CSCI 1411: Fundamentals of Computing
Lab 11
Due Date: November 10, 2023

Name: Brandon Perez

Goals:

- Learn modular programming.
- Learn test driven development methodology.

Development Environment: IDLE

Deliverables:

1. This completed document with required screen shots.
2. Python file created for the lab. Name the file using the following format: `lastnameLab11.py`.

How to take a **screen shot**:

- For a Windows 10: Use Snipping Tool to copy and press CTRL + V to paste screen shot.
- For Mac: Press Shift + Command (⌘) + 4 to copy and press Command (⌘) + V to paste screen shot.

In this lab we will write a program which will perform the following tasks:

1. Ask user for a set of integers.
2. Calculate sum, mean and standard deviation of the given set of integers.
3. Display the result.

The modular programming and test driven development methodology works as follows:

1. Write some code.
2. Test the code.
3. Write some more code and test the new code.
4. Keep on doing step 3 until you are done with your program.

We will complete the development process in 6 steps. In steps 1 to 5 we will write functions. Each function will perform one single task. We will also test our functions as we write code for each one of them. In step 6 we will write `main` function which will use the functions developed in steps 1 to 5 to accomplish steps stated above for our program.

Note: Each of the function must be documents with a block of comments. An example block is provided for the function developed in Step 1.

Step 1

In this step we will write a function (`read_data`) which will ask user for a set of numbers, one number at a time. It will return a list of numbers. It will complete the task as follows:

- Initialize an empty list.
- Use a while loop that will iterate as long as user has more data. You can either use a y/n prompt or ask the user to press enter key (empty string) when they are done.
  - Prompt the user for and read in the next number.
- Return the list.

A block of comments after the start of the function will be used to document our function. An example of such a block is as follows:

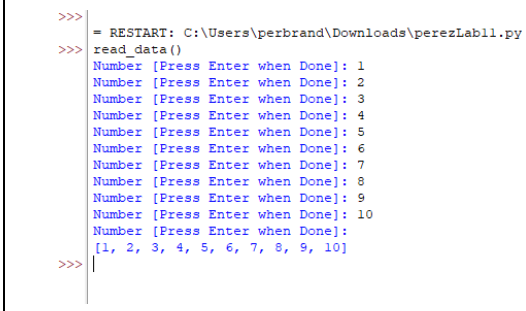
```
def read_data ():
    """
    Function Name: read_data
    Description: Prompts user for and read in a set of
    numbers, one number at a time.
    Parameter: None
    Returns list of numbers
    """
   # Code for the function

   return list_of_numbers
```

Make sure that you have similar block of comments after start each of the functions that we will develop in this lab. This is a special way of documenting python functions. It can be used by the Python help feature to provide information about the function to the user of the function. Once you are done with writing the code for the function you can test it as follows:

- Load your program into shell by selecting "Run Module" from Run menu.
- Type `read_data()` in shell to run the function. Enter the following numbers as it prompts for the next number: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
- It will return a list of numbers as follows:
  `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`
- If it returns a list of `strings` instead of `ints` that mean that you are missing a step in the read process.
- If there are any errors then make sure you fix those errors and run the function again.
- Once you are satisfied with the result of your step 1, take a screen shot and paste it below.
- Type help (read_data) to display information about read_data function. This will display the block of comments that we added after the start of the function.
- Note: You must complete this step before starting on step 2.

Take a screen shot of your results and paste it in the box below:

Screen Shot 1



Sample I/O

```
>>> read_data()
    Enter a number (press enter only when done)1
    Enter a number (press enter only when done)2
    Enter a number (press enter only when done)3
    Enter a number (press enter only when done)4
    Enter a number (press enter only when done)5
    Enter a number (press enter only when done)6
    Enter a number (press enter only when done)7
    Enter a number (press enter only when done)8
    Enter a number (press enter only when done)9
    Enter a number (press enter only when done)10
    Enter a number (press enter only when done)
    [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>>> help (read_data)
    Help on function read_data in module __main__:

    read_data()
        Function Name: read_data
        Description: Prompts user for and read in a set of
        numbers, one number at a time.

        Parameter: None
        Returns list of numbers
```

Step 2

In this step we will write a function (`compute_sum`) to calculate the sum of the numbers in a list. It will receive a list of integer numbers as parameter and will return the sum of the numbers in the given list. It will use a definite loop to add the numbers in the given list.

