Major Assignment 2

ENGINEER 1D04
Dr. William Farmer and Dr. Spencer Smith
McMaster University, Winter 2012

Revised: June 19, 2013

Please use AutoMarker (`automarker.mcmaster.ca`) and Avenue to acquire, test, package and submit your assignment. The procedure for submitting assignments is summarized on Avenue, with additional details provided by AutoMarker. **Please frequently back up your work by creating a submission package in AutoMarker**. This will provide a chance to recover your work in the event of an equipment failure.

Background

This assignment is motivated by scientific and engineering applications that analyze experimental data. The purpose of this assignment is to write a Python program that reads two sequences of numbers stored in two separate files, represents the sequences of numbers as two lists of floats, performs a calculation on the members of the lists, and then outputs the result.
Design, implement, and test a program that satisfies the requirements below.

****Important**:** This assignment will be run through an automated testing program to be graded. The program will make the assumption that any output containing an equals sign (=) is an answer to be marked. **Do not** print output containing the equals sign, except where specified in the requirements below. Outputs must also be printed on separate lines. Additionally, input and output statements must be **exactly** in the order specified. Failure to precisely follow the requirements below will result in a significant loss of marks.

1. The program asks the user to enter the name of a text file F_x that stores a (finite) sequence $x_0, x_1, \ldots, x_{n-1}$ of numbers. It is assumed that the ith word in the text file is a Python literal of type `int`, `long`, or `float` that represents the x_i member of the sequence.

2. The program reads the sequence of numbers in F_x and constructs a list L_x of floats that represents the sequence.

3. The program repeats the previous two requirements for a second text file F_y that stores the finite sequence $y_0, y_1, \ldots, y_{n-1}$ of numbers and constructs a list L_y of floats.

4. The program computes the product moment correlation coefficient r, which will lie between -1 and 1. The equation for determining r for the n data points in the input file is as follows:

$r = \frac{a}{\sqrt{b}\sqrt{c}}$, where
$a = \sum_{i=0}^{n-1}(x_i - \bar{x})(y_i - \bar{y})$, $b = \sum_{i=0}^{n-1}(x_i - \bar{x})^2$ and $c = \sum_{i=0}^{n-1}(y_i - \bar{y})^2$
with $\bar{x} = \frac{1}{n}\sum_{i=0}^{n-1} x_i$ and $\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1} y_i$.
As an example, one of the summations expands as follows:

$$\sum_{i=0}^{n-1}(x_i-\bar{x})(y_i-\bar{y}) = (x_0-\bar{x})(y_0-\bar{y})+(x_1-\bar{x})(y_1-\bar{y})+\cdots+(x_{n-1}-\bar{x})(y_{n-1}-\bar{y})$$

5. The program prints the value of L_x, L_y and r in the format "*label* = `value`," where *label* is an appropriate description of `value` and "=" is the equals sign.

6. The program is written in Python in a module, NOT in the Python Shell. To create a new module in IDLE, go to File → New Window. You must save this file with a .py extension. For more information on submitting your program, click the "AutoMarker Instructions" button above.

7. Your name, MacID, student number, and the date are given in comments at the top of your Python (.py) file before your program.

8. Your answers to the design questions and the test plan (see below) are given in comments at the bottom of your Python (.py) file after your program.

9. Your program MUST have valid Python syntax and it must run without errors. Ensure that your program runs properly by running it before you submit.

10. You must sign out with a TA or IAI after you have submitted your lab at the submission station. Failure to do so could result in a zero.

Design and Implementation Instructions

You may assume without penalty that there is exactly one number per line in the files F_x and F_y and that both files are the same length.

Design Question

Can you solve for r using one loop? Please justify your answer.

Test Plan

Produce a test plan for each test case `i` in the following form:

`Test:` `i`
`Input:` `[`L_x, L_y`]`
`Expected Output:` r

You must have no less than 3 test cases. Have at least 1 case where your program does not fail. For the other cases, we encourage you to seek out test cases where your program would fail. That is, where the expected output is a failure.