LAB 10-2 ASSIGNMENT

Ch. 6 Functions
Ch. 7 Input Validation

Start: 04/30/2024 02:17 PM
Name: Imani Hollie

CH. 6 EXCERCISE 7 – TEST AVERAGE AND GRADE

Write the Algorithm, Pseudocode, Flowchart, and Python Code for the following programming problem:

Scenario: Test Average and Grade

Write a program that asks the user to enter five text scores. The program should display a letter grade for each score and the average test score. Design the following functions in the program:

- `calcAverage` – This function should accept five test scores as an argument and return the average of the scores.
- `determineGrade` – This function should accept a test score as an argument and return a letter grade for the score (as a `String`), based on the following grading scale:

SCORE	GRADE
90 – 100	A
80 – 89	B
70 – 79	C
60 – 69	D
BELOW 60	F

Step 1: The Algorithm

1. MODULE 1 – main()
    a. Get the test scores:
        i. Prompt for Score 1
        ii. Prompt for Score 2
        iii. Prompt for Score 3
        iv. Prompt for Score 4
        v. Prompt for Score 5
    b. Call Module 2
    c. Call Module 3
2. MODULE 2 – calcAverage()
    a. Calculate the average of the five scores:
        i. Add all the scores and divide by five for the average.
    b. Display the average score calculated:
        i. Display Average Score
3. MODULE 3 – determineGrade()
    a. Calculate and display the letter grade of the average score:
        i. If the average score is between 90 – 100, then display A
        ii. Else, if the average score is between 80 – 89, then display B
        iii. Else, if the average score is between 70 – 79, then display C
        iv. Else, if the average score is between 60 – 69, then display D
        v. Else, if the average score is below 60, then display F

The Input, Processing, and Output

Table 1-1 Calculating Average of Scores (x)			
INPUTS	**Input Type**	**Value**	**Data Type**
Score 1 (s1)	Variable	(a)	Float
Score 2 (s2)	Variable	(b)	Float
Score 3 (s3)	Variable	(c)	Float
Score 4 (s4)	Variable	(d)	Float
Score 5 (s5)	Variable	(e)	Float
PROCEDURE	$x = \frac{a+b+c+d+e}{5}$ $avgScore = \frac{s1+s2+s3+s4+s5}{5}$		
OUTPUTS	**Output Type**	**Value**	**Data Type**
Average Score (avgScore)	Variable	(x)	Float

The IPO for Table 1-1 is as follows:

1. The inputs for Table 1-1 are as follows:
    a. Score 1 (a)
    b. Score 2 (b)
    c. Score 3 (c)
    d. Score 4 (d)
    e. Score 5 (e)
2. The procedure for Table 1-1 are as follows:
    a. $x = \frac{a+b+c+d+e}{5}$
    $avgScore = \frac{s1+s2+s3+s4+s5}{5}$
3. The output for Table 1-1 are as follows:
    a. Average Score (x)

Table 1-2 Nested Decision Structure: Letter Grade									
Avg. Score 101 < x > 89 (100 – 90)		**Avg. Score 91 < x > 79 (89 – 80)**		**Avg. Score 100 < x > 90 (79 – 70)**		**Avg. Score 100 < x > 90 (69 – 60)**		**Avg. Score 60 < x (59 – 0)**	
avgScore (x)	A	**avgScore (x)**	B	**avgScore (x)**	C	**avgScore (x)**	D	**avgScore (x)**	F