Once you are done with writing the code for the function you can test it as follows:

- Load your program into shell by selecting "Run Module" from Run menu.
- Type:
  `my_list = read_data()`
- Enter the following numbers as it prompts for the next number: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
- Type
  `compute_sum (my_list)`
- It will return `55`
- If there are any errors then make sure you fix those errors and run the function again.
- Once you are satisfied with the result of your step 2, take a screen shot and paste it below.
- Note: You must complete this step before starting on step 3.

Take a screen shot of your results and paste it in the box below:

Screen Shot 2

```
= RESTART: C:\Users\perbrand\Downloads\perezLab11.py
>>> my_list = read_data()
Number [Press Enter when Done]: 1
Number [Press Enter when Done]: 2
Number [Press Enter when Done]: 3
Number [Press Enter when Done]: 4
Number [Press Enter when Done]: 5
Number [Press Enter when Done]: 6
Number [Press Enter when Done]: 7
Number [Press Enter when Done]: 8
Number [Press Enter when Done]: 9
Number [Press Enter when Done]: 10
Number [Press Enter when Done]:
>>> compute_sum(my_list)
55
```

Step 3

In this step we will write a function (`compute_mean`) to calculate the mean of the numbers in a list. It will receive a list of integer numbers as parameter and will return the mean of the numbers in the given list. It will complete the task as follows:

- Use `compute_sum` function to calculate sum of the numbers in the list.
- Calculate the size of the list.
- Divide sum of the numbers by the size of the list to compute the mean.
- Return the mean of the numbers in the list.

Once you are done with writing the code for the function you can test it as follows:

- Load your program into shell by selecting "Run Module" from Run menu.
- Type:
  `my_list = read_data()`
- Enter the following numbers as it prompts for the next number: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
- Type:
  `compute_mean (my_list)`
- It will return `5.5`
- If there are any errors then make sure you fix those errors and run the function again.
- Once you are satisfied with the result of your step 3, take a screen shot and paste it below.
- Note: You must complete this step before starting on step 4.

Take a screen shot of your results and paste it in the box below:

Screen Shot 3

```
>>>
    = RESTART: C:\Users\perbrand\Downloads\perezLab11.py
>>> my_list = read_data()
    Number [Press Enter when Done]: 1
    Number [Press Enter when Done]: 2
    Number [Press Enter when Done]: 3
    Number [Press Enter when Done]: 4
    Number [Press Enter when Done]: 5
    Number [Press Enter when Done]: 6
    Number [Press Enter when Done]: 7
    Number [Press Enter when Done]: 8
    Number [Press Enter when Done]: 9
    Number [Press Enter when Done]: 10
    Number [Press Enter when Done]:
>>> compute_mean(my_list)
    5.5
```

Step 4

In this step we will write a function (`compute_sd`) to calculate the standard deviation of the numbers in a list. It will receive a list of integer numbers as parameter and will return the standard deviation of the numbers in the given list. It will complete the task as follows:

