Worksheet 7: Modularity

Thi Van Anh DUONG Student ID: 90023112

Diploma of Information Technology, Curtin College

ISEN1000: Introduction to Software Engineering

Coordinator: Khurram Hameed

2 May 2022

Student Declaration of Originality

☒	This assignment is my own original work, and no part has been copied from another student's work or source, except where it is clearly attributed.
☒	All facts/ideas/claims are from academic sources are paraphrased and cited/referenced correctly.
☒	I have not previously submitted this work in any form for THIS or for any other unit; or published it elsewhere before.
☒	No part of this work has been written for me by another person.
☒	I recognise that should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with Curtin College policy.

Electronic Submission:

☒	I accept that it is my responsibility to check that the submitted file is the correct file, is readable and has not been corrupted.
☒	I acknowledge that a submitted file that is unable to be read cannot be marked and will be treated as a non-submission.
☒	I hold a copy of this work if the original is damaged, and will retain a copy of the Turnitin receipt as evidence of submission.

1. Discussion

(a) Do global constants create a modularity problem? Why or Why not?

Answer: The answer would be **yes** or **no** and it also depend on how you use them.

Yes, it creates coupling issue when you abuse using it. This is simply because the global variables and global constant are just the same in Python. They can cause coupling issue like:

- Tight coupling: exist between modules and the content of one can greatly affect others).
- It is difficult to work with when it comes to understanding all of modules at the same time, time consuming to write, test and debug.

No, if you use it in a wise manner. For example, if we have multiple modules in a same program about something, we can take advantage of using global constants to make our code more readable and shorter instead of moving back and forth between different functions just to avoid using global constants.

(b) If a method/function takes 10 parameters, how/why could this create a high level of coupling?

Answer: Assuming I have the function as below:

Def function1(para1, para2, para3, para4, para5, para6, para7, para8, para9, para10)

➔ It creates high coupling problem. Too many parameters will be difficult to manipulate through the program (design the code, test and debug as well as maintenance).

➔ We should not have more than 6 parameters.

(c) Why do we want to avoid code duplication?

Answer: There are 3 main reasons why we should avoid code duplication:

- Duplication code definitely makes our program lengthy and complex without any benefits. Furthermore, it aslo consumes more space in your computer and slow down out program's speed.
- Duplication code affects our code quality as it is guaranteed to fail simultaneously (under the same condition), so our software program cannot last too long.
- Duplication code waste time to debug: we have to fix every location that we put the code in just only to fix one bug which tremendously waste surplus amount of time and lose out temper as well.

- Duplication code increase the probability for the software system being hacked by hackers. This is due to the fact that, hackers can easily unlock the whole system if they already hack one attribute (that one we duplicate multiple times over our program).

2. Checklist

a. Does this code have global variables?

b. Does this code have any duplication code?

c. Is there any function having more than 6 parameters?

d. Are there any functions having flags?

e. Does all functions perform a well-defined task?

f. Does any function performs overlapping tasks?

g. Is Any parameter in incorrect order?

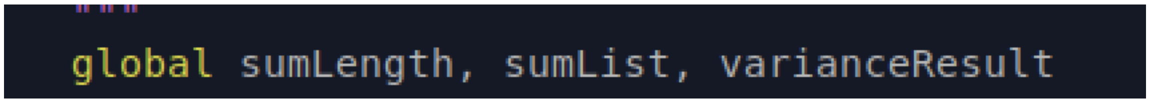
h. Do all parameters have correct data types?

i. Does the program have appropriate test cases/ values there?

3. Review

a. Does this code have global variables?

➔ **YES** (coupling issue)

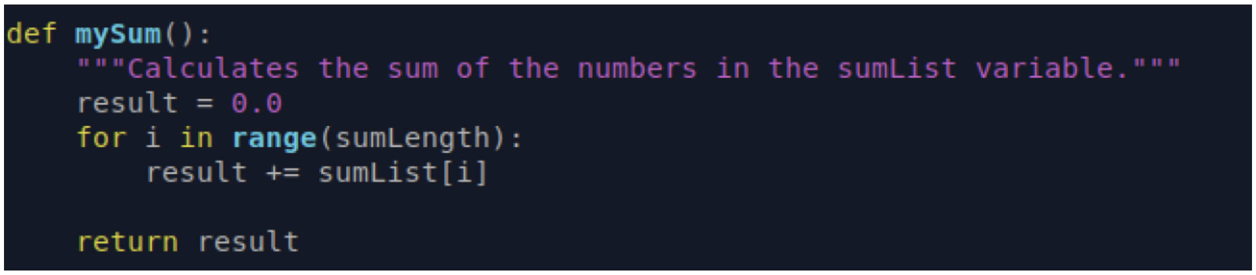


```
global sumLength, sumList, varianceResult
```

