**Assignment 1 – Basic Calculations**

**Dean Zeller 20 points**

### CG120 Due: Friday, August 31st

**Fall 2018**

***Objective***The student will calculate the sum and average on integer input, outputting the results.

***Background***: ***Overall Program Design***

This assignment sets the stage for all future assignments. There will be many assignments feature on data processing. No matter how complex the processing will get, programs dealing with data analysis will generally follow three phases.

Phase 1: Gather input

Phase 2: Perform analysis

Phase 3: Output results

For this assignment, all three phases are quite simple. User input will be exactly three integers, the analysis will consist of just a *sum* and *average*, and the output does not require formatting. All three of these aspects will be covered in detail throughout the semester. For now, simply observe the overall program structure.

***Statements***

You will program a series of Python statements in a program to produce the test run below. The output should look exactly like the test run above. To do this you will only need the following statements:

Documentation # This is documentation, ignored by IDLE

Printing print("Hello World!!!")

Input (String) name = input("Please enter you name =>")

Input (Integer) number = int(input("Please enter a number =>"))

Perform Calculation total = number1 + number2

***Output Notes:***

1. Most lines above are generated by print statements.
2. Any line with **boldface** **text** indicates an input statement, with the user input shown as **bold**. Use a prompt of **=>** any time input is requested. Without this consistent prompt, it is unclear whether the program is busy processing or simply waiting for user input.
3. Observe that there is detailed user instructions during execution. Follow this concept of giving instruction and feedback for all course programming. It is entirely possible someone else will be running your program as the user, so always give clear instructions and feedback on progress.
4. Indentation and blank lines in the output are important for readability. Implement all indents and blank lines in both print and input statements. When complete, the overall output should look exactly like the test-run.

***Test Run:***

The following is a test run of the program you are going to create for this assignment. Any text in **boldface** is data entered by the user.

Welcome to Assignment 1, Basic Calculations

Please enter your name => **Dean**

Hello, Dean

Introduction:

This program will calculate basic statistics on three integer numbers. It will run in three phases:

Phase 1 – Gather input from user

Phase 2 – Calculate sum and average

Phase 3 – Output results

Phase 1: Gather user input

--------------------------

Please enter three numbers. Only enter integer values, as error-checking

has not yet been implemented.

First number => **12**

Second number => **2**

Third number => **8**

Numbers entered: 12 2 8

Phase 1 complete

Phase 2: Perform calculations

--------------------------------

Calculated sum

Calculated average

Phase 2 complete

Phase 3: Output table

---------------------

Data: 12 2 8

Sum: 22

Average: 7.333333333333333

Phase 3 complete

Exiting program

***Instructions***

1. Open the IDLE shell window.
2. Create new program file named assignment1.py
3. Enter the appropriate file documentation to identify you as the primary programmer. See below for documentation templates.
4. Save and execute your file. At this point, it should actually do nothing as it is only documentation and not code. All successive steps involve code to implement lines in the test run. After each step, test and execute your code, fixing any errors before moving to the next step.
5. The first two output lines are simple print statements. Print a single string surrounded by “quotation” marks. You can include multiple strings, separate by a comma (,) or a plus sign (+).
6. The third output line is requesting string input from the user using the input statement discussed above. Write the code for the input statement, using the appropriate prompt, as shown in the test output.
7. Following the name input statement, the remainder of the introduction are again simple print statements. Complete the introduction section, and test your code.
8. Complete the print statements for the first four lines of Phase 1.
9. Within phase 1, the program accepts integer input from the user. Create three integer variables and accept user input for their initial values. You do not need to perform error-checking yet, though it will be introduced later in the course. See the input statement above on how to input integers.
10. Print the numbers after the third number is entered. This is simply to confirm the numbers were entered and stored correctly. Multiple items may be printed in a single print statement, separate by commas.
11. Phase 2 contains the code to the calculate the sum and average. Create the variables sum and average and use common mathematical operations (+ - \* /) to calculate them appropriately. See the reference on calculations above. After each is calculated, print a message indicating it is complete.
12. Phase 3 contains the code to output the data and results. Use print statements to print the results, exactly as shown in the test run.
13. Make sure there are appropriate phase separators as documentation. Each phase should begin with a separator block. See below for templates you can use.
14. Upon completion, your program should execute exactly like the program test run. Test your program several times with different integer values, to make sure your program works correctly.
15. Use screencastomatic.com to record a two-minute screen capture video of your program demo, with the following requirements.
    1. Introduce yourself, and greet the teachers, TA, and students.
    2. Scroll through your code, making note of the assignment requirements and how you solved them.
    3. Run the test data used on the assignment write-up.
    4. Run the program two more times with other test data, and ensure it works correctly.
    5. Thank your audience and indicate that the video is complete. You may have some sort of fun ending to entertain the viewer.
16. Complete the self-evaluation of your assignment. This will be the same rubric used to evaluate your work. The self-evaluation may be turned into the instructor the following week.
17. Create a zip file with your program file and screen capture video. Make sure all necessary files are in the submission, and no unnecessary files. Submit the zip file to the appropriate dropbox on canvas by the due date.