Step 2: The Pseudocode

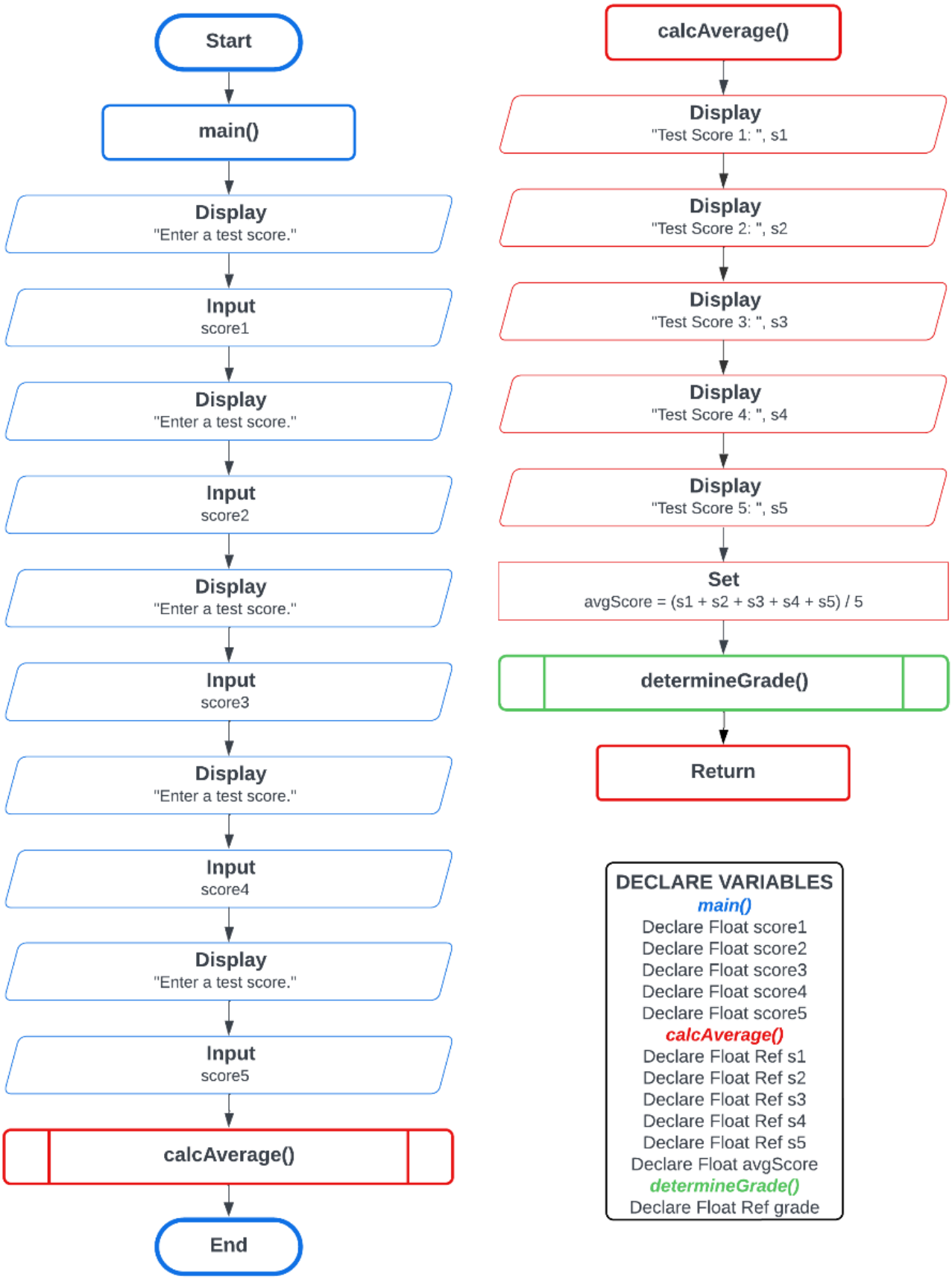
Refer to Tables 1-1 and 1-2 in Step 1 for the needed variables.