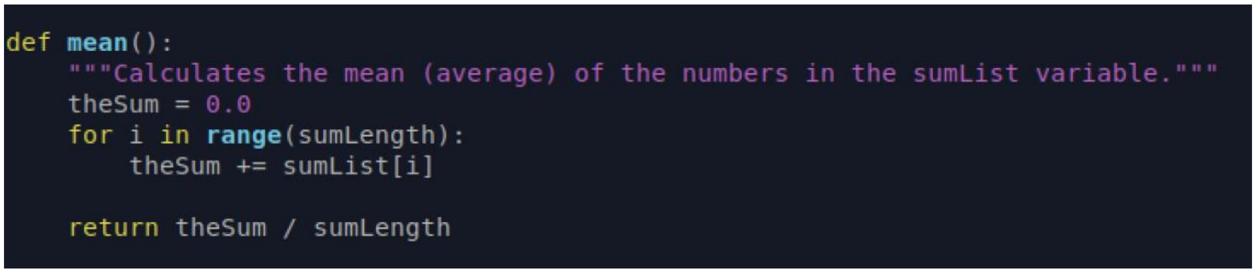
b. Does this code have any duplication code?

➔ **YES** (Redundancy)

def mySum() and def mean() as below all reuse the code to calculate the sum of numbers.



```
def mySum():
    """Calculates the sum of the numbers in the sumList variable."""
    result = 0.0
    for i in range(sumLength):
        result += sumList[i]

    return result
```



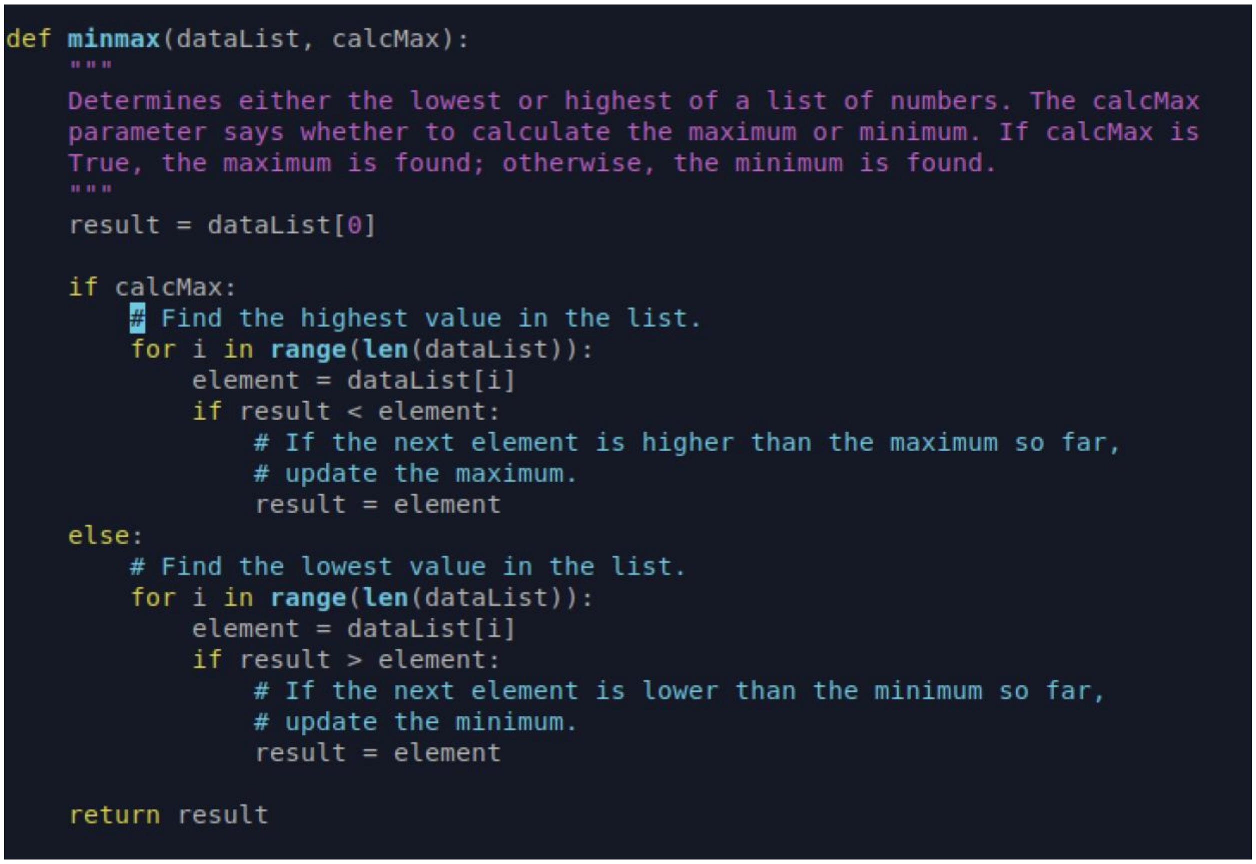
```python
def mean():
    """Calculates the mean (average) of the numbers in the sumList variable."""
    theSum = 0.0
    for i in range(sumLength):
        theSum += sumList[i]

    return theSum / sumLength
```

c. Is there any function having more than 6 parameters?

