Project 1:

Submission deadlines: 8:00 am, Tuesday 26th April 2022
Value: **15%** of CITS1401.

To be completed individually.

You should construct a Python 3 program containing your solution to the following 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. Your program will be automatically run on Moodle for sample test cases provided in the project sheet if you click the "check" link. However, your submission will be tested thoroughly 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 are expected to have read and understood the University's guidelines on academic conduct. Following 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 learned 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 (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.

Overview

Researchers at UWA analyse 3D facial features to find their relationship with certain syndromes, like Autism. Recently, they published a research paper which looked at the facial asymmetry of parents of Autistic Children and compared it with normal adult population. You can read the paper here.

The researchers now want to analyse the 3D facial asymmetry of upper and lower parts of the faces of adults and need your help. They have shortlisted five landmarks on the upper face and four on the lower face. They use one landmark (i.e. the nose tip) for calibration. Table 1 provides the details of each point and Figure 1 shows their location on the face.

In this project, you are required to write a computer program that can read the data from a CSV (comma separated values) file provided to you. The file contains the facial asymmetry in X, Y and Z axes for the 10 facial landmarks mentioned in Table 1 for each adult. (Remember, these points do not provide the location of a point in 3D space). For simplicity, the landmark numbers are provided in the CSV file instead of their names. Your task is to write a program which fulfills the following requirements.

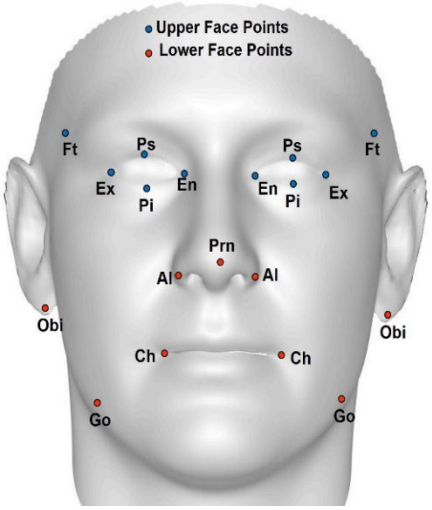

Figure 1: Facial landmarks' locations on the face.

Table 1: Facial landmarks' identification numbers and their details.

Point Number	**Landmark Name**	**Location**	**Upper/ Lower face**
1	Ex	Outer eye corners	Upper Face
2	En	Inner eye corners	Upper Face
3	Ps	Centre of upper eyelids	Upper Face
4	Pi	Centre of lower eyelids	Upper Face
5	Ft	Temple points	Upper Face
6	Al	Outer edge of nostrils	Lower Face
7	Ch	Outer mouth corners	Lower Face
8	Go	Joint between the two jaws	Lower Face
9	Obi	Bottom point of ear lobes	Lower Face
10	Prn	Nose Tip	For calibration

Specification: What your program is required to do

Input:

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

```
def main(csvfile, adults, type):
```

The input arguments to this function are:

- `csvfile`: The name of the CSV file containing the record of the facial asymmetry which needs to be analysed. Below are the first two rows of a sample file.

Adult ID	Point Number	X	Y	Z
A0001	1	0.78	0.23	-1.2

  The first row of the CSV file contains the following headers:

  - Adult ID: The de-identified ID of an adult.
  - Point Number: The facial landmark's reference number as mentioned in Table 1.
  - "X", "Y" and "Z": The facial asymmetries in X, Y and Z axes respectively.

  We do not have prior knowledge about the number of adults we have to analyse (i.e. the number of rows) that the CSV file contains. Adult ID is a string while the remaining values are numeric.

- `adults`: The adult ID of an adult or a list containing two adult IDs, which need to be analysed depending on the third input. It can contain a single adult ID, or a pair of adult IDs given as a list. Remember that the ID is a string and is case insensitive.

- `type`: The input argument which mentions what analysis are required. It can take only one of the two string inputs: "stats" or "corr". If the third input argument is "stats", then the objective of the program is to carry out some statistical analysis of the adult whose ID is given in the second argument. Otherwise, if the third input argument is "corr" then the objective of the program is to do some correlation analysis between asymmetries of the pair of adults whose IDs are given in the second argument.

Pre-processing:

The program requires to apply the following pre-processing step on the data in CSV file before performing any analysis.

- The facial asymmetry considers the nose-tip as a calibration point. Hence, the asymmetry, for a given face, at the nose-tip (Point Number 10) must always be zero. If it is non-zero, then facial asymmetry values of nose-tip (Point Number 10) in each dimension must be subtracted from the respective dimensions of the remaining 9 landmarks asymmetry values to calibrate them.