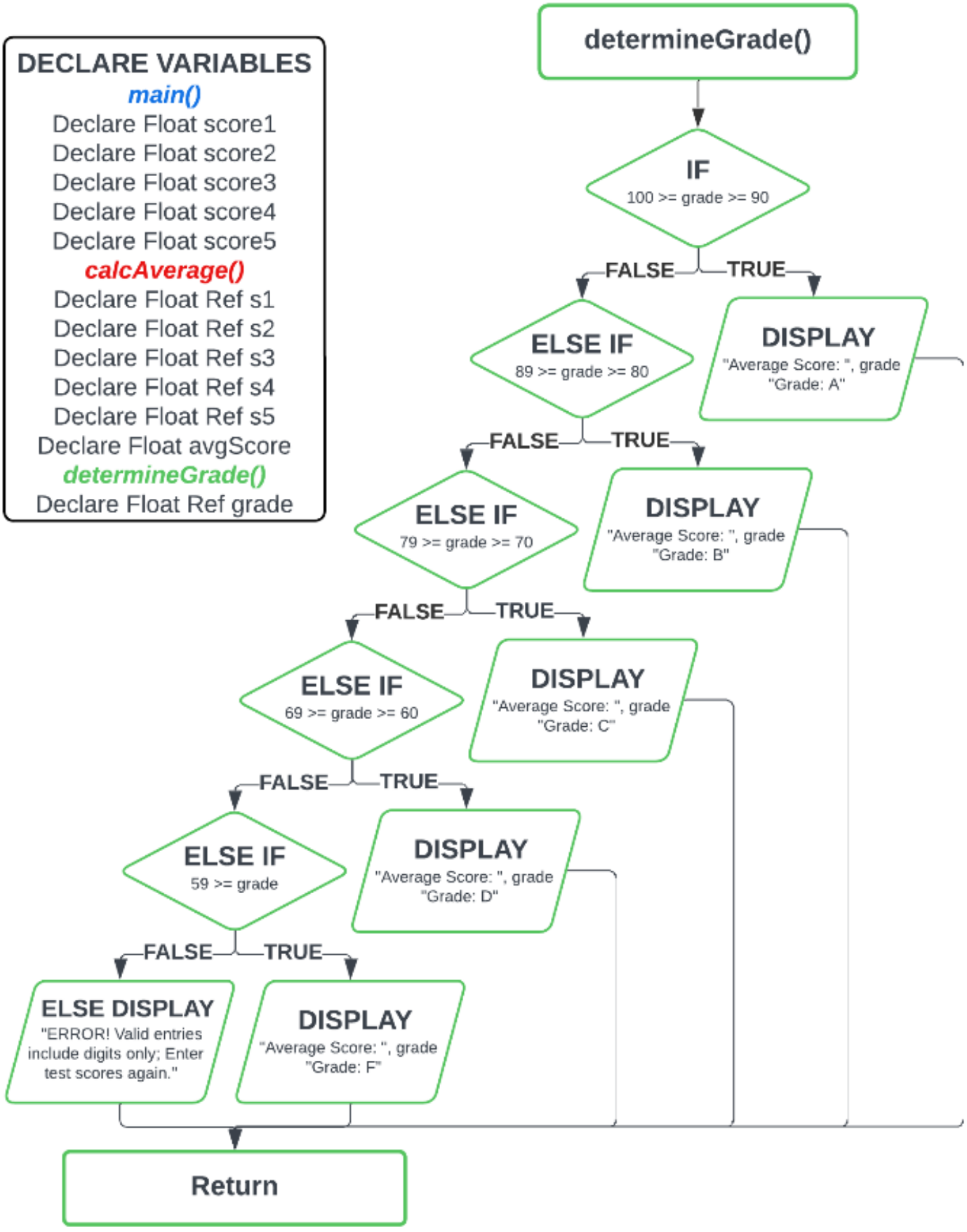
```
1. //This program takes in five test scores.
2. //Output is then printed to the screen.

3. //Declare the main module
4. //main() input and calls calcAverage() and determineGrade()
5. Module main()

   a. //Declare variables
   b. Declare Float score1
   c. Declare Float score2
   d. Declare Float score3
   e. Declare Float score4
   f. Declare Float score5

   g. //Input test scores
   h. Display "Enter a test score."
   i. Input score1
   j. Display "Enter a test score."
   k. Input score2
   l. Display "Enter a test score."
   m. Input score3
   n. Display "Enter a test score."
   o. Input score4
   p. Display "Enter a test score."
   q. Input score5

   r. //Call function
   s. Call calcAverage(score1, score2, score3, score4, score5)
6. End Module

7. //Declare the calcAverage function
8. //calcAverage() calculates
9. Function calcAverage(Float Ref s1, s2, s3, s4, s5)

   a. //Declare variables
   b. Declare Float s1
   c. Declare Float s2
   d. Declare Float s3
   e. Declare Float s4
   f. Declare Float s5
   g. Declare Float avgScore

   h. //Calculate avgScore
   i. Set avgScore = (s1 + s2 + s3 + s4 + s5) / 5

   j. //Call module
   k. Call determineGrade(avgScore)
10. End Function
```

```
11. //Declare determineGrade Function
12. //determineGrade() calculates and outputs
13. Function determineGrade(Float Ref avgScore)

    l. //Declare variables
    m. Declare Float grade

    n. //Calculate grade with nested If-Else-Then


    o. //1st If calculates for 100 - 90 (A)
    p. If 100 >= grade >= 90 Then
        ▪ Display "Average Score: ", grade
        ▪ Display "Grade: A"

        ▪ //2nd If calculates for 89 - 80 (B)
        ▪ Else If 89 >= grade >= 80 Then
            ▪ Display "Average Score: ", grade
            ▪ Display "Grade: B"

            ▪ //3rd If calculates for 79 - 70 (C)
            ▪ Else If 79 >= grade >= 70 Then
                ▪ Display "Average Score: ", grade
                ▪ Display "Grade: C"

                ▪ //4th If calculates for 69 - 60 (D)
                ▪ Else If 69 >= grade >= 60 Then
                    ▪ Display "Average Score: ", grade
                    ▪ Display "Grade: D"

                    ▪ //5th If calculates for 59 ≤ x (F)
                    ▪ Else If 59 >= grade Then
                        ▪ Display "Average Score: ", grade
                        ▪ Display "Grade: F"

                    ▪ //Output and error message if argument passes
                    ▪ Else
                        ▪ Display "ERROR! Enter test scores in digits."

                      //End the If-Then-Else Loop
                    ▪ End If
                ▪ End If
            ▪ End If
        ▪ End If
    q. End If

14. End Function
```