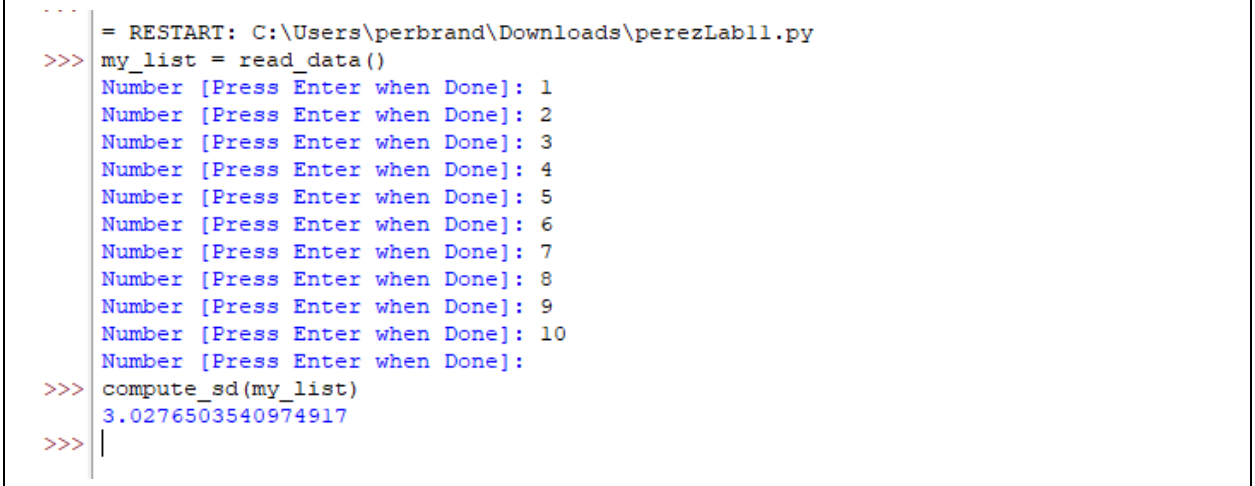
- Use `compute_mean` function to calculate mean of the numbers in the list.
- Initialize `sum` to 0
- Use definite loop (`for num in list_of_numbers`)
  - Calculate `deviation` by subtracting `mean` from the `num`
  - Calculate the square of the `deviation`
  - Add the square of the `deviation` to `sum`
- Calculate the `size` of the list.
- Divide `sum` by the **`size-1`** to compute the `result`.
- Take the square root of the `result` to compute the standard deviation.
- Return the standard deviation of the numbers in the list.

Once you are done with writing the code for the function you can test it as follows:

- Load your program into shell by selecting "Run Module" from Run menu.
- Type:
  ```
  my_list = read_data()
  ```
- Enter the following numbers as it prompts for the next number: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
- Type:
  ```
  compute_sd (my_list)
  ```
- It will return `3.0276503540974917`
- If there are any errors then make sure you fix those errors and run the function again.
- Once you are satisfied with the result of your step 4, take a screen shot and paste it below.
- Note: You must complete this step before starting on step 5.

Take a screen shot of your results and paste it in the box below:

Screen Shot 4



Step 5

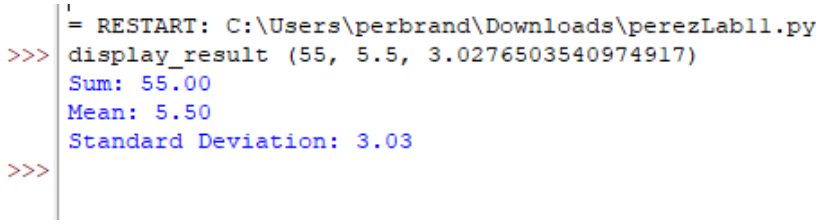
In this step we will write a function (`display_result`) to display the sum, mean, and standard deviation of the given list of numbers. It will receive sum, mean and standard deviation as parameters. It will complete the task as follows:

- Display the result (sum, mean, and standard deviation – one per line) with appropriate labels. Make sure you format the numbers to two decimal places.

Once you are done with writing the code for the function you can test it as follows:

- Load your program into shell by selecting "Run Module" from Run menu.
- Type:
  `display_result (55, 5.5, 3.0276503540974917)`
- The output will be as follows:
  ```
  Sum is: 55.00
  Mean is: 5.50
  Standard Deviation is: 3.03
  ```
- If there are any errors then make sure you fix those errors and run the function again.
- Once you are satisfied with the result of your step 5, take a screen shot and paste it below.
- Note: You must complete this step before starting on step 6.