Output:

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

- When the third input argument is *"stats"*, then: (in this case the second input argument will have a string containing an adult ID)
    1. A list containing the 3D asymmetry values of each of the 9 facial landmarks. The formula for finding the 3D asymmetry value is provided at the end of the sheet.
    2. A list containing the minimum (non-zero) 3D asymmetry of the upper and lower face.
    3. A list containing the maximum (non-zero) 3D asymmetry of the upper and lower face.
    4. A list containing the average 3D asymmetry of the upper and lower face.
    5. A list containing the standard deviation of the 3D asymmetry of the upper and lower face.

- When the third input argument is "corr": (in this case the second input argument will have a list containing two adult IDs)
    1. A single value that is the correlation of the 3D asymmetry values of all 9 landmarks between the pair of adults given in the input argument "adults".

All returned outputs should follow the below mentioned specifications.

- For the output list of 3D asymmetry of all landmarks, the list should have values for Point-1 through to Point-9. For all the remaining output lists, the values should be for the value of "upper face" followed by the "lower face".
- 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 and round them only at the time that you save them into the final output variables.

Examples:

Download `asymmetry_sample.csv` file from the folder of Project 1 on LMS or Moodle. Some examples of how you can call your program from the Python shell (and examine the results it returns) are:

```
>>> asym3D1,mn1,mx1,avg1,std1=main('asymmetry_sample.csv','C4996','stats')
```

The output variables returned are:

```
>>> asym3D1
[3.1384, 3.8952, 4.0838, 3.2533, 1.295, 3.1759, 2.7899, 3.3195, 2.7004]

>>> mn1
[1.295, 2.7004]

>>> mx1
[4.0838, 3.3195]

>>> avg1
[3.1331, 2.9964]

>>> std1
[0.9877, 0.2583]


>>> asym3D2,mn2,mx2,avg2,std2=main('asymmetry_sample.csv','G8328','stats')
```

The output variables returned are:

```
>>> asym3D2
[1.0436, 1.3438, 1.2836, 1.2887, 1.2887, 1.3336, 1.3501, 1.2513, 1.252]


>>> mn2
[1.0436, 1.2513]
>>> mx2
[1.3438, 1.3501]
>>> avg2
[1.2497, 1.2967]
>>> std2
[0.1054, 0.0455]
>>> corr=main('asymmetry_sample.csv',['G8328','C4996'],'corr')
```

The output variable returned is:

```
>>> corr
0.0262
```

Assumptions:

Your program can assume the following:

- Anything that is meant to be a string (i.e. header row) will be a string, and anything that is meant to be a numeric will be numeric.
- The order of columns in each row will follow the order of the headings provided in the first row. However, the rows may be in a random order (i.e. they will not necessarily be asymmetries from Point-1 through to Point-9 in order) in CSV files.
- No data will be missing in the csv file; however asymmetry values can be zero and must be accounted for when calculating averages and standard deviation.
- The `main()` function will always be provided with valid input parameters.
- The formula for calculating 3D asymmetry, standard deviation and correlation are provided at the end of the project sheet.

Important grading instruction:

Note that you have not been asked to write specific functions. This task has been left to you. However, it is essential that your program defines the top-level function *main(csvfile, adult, type)* (hereafter referred to as "main()" in the project sheet 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 the project sheet.

Things to avoid:

There are a few things for your program to avoid.

- **You are not allowed to import any Python module.** 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.
- 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. Do not enforce that within your program that the file must end with a .csv or any other extension (or try to add an extension onto the provided csvfile argument).
- 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).
- Your program should also not call the `print()` function at any time except for the case of graceful termination. If your program has encountered an error state and is exiting gracefully then your program needs to return zero as the value for the correlation otherwise empty lists and print an appropriate message.

Disclaimer: Although this project addresses a real-world problem, the files provided to you contain synthetically generated data.

Submission:

Submit your solution on Moodle before the deadline as per details provided at the start of the project sheet.

You need to contact unit coordinator if you have special considerations or making a submission after the mentioned due date.

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, but not limited to, different numbers of rows in the input file, missing asymmetry record 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 look into 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.

8 out of 30 marks will be awarded on *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 watch the lectures 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 you need to minimise the number of times your program looks at the same data items.

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 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;
- is 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.

Formulas:

3D Asymmetry:

$$Asym3D = \sqrt{(X^2 + Y^2 + Z^2)}$$

where X, Y and Z are the asymmetry values of each point in the 3D plane.

Standard deviation:

It is mathematically expressed as:

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

σ = population standard deviation

N = the size of the population

x_i = each value from the population

μ = the population mean

You can find more details at https://en.wikipedia.org/wiki/Standard_deviation

Correlation:

The correlation *rxy* for paired data *{(x1,y1),…(xn,yn)}* consisting of *n* pairs, is mathematically expressed as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad \text{(Eq.3)}$$

where:

n is sample size

x_i, y_i are the individual sample points indexed with *i*

$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (the sample mean); and analogously for $\bar{y}$