Step 3: The Flowchart

Refer to the png file submitted along with the PDF file as it contains the Flowchart.




DECLARE VARIABLES
main()
Declare Float score1
Declare Float score2
Declare Float score3
Declare Float score4
Declare Float score5
calcAverage()
Declare Float Ref s1
Declare Float Ref s2
Declare Float Ref s3
Declare Float Ref s4
Declare Float Ref s5
Declare Float avgScore
determineGrade()
Declare Float Ref grade
determineGrade()
IF
100 >= grade >= 90
FALSE
TRUE
DISPLAY
"Average Score: ", grade
"Grade: A"
ELSE IF
89 >= grade >= 80
FALSE
TRUE
DISPLAY
"Average Score: ", grade
"Grade: B"
ELSE IF
79 >= grade >= 70
FALSE
TRUE
DISPLAY
"Average Score: ", grade
"Grade: C"
ELSE IF
69 >= grade >= 60
FALSE
TRUE
DISPLAY
"Average Score: ", grade
"Grade: D"
ELSE IF
59 >= grade
FALSE
TRUE
ELSE DISPLAY
"ERROR! Valid entries include digits only; Enter test scores again."
DISPLAY
"Average Score: ", grade
"Grade: F"
Return

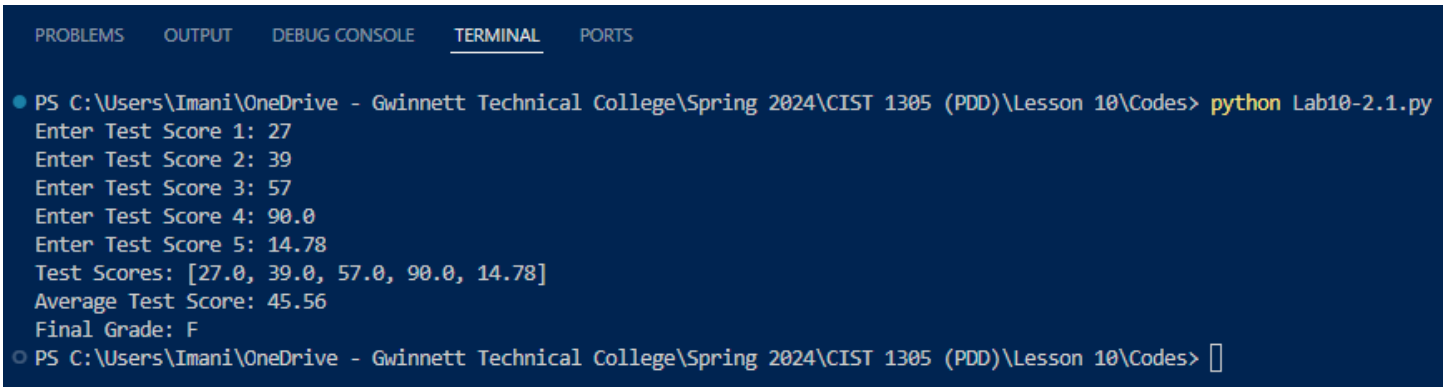
Step 4: The Python Code

Refer to the txt file submitted along with the PDF file as it contains the Python Code.

```
#Imani Hollie 04.30.2024
#This program takes five test scores and calculates
#the average score and grade or displays an error message.

#Module 1 - main() [Calls getScores(), calcAverage() and determineGrade() and Outputs]

#Modularize the scores for simplification and cleaner code
#Function 1 - getScores() [Inputs and Outputs]
def getScores():
    #Inputs-----------------------------------------------------------------------------
    #This is the same as repeating 'score1 = float(input('Enter Test Score 1: '))' five times
    #The for loop has a range of 5 and a counter ({num + 1}) to count/stop the loop, or:
    #for num in range(5):
        #score = float(input(f'Enter Test Score {num + 1}: '))
        #scores.append(score)
    scores = [float(input(f'Enter Test Score {num + 1}: ')) for num in range(5)]
    #Output-----------------------------------------------------------------------------
    return scores
#End Function 1

#Function 2 - caclAverage() [Calculates and Outputs]
#Instead of having test scores and averages printed here, just calculate the scores
def calcAverage(scores):
    #Calculations-----------------------------------------------------------------------
    #The sum function literally add all of the inputs
    avgScore = sum(scores) / 5
    #Output-----------------------------------------------------------------------------
    return avgScore
#End Function 2

#Function 3 - determineGrade() [Calculates and Outputs]
def determineGrade(avgScore):
    #Calculations-----------------------------------------------------------------------
    #IF-THEN-ELSE decision structure will display letter grade or an error message if false
    #IF 100 >= grade >= 90 THEN Display A
    if 100 >= avgScore >= 90:
        #This is the same as repeating 'print('Final Grade: #')'
        return 'A'
    #ELSE IF 89 >= grade >= 80 THEN Display B