***File Documentation***

Include this section of documentation at the top of your Python program. It is documentation that contains information about the file, but no programming itself. Make changes to the appropriate fields, denoted with dashes-in-the-name; this will reflect that this is your code.

################################################################################

# title-of-assignment #

# #

# PROGRAMMER: your-name #

# CLASS: CS102 #

# ASSIGNMENT: Assignment assig-# #

# INSTRUCTOR: Dean Zeller #

# TA: Robert Carver #

# SUBMISSION DATE: date-of-submission #

# #

# DESCRIPTION: #

# Describe-the-program-implemented-in-your-own-words #

# #

# COPYRIGHT: #

# This program is (c) 2018 your-name and Dean Zeller. This is original work, #

# without use of outside sources. #

################################################################################

***Phase Separators***

Between each phase, include an outlined block with the phase number. Also include one for the introduction and ending message. Use the following as a template.

################################################################################

# Phase 1 -- Gather input from user #

################################################################################

***Grading***

You will be graded on the following criteria:

*Effort* Creating the code to calculate the sum and average.

*Code Formatting* Entering correctly formatted file documentation and phase separators

*Output Readability* Organization and flow of the output, follows test-run perfectly

Extra credit will be given for the following:

* Implement pauses to display text slowly, after the introduction and between phases.
* Extra efforts in the program output, including detailed instructions and text-formatting. Complete the required formatting before adding your own. It should look presentable and organized, on top of the existing formatting.
* Calculating more than just sum and average. (Geometric mean, standard deviation, variance, etc…)

***Evaluation Instructions (Summary)***

On Monday after the due date, the solution and all student submissions will be made available for download. Find three students in class to evaluate your work, and you are to evaluate their work. You will be given the rubrics similar to the following page to evaluate their work.

***Assignment 1 Grading Rubric Programmer Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***CG120 -------------SELF EVALUATION-------------***

Program Testing

\_\_\_\_\_\_ Execution

Does the program execute correctly?

-20 for inexecutable code

\_\_\_\_\_\_ Output

Does the video correctly demonstrate the execution of the program?

-20 for altered or fake output

\_\_\_\_\_\_ Test input/output

Does the program work correctly for the given test input from the assignment write-up?

-2 to -10 for errors in output

\_\_\_\_\_\_ Other input/output

Test the code for other combinations of data, does it work correctly?

-2 to -10 for insufficient testing

\_\_\_\_\_\_ Is the output formatted exactly like the test run?

-2 to -5 for incorrect formatting

-2 to -10 for elements missing

Submission Requirements

\_\_\_\_\_\_ Program

Does the submission have all required files to execute?

-10 for missing any necessary program file(s)

\_\_\_\_\_\_ Video

Is there a video demonstration included in the submission?

-10 for missing video

Does the video include the five requirements (intro, code, given test, other test, thanks)

-2 to -9 for insufficient video or missing elements

\_\_\_\_\_\_ Program Style

Does the program follow the specified style guidelines including indentation, variable names, and blank spaces?

-2 to -5 for insufficient attention to program style

Documentation Requirements

\_\_\_\_\_\_ File documentation

Is there correct documentation at the top of the file following the documentation guidelines in the writeup.

-10 for no documentation

-2 to -9 for insufficient or incorrect documentation

\_\_\_\_\_\_ Spelling/grammar

Is the documentation properly written and free of spelling and grammar errors?

-1 to -5 for spelling and grammar errors