➔ NO

d. Are there any functions having flags?

➔ YES



```python
def minmax(dataList, calcMax):
    """
    Determines either the lowest or highest of a list of numbers. The calcMax
    parameter says whether to calculate the maximum or minimum. If calcMax is
    True, the maximum is found; otherwise, the minimum is found.
    """
    result = dataList[0]

    if calcMax:
        # Find the highest value in the list.
        for i in range(len(dataList)):
            element = dataList[i]
            if result < element:
                # If the next element is higher than the maximum so far,
                # update the maximum.
                result = element
    else:
        # Find the lowest value in the list.
        for i in range(len(dataList)):
            element = dataList[i]
            if result > element:
                # If the next element is lower than the minimum so far,
                # update the minimum.
                result = element

    return result
```

e. Does all functions perform a well-defined task?

➔ YES

f. Does any function performs overlapping tasks?

➔ NO

g. Is there any parameter in incorrect order?

➔ YES

h. Do all parameters have correct data types?

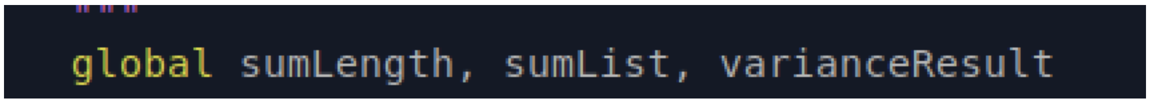
➔ YES

i. Does the program have appropriate values there?

➔ YES

4. Refactoring

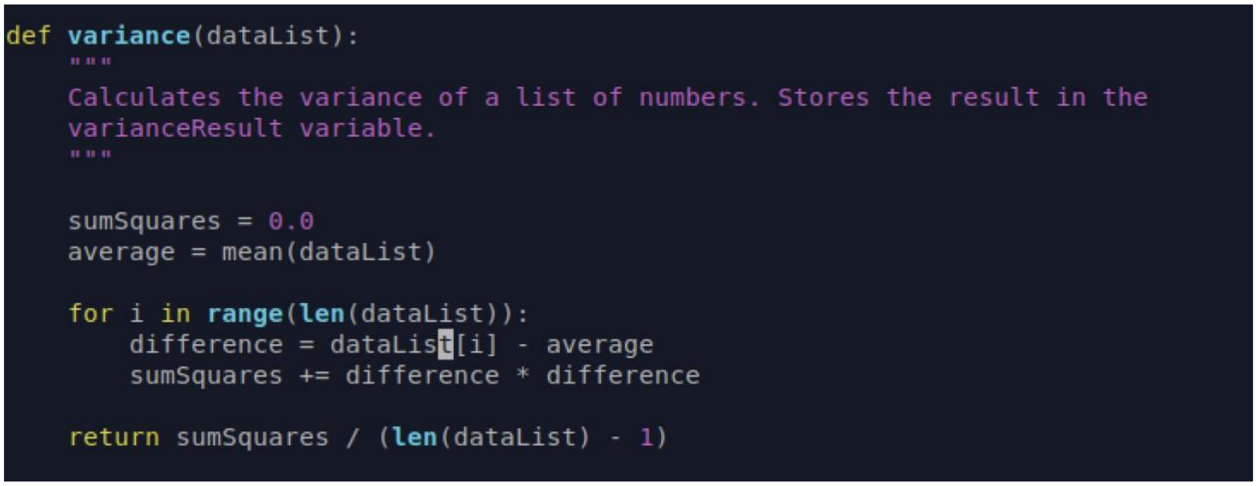
- **Coupling issue** with global variables



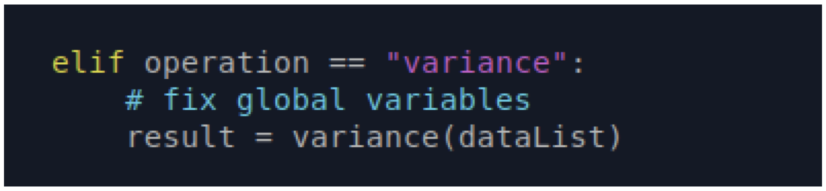
```
    global sumLength, sumList, varianceResult
```

(a) *How to fix:* I deleted all global variables, and return the variance result values in the function. When we calculate variance, we just need to call the variance(dataList) again.