    elif 89 >= avgScore >= 80:
        return 'B'
    #ELSE IF 79 >= grade >= 70 THEN Display C
    elif 79 >= avgScore >= 70:
        return 'C'
    #ELSE IF 69 >= grade >= 60 THEN Display D
```

```
    elif 69 >= avgScore >= 60:
        return 'D'
    #ELSE IF 59 >= grade >= 0 THEN Display F
    elif 59 >= avgScore >= 0:
        return 'F'
    #If argument passes through - display error message
    else:
        return 'ERROR! Valid entries include digits with 2 decimal places'
    #End If
#End Function 3

#Call Functions and Declare Variables-------------------------------------------------------
scores = getScores()
avgScore = calcAverage(scores)
grade = determineGrade(avgScore)
#Outputs-----------------------------------------------------------------------------------
#This is the same as repeating 'print(f'Test Score 1: {s1}')' five times
print(f'Test Scores: {scores}')
print(f'Average Test Score: {avgScore:.2f}')
print(f'Final Grade: {grade}')

#End Module 1
```

Screenshot of Terminal



TIC-TAC-TOE GAME AND MODIFICATION

CH. 6 EXERCISE 12 – ROCK, PAPER, SCISSORS GAME

Write the Algorithm, Pseudocode, Flowchart, and Python Code for the following programming problem:

Scenario: Rock, Paper, Scissors Game

Design a program that lets the user play a game of Rock, Paper, Scissors against the computer. The program should work as follows:

A. When the program begins. a random number in the range of 1 – 3 is generated. If the number is 1, then the computer has chosen rock; If the number is 2, then the computer has chosen paper; If the number is 3, then the computer has chosen scissors. (Don't display the computer's choice)
B. The user enters their choice of 'Rock', 'Paper', or 'Scissors', at the keyboard.
C. The computer's choice is displayed.
D. The program should display a message indicating whether the user or the computer was the winner. A winner is selected according to the following rules:
   - If one player chooses 'Rock' and the other chooses 'Scissors', the 'Rock' wins.
     - (`The rock smashes the scissors`)
   - If one player chooses 'Scissors' and the other chooses 'Paper', the 'Scissors' wins.
     - (`The scissors cuts the paper`)
   - If one player chooses 'Paper' and the other chooses 'Rock', the 'Paper' wins.
     - (`The paper covers the rock`)
   - If both players make the same choice, the game must be played again to determine the winner.

CH. 7 EXERCISE 5 – ROCK, PAPER, SCISSORS MODIFICATION

Write the Algorithm, Pseudocode, Flowchart, and Python Code for the following programming problem:

Scenario: Rock, Paper, Scissors Modification

Ch. 6 Exercise 12 instructed how to design a program that plays the Rock, Paper, Scissors game. In the program, the user enters one of three strings – `Rock`, `Paper`, or `Scissors` – at the keyboard. Add input validation (with a case-insensitive comparison) to make sure the user enters one of those strings only.

