mode: none`
- if there are multiple values for the mode (with frequency greater than 1), report each modal value separated by a comma, in ascending order (e.g. mode: 19, 23, 72)

Challenge section

Once you have completed the basic program above, we have some additional functionality to add if you want to achieve full marks. You should only attempt this once you've completed (and polished!) the basic version of the program, and should create a separate file (LastnameFirstname_A2_challenge.py) for this version. We will mark most criteria for your assignment on your basic version, and only the **Challenge Functionality** section on your challenge version.

While we will provide advice, hints, and tips on the basic functionality for this assignment, for the challenge section you'll be on your own - it is a *challenge* after all :)

See the additional challenge PDF for further instructions.

Getting started

As a starting point we suggest you plan then implement your main function, which will display the welcome and menu. You will have planned a similar main function in the first assignment. For each of the menu options (other than quitting) just display a message.

Then create separate functions and handle the menu options, and move the dummy messages there.

Next, create your list of lists (or dictionary of lists) for storing the datasets (with dummy data for now), and pass it to each function, just to test that the data flows.

Now that you have a basic idea of how your program should work at a high level, begin planning the details. What helper functions will you need? How will you calculate the statistics?

Hints and tips

You will be marked not only on whether your code works, but also the quality of your code.

- Your variables and functions should be well named.
- Loops and selection structures should not be unnecessarily complicated, and conditions should be written to make your code's intention clear - explicit conditions should be used rather than

`while True`, for example.

- Don't repeat yourself. Make good use of functions when you find you need to do the same job multiple times.
- Your program should not crash under any foreseeable circumstances (i.e. inappropriate input from the user or unexpected data in the file). Make good use of exception handling to deal with errors. Exception handling will be especially handy when `try`ing to load from the file.

You may find it easier to develop and test a large-ish program like this using an IDE such as Pycharm, rather than a Jupyter Notebook. We will provide a basic overview of IDEs in a tutorial session.

Submission instructions

You should name your basic program as LastnameFirstname_A2.py, and your challenge program as LastnameFirstname_A2_challenge.py. Attach both files to your submission.