(b) Here is my changes:

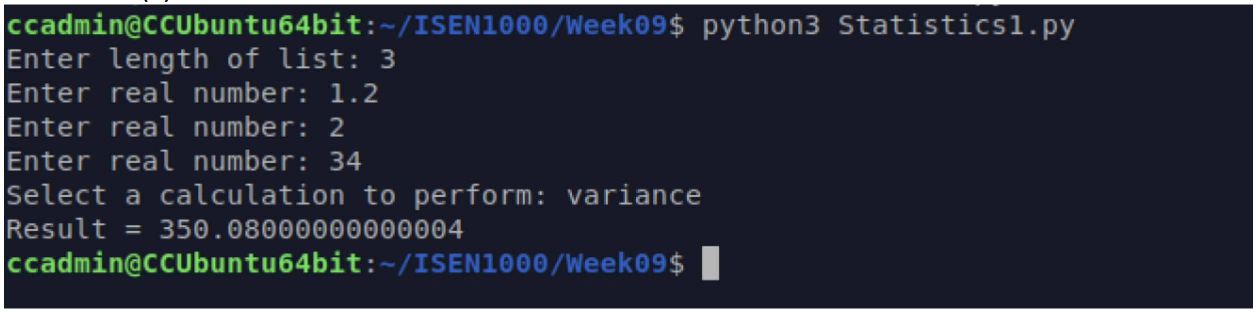


```
def variance(dataList):
    """
    Calculates the variance of a list of numbers. Stores the result in the
    varianceResult variable.
    """

    sumSquares = 0.0
    average = mean(dataList)

    for i in range(len(dataList)):
        difference = dataList[i] - average
        sumSquares += difference * difference

    return sumSquares / (len(dataList) - 1)
```



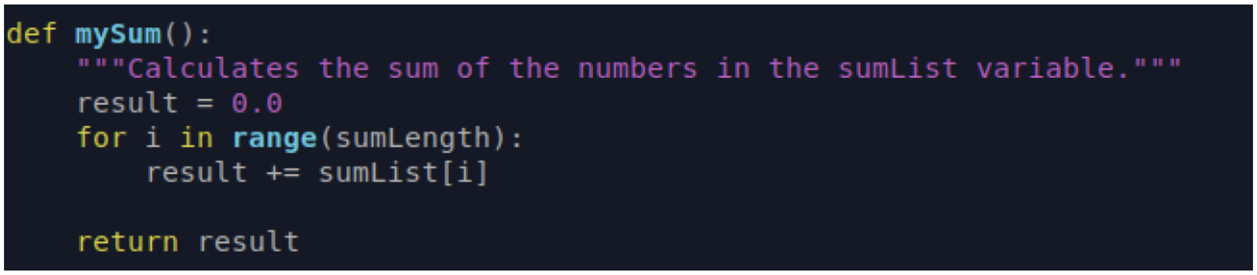
```
elif operation == "variance":
    # fix global variables
    result = variance(dataList)
```

(c) And it still works!

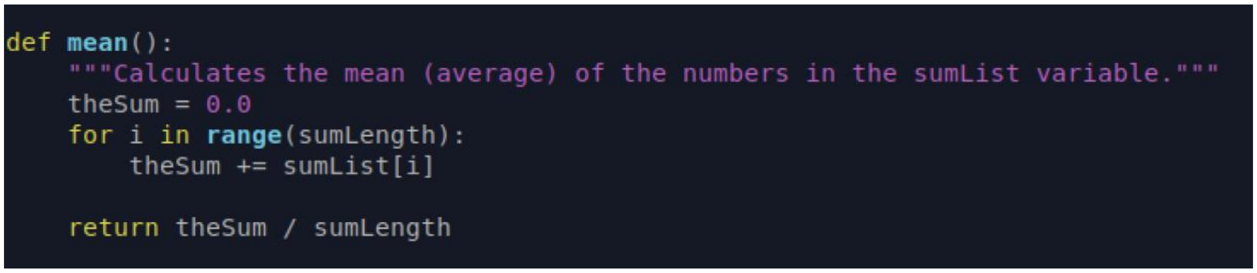


```
ccadmin@CCUbuntu64bit:~/ISEN1000/Week09$ python3 Statistics1.py
Enter length of list: 3
Enter real number: 1.2
Enter real number: 2
Enter real number: 34
Select a calculation to perform: variance
Result = 350.08000000000004
ccadmin@CCUbuntu64bit:~/ISEN1000/Week09$ 
```