Step 1: The Algorithm

1. MODULE 1 – main()
   a. Import Random library function
   b. Call Module 5
2. MODULE 2 – compMove()
   a. Get the computer sign:
      i. Generate a random integer from 1 – 3:
         1. Rock,
         2. Paper
         3. Scissors
3. MODULE 3 – userMove()
   a. Prompt for userMoves
   b. Validate the user input
4. MODULE 4 – winCondition()
   a. Calculate the win condition using nested If-Then-Else structure:
      i. If the user input is the same as the computer, then return a tie message
      ii. Else If the user input beats the computer, then return a win message
      iii. Else If the computer beats the user input, then return a loss message
5. MODULE 5 – rpsGame()
   a. Call Module 2
   b. Call Module 3
   c. Call Module 4
   d. Display the signs and the result:
      i. Display the computer sign
      ii. Display the user sign
      iii. Display the results

The Input, Processing, and Output

Table 2-1 Nested Decision Structure: Win Condition (x, y, z)			
INPUTS	**Input Type**	**Value**	**Data type**
User Move (userSign)	Variable	(a)	Integer
Computer Move (compSign)	Variable	(b)	String
PROCEDURE	$x = [a = b]$ $winCon\ (TIE) = (userSign = compSign)$ $y = [a > b]$ $winCon\ (WIN) = (userSign > compSign)$ $z = [a < b]$ $winCon\ (LOSE) = (userSign < compSign)$		
OUTPUTS	**Output Type**	**Value**	**Datatype**
Win Condition (winConTie)	Constant	(x)	String
Win Condition (winConWin)	Constant	(y)	String
Win Condition (winConLose)	Constant	(z)	String

The IPO for Table 2-1 is as follows:

1. The inputs for Table 2-1 are as follows:
    a. User Move (a)
    b. Computer Move (b)
2. The procedures for Table 2-1 are as follows:
    a. $x = [a = b]$
    $winCon\ (TIE) = (userSign = compSign)$
    b. $x = [a > b]$
    $winCon\ (WIN) = (userSign > compSign)$
    c. $x = [a < b]$
    $winCon\ (LOSE) = (userSign < compSign)$
3. The output for Table 2-1 are as follows:
    a. Tie (x)
    b. Win (y)
    c. Loss (z)

Step 2: The Pseudocode

Refer to Tables 2-1 and 2-2 in Step 1 for the needed variables.

```
1. //This program takes in a choice of rock, paper, or scissors.
2. //Output is then printed to the screen.

3. //Declare the main module
4. //main() calls rpsGame()
5. Module main()

   a. //Call function
   b. Call rpsGame()

6. End Module

7. //Declare the compMove function
8. Function compMoves()
   a. //Declare variable
   b. Declare Integer compMoves

   c. //Calculate compMoves sign
   d. Set compMoves = random(1,3)
   e. If compMoves == 1
      ▪ return 'rock'
   f. Else If compMoves == 2
      ▪ return 'paper'
   g. Else
      ▪ return 'paper'
   h. End If

   i. //Return compMoves
   j. Return compMoves
9. End Function

10. //Declare the userMove function
11. Function userMoves()
   a. //Declare variable
   b. Declare Integer userMoves

   c. //Input sign
   d. Display "Enter Move (Rock, Paper, Scissors)."
   e. Input userMoves

   f. //Validate user input
   g. If userMoves.lower() != 'Rock', 'Paper', 'Scissors'
      ▪ Display "ERROR! Valid Input Includes: Rock, Paper, or Scissors"
   h. Else
      ▪ return userMovers.capitalize()
   i. End If
12. End Function
```

```
13. //Declare the winCondition function
14. Function winCondition()
    k. //Declare variable
    l. Declare String wimCon

    m. //Calculate winning conditions
    n. Set winCon = {'Rock': 'Scissors', 'Scissors': 'Paper', 'Paper':
       'Rock',}
    o. If user == comp
       ▪ return 'TIE: Try again'
    p. Else If wincon[user] == comp
       ▪ return 'USER WINS'
    q. Else
       ▪ return 'COMP WINS'
    r. End If
15. End Function

16. //Declare the rpsGame() function
17. //main() calls compMove() and userMove()
18. Function rpsGame()

    c. //Declare Variables
    d. Declare String compSign
    e. Declare String userSign

    f. //Call function and set to variables
    g. Set compSign = compMove()
    h. Set userSign = userMove()

    i. //Display Output
    j. Display "COMP: ", compSign
    k. Display "USER: ", userSign
    l. Display (wincondition(userSign, compSign))

19. End Function
```