Take a screen shot of your results and paste it in the box below:

Screen Shot 5


Step 6

In this step we will write `main` function which will use the functions developed in steps 1 to 5 to read the data, calculate sum, mean & standard deviation, and display the result. It will complete the task as follows:

- Use `read_data` function to read the data.
- Use the `compute_sum` function to compute the sum of the numbers.
- Use `compute_mean` function to compute the mean of the numbers.
- Use `compute_sd` function to compute the standard deviation of the numbers.
- Use `display_result` function to display the result.

Once you are done with writing the code for the function you can test it as follows:

- Load your program into shell by selecting "Run Module" from Run menu.
- Type:
  ```
  main ()
  ```
- Enter the following numbers as it prompts for the next number: 17, 53, 39, 96, 39, 33, 75, 1, 68, 22, 9, 88
- The output file will be as follows:
  ```
  Sum is: 540
  Mean is: 45.00
  Standard Deviation is: 31.19
  ```
- If there are any errors then make sure you fix those errors and run the function again.
- Once you are satisfied with the result of your step 6, take a screen shot and paste it below.

Take a screen shot of your results and paste it in the box below:

Screen Shot 6

```
>>>
    = RESTART: C:\Users\perbrand\Downloads\perezLab11.py
>>> main()
    Number [Press Enter when Done]: 17
    Number [Press Enter when Done]: 53
    Number [Press Enter when Done]: 39
    Number [Press Enter when Done]: 96
    Number [Press Enter when Done]: 39
    Number [Press Enter when Done]: 33
    Number [Press Enter when Done]: 75
    Number [Press Enter when Done]: 1
    Number [Press Enter when Done]: 68
    Number [Press Enter when Done]: 22
    Number [Press Enter when Done]: 9
    Number [Press Enter when Done]: 88
    Number [Press Enter when Done]:

    Sum: 540.00
    Mean: 45.00
    Standard Deviation: 31.19
>>>
```

Add the following comment block at the top of your file. Note that we are using a different format then what we used before:

```
"""
Name:
Class: CSCI 1411-00X
Due Date:
Description:
Status:
"""
```

Rubric for Lab 11:

Criteria	Rating
Step I: read_data (Screen shot 1)	Screen shot included – 5 points No screen shot included – 0 points
Step 1: read_data	Reads data one at a time – 5 points Reads all the data at the same time – 2 points Does not read the data – 0 points
Step 1: read_data	Returns a list – 5 points Does not return a list – 0 points
Step 2: compute_sum (Screen shot 2)	Screen shot included – 5 points No screen shot included – 0 points
Step 2: compute_sum	Correctly computes the sum – 5 points Does not correctly compute the sum – 0 points
Step 3: compute_mean (Screen shot 3)	Screen shot included – 5 points No screen shot included – 0 points
Step 3: compute_mean	Uses compute_sum function to compute the sum – 5 points Does not use compute_sum function to compute the sum – 0 points
Step 3: compute_mean	Correctly computes the mean – 5 points Does not correctly computes the mean – 0 points
Step 4: compute_sd (Screen shot 4)	Screen shot included – 5 points No screen shot included – 0 points
Step 4: compute_sd	Uses compute_mean to compute the mean – 5 points Does not use compute_mean to compute the mean – 0 points
Step 4: compute_sd	Correctly computes the sd – 5 points Does not correctly computes the sd – 0 points
Step 5: display_result (Screen shot 5)	Screen shot included – 5 points No screen shot included – 0 points
Step 5: display_result	Displays the output with labels – 5 points Displays the outputs without any labels – 2 points Does not display any output – 0 points
Step 6: main	Screen shot included – 5 points No screen shot included – 0 points
Step 6: main	Uses the function to perform different tasks – 5 points Does not use functions to perform different tasks – 0 points
Total	75