- **Redundancy issue**



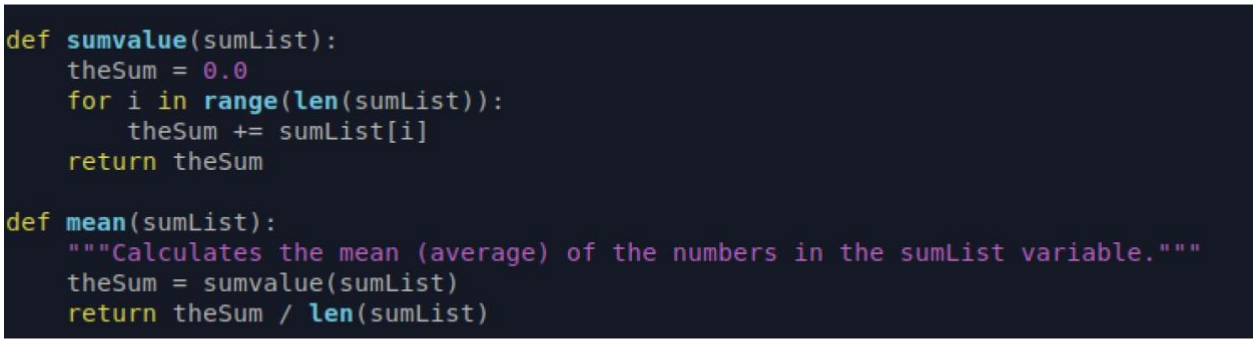
```python
def mySum():
    """Calculates the sum of the numbers in the sumList variable."""
    result = 0.0
    for i in range(sumLength):
        result += sumList[i]

    return result
```



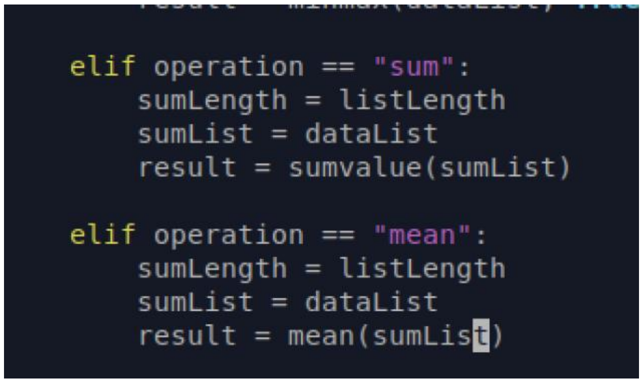
```python
def mean():
    """Calculates the mean (average) of the numbers in the sumList variable."""
    theSum = 0.0
    for i in range(sumLength):
        theSum += sumList[i]

    return theSum / sumLength
```

(a) *How to fix*: I remove the duplication code and put it in a new function called: def sumvalue() and delete mySum() function as the code below. When I use mysum() functiona and mean() function, I just need to call my sumvalue() again.

(b) Here is my changes:

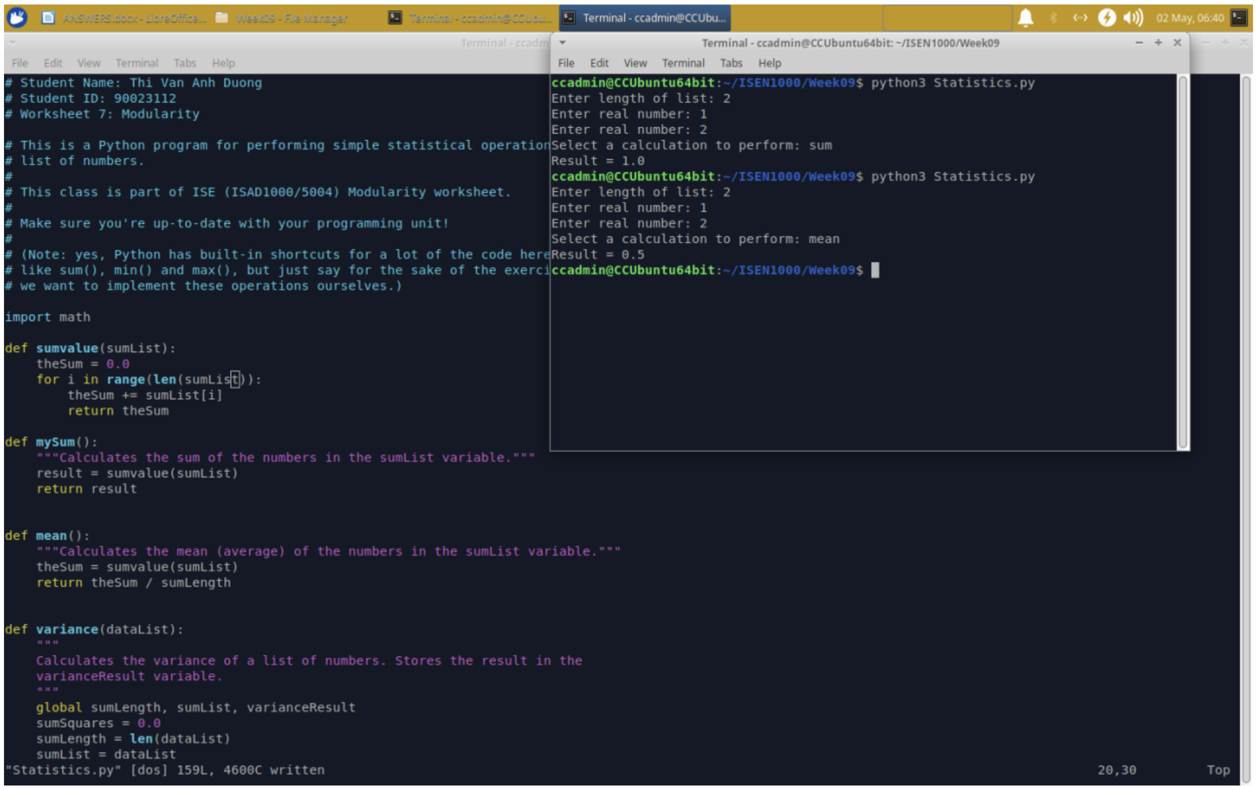


```python
def sumvalue(sumList):
    theSum = 0.0
    for i in range(len(sumList)):
        theSum += sumList[i]
    return theSum

def mean(sumList):
    """Calculates the mean (average) of the numbers in the sumList variable."""
    theSum = sumvalue(sumList)
    return theSum / len(sumList)
```



```python
    elif operation == "sum":
        sumLength = listLength
        sumList = dataList
        result = sumvalue(sumList)

    elif operation == "mean":
        sumLength = listLength
        sumList = dataList
        result = mean(sumList)
```

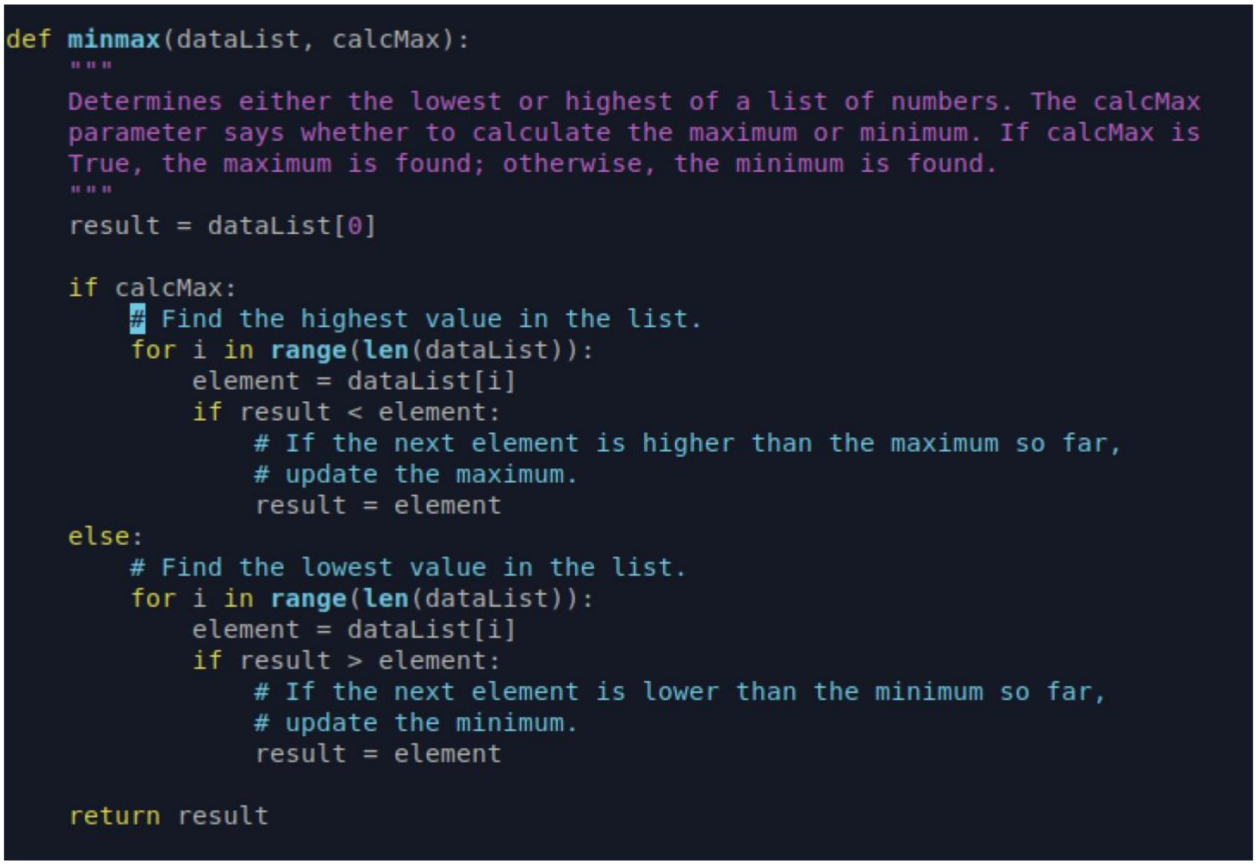
(c) And the code still works!



```
# Student Name: Thi Van Anh Duong
# Student ID: 90023112
# Worksheet 7: Modularity

# This is a Python program for performing simple statistical operation
# list of numbers.
#
# This class is part of ISE (ISAD1000/5004) Modularity worksheet.
#
# Make sure you're up-to-date with your programming unit!
#
# (Note: yes, Python has built-in shortcuts for a lot of the code here
# like sum(), min() and max(), but just say for the sake of the exerci
# we want to implement these operations ourselves.)

import math

def sumvalue(sumList):
    theSum = 0.0
    for i in range(len(sumList)):
        theSum += sumList[i]
        return theSum

def mySum():
    """Calculates the sum of the numbers in the sumList variable."""
    result = sumvalue(sumList)
    return result


def mean():
    """Calculates the mean (average) of the numbers in the sumList variable."""
    theSum = sumvalue(sumList)
    return theSum / sumLength


def variance(dataList):
    """
    Calculates the variance of a list of numbers. Stores the result in the
    varianceResult variable.
    """
    global sumLength, sumList, varianceResult
    sumSquares = 0.0
    sumLength = len(dataList)
    sumList = dataList
"Statistics.py" [dos] 159L, 4600C written                                20,30          Top
```

```
ccadmin@CCUbuntu64bit:~/ISEN1000/Week09$ python3 Statistics.py
Enter length of list: 2
Enter real number: 1
Enter real number: 2
Select a calculation to perform: sum
Result = 1.0
ccadmin@CCUbuntu64bit:~/ISEN1000/Week09$ python3 Statistics.py
Enter length of list: 2
Enter real number: 1
Enter real number: 2
Select a calculation to perform: mean
Result = 0.5
ccadmin@CCUbuntu64bit:~/ISEN1000/Week09$
```

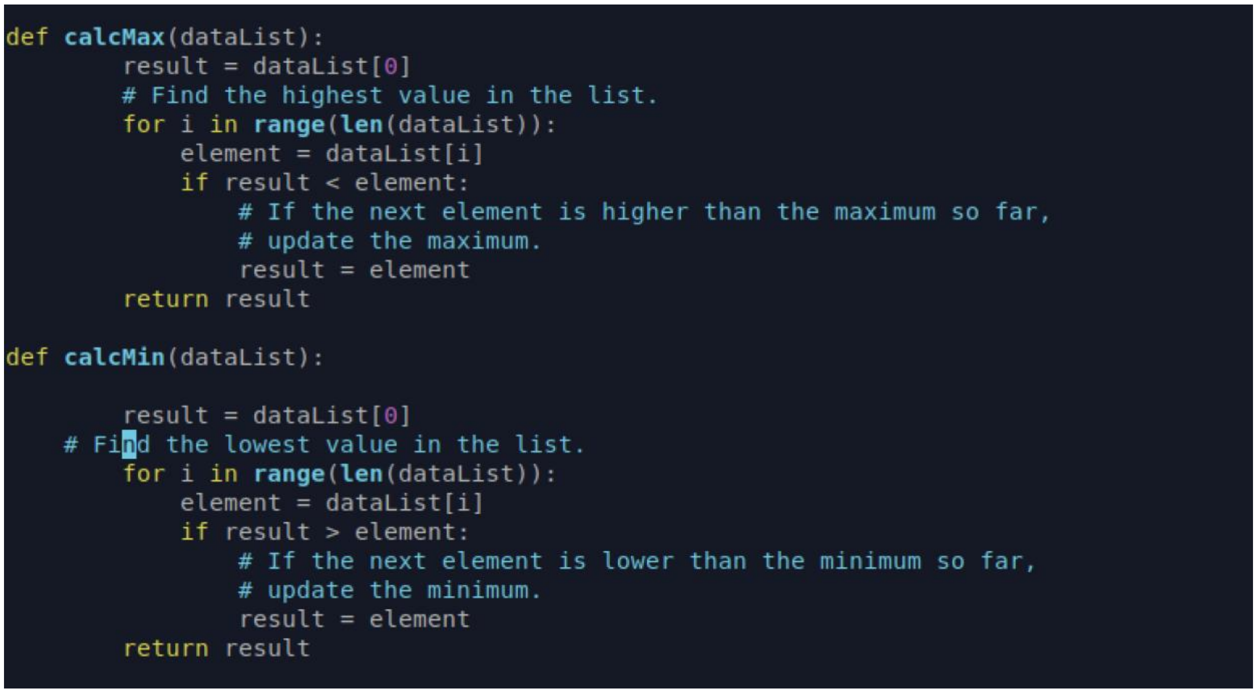
- **Control flag**



```
def minmax(dataList, calcMax):
    """
    Determines either the lowest or highest of a list of numbers. The calcMax
    parameter says whether to calculate the maximum or minimum. If calcMax is
    True, the maximum is found; otherwise, the minimum is found.
    """
    result = dataList[0]

    if calcMax:
        # Find the highest value in the list.
        for i in range(len(dataList)):
            element = dataList[i]
            if result < element:
                # If the next element is higher than the maximum so far,
                # update the maximum.
                result = element
    else:
        # Find the lowest value in the list.
        for i in range(len(dataList)):
            element = dataList[i]
            if result > element:
                # If the next element is lower than the minimum so far,
                # update the minimum.
                result = element

    return result
```

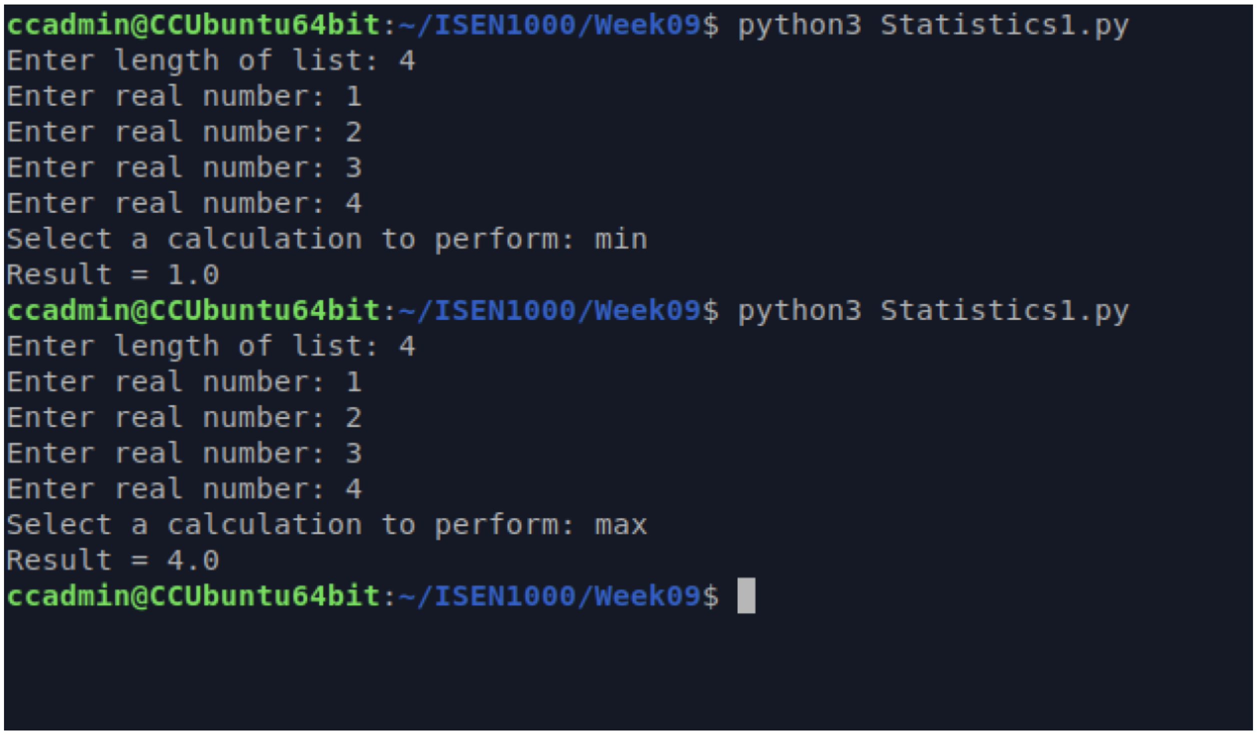
(a) *How to fix*: I divide the minmax function into two separate functions called: calcMin() and calcMax().

(b) Here is my changes:



```
def calcMax(dataList):
        result = dataList[0]
        # Find the highest value in the list.
        for i in range(len(dataList)):
            element = dataList[i]
            if result < element:
                # If the next element is higher than the maximum so far,
                # update the maximum.
                result = element
        return result

def calcMin(dataList):

        result = dataList[0]
    # Find the lowest value in the list.
        for i in range(len(dataList)):
            element = dataList[i]
            if result > element:
                # If the next element is lower than the minimum so far,
                # update the minimum.
                result = element
        return result
```

(c) And it still works!



```
ccadmin@CCUbuntu64bit:~/ISEN1000/Week09$ python3 Statistics1.py
Enter length of list: 4
Enter real number: 1
Enter real number: 2
Enter real number: 3
Enter real number: 4
Select a calculation to perform: min
Result = 1.0
ccadmin@CCUbuntu64bit:~/ISEN1000/Week09$ python3 Statistics1.py
Enter length of list: 4
Enter real number: 1
Enter real number: 2
Enter real number: 3
Enter real number: 4
Select a calculation to perform: max
Result = 4.0
ccadmin@CCUbuntu64bit:~/ISEN1000/Week09$
```