INF1002 Programming Fundamentals - Lab1

Topics:

1. Variables/types/converter
2. Operators
3. Conditional/selection
4. Input/output from the keyboard & data formatting

Warmup exercises:

The following lab assignment requires the use of all topics discussed so far in the module. You may wish to practice some of the concepts with simple exercises before attempting the lab assignment. You are not required to include these exercises in your submission, though you may wish to do so, to help you in the test.

1. Use the Python shell to evaluate the following:

a. `print ('Hello") _____`
b. `print ('Let's Go') _____`
c. `print ("Let's go!") _____`
d. `print ‘hello’ _____`
e. `print ('Hello World\n') _____`
f. `print ('\nHello World') _____`
g. `print ('Hello\nWorld') _____`
h. `print (1, '\n', 2, '\n', 3) _____`
i. `print ('\n', 1, '\n', 2, '\n', 3) _____`

2. Compute the following:

a. `1976 / 10 _____`
b. `1976 // 10 _____`
c. `1976 % 10 _____`
d. `1976.0 / 10 _____`
e. `1976.0/10.0 _____`
f. `1976.0//10 _____`
g. `1976.0%10 _____`
h. `4**3 _____`

3. Perform the following type conversions:

a. `int(5.7) _____`
b. `int(‘5’) _____`
c. `int(‘5.7’) _____`
d. `float(5) _____`
e. `float(‘5’) _____`
f. `float(‘5.7’)_____`

4. Perform the following logic operations:

a. True or False _____
b. True or 0 _____
c. True or 1 _____
d. true and False _____

e. True and True _____
f. True and 0 _____
g. True and 0 _____
h. not True _____
i. not False _____
j. not 1 _____
k. not 0 _____
l. False and 1 _____
m. False or 1 _____
n. False or a=1 _____
o. False or (a=0)

5. Check the ASCII codes of the following characters (Refer to the reading material in the lecture notes):

a. '1' ___
b. '2' ___
c. '3' ___
d. 'a' ___
e. 'b' ___
f. 'c' ___
g. 'A' ___
h. 'B' ___
i. 'C' ___
j. '!' ___
k. '@' ___
l. '#' ___

6. Write an input statement that requests the user to enter a name. Save the name into a variable 'firstName' and print the name (write the lines of code below): HINT- input() is used to input something from keyboard.

_____
_____
_____
_____

7. Write an input statement that requests the user to enter an age. Save the age as an integer in a variable 'age'. Increment the age by one and print the result (write the lines of code below):

_____
_____
_____
_____

8. Write an input statement that requests the user to enter his salary. Save the salary as an integer in a variable 'salary'. If the salary is smaller than 1000, increment the salary by 500 and print the new salary. If the salary is bigger than 5000, decrease the salary by 600 and print the new salary. If the salary is 6000, directly print his salary. Otherwise, increment the salary by 20 and print the new salary. (write the lines of code below and IDLE):

_____
_____
_____

Lab Assignment:

To help you better practice, you need to perform a set of tasks in one auto-grading system, Gradescope in LMS/xSiTe. Gradescope will provide you immediate feedback of your program, such that you will know the issue of your program.

In this lab, you need to finish three tasks in Gradescope system. Gradescope system will provide you immediate feedback about the correctness of your program. You can view the feedback of your system by clicking the view detail beside your grade. Gradescope adopts a test cases-based approach to check your program. Your score is depending on the number of test cases that you program can pass. Note that to train you to have a good programming practice, you have to write your program strictly according to the requirement of the tasks, including your input and output format. If there is any difference (even one space), your program will fail on the test cases. SO, TRAIN YOURSELF TO BE AN EXACT THINKER!

To help you practice, you are allowed to do multiple attempts for each task. Enjoy your learning!

Task 1: Average Calculator

Task Description:

Develop a simple average calculator program. The program requirement is as follows:

1. Allow users to run your program with three input arguments by passing three values to the program: a, b and c.
2. Your program will read the three arguments and calculate the average value.
3. After user inputs all the numbers, if the input numbers are invalid, you need to present an error message "Your input is invalid!". Otherwise, you need to print out the average value. The output average value requires to have 2 precisions. For instance, if the value is 23.456, it should print 23.45. If it is 23, it should print 23.00.

NOTE: You must strictly follow the input and output format.

Assume your program is named as AverageCalculator.py. Example output is as follows:

Running examples:

```
C:\INF1002\Lab1\AverageCalculator>python AverageCalculator.py 3 4 5
Average:4.00

C:\INF1002\Lab1\AverageCalculator>python AverageCalculator.py 60 39 92
Average:63.67

C:\INF1002\Lab1\AverageCalculator>python AverageCalculator.py abc 10 20
Your input is invalid!
```

Instructions to submit and auto grade your code to Gradescope:

1. Download the skeleton code file "AverageCalculator.py" from the "Python\Lab\Lab1" xSite folder.
2. Add your code to the function "*def AverageCalculator()*".
3. Do not change file name or function names.
4. Drag and drop your locally tested code to Gradescope. You can submit and auto graded any number of times. Last submission counts.
5. There are 5 test cases and maximum score for this task is 5.

Task 2: BMI Calculator

Task Description:

In this task, we develop a Body Mass Index (BMI) Calculator that can be used to calculate your BMI value and weight status while taking your age into consideration. Use the "metric units" tab if you are more comfortable with the international standard metric units. The referenced weight range and calculation formula is listed below. The program requirement is as follows:

1. Allow users to run your program with three input arguments by passing three values to the program: the choice of units, height, and weight.
2. Your program will read the three arguments and calculate BMI using the following two formulas:

   $BMI = weight(kg)/height^2(m^2)$ (Metric Units)

   $BMI = 703 \cdot weight(lb)/height^2(in^2)$ (Imperial (U.S.) Units)

   NOTE: The formulas to calculate BMI are based on two of the most used unit systems.

3. After user inputs all the numbers, if the input numbers are invalid, you need to present an error message "Your input is invalid!". Otherwise, you need to print out BMI and category. The output payment requires to have 2 precisions. For instance, if BMI is 23.456, it should print 23.46. If BMI is 23, it should print 23.00.

Reference:

Your BMI is a measurement of your body weight based on your height and weight. Although your BMI does not actually "measure" your percentage of body fat, it is a useful tool to estimate a healthy body weight based on your height. Due to its ease of measurement and calculation, it is the most widely used diagnostic indicator to identify a person's optimal weight depending on his height. Your BMI "number" will inform you if you are underweight, of normal weight, overweight, or obese. However, due to the wide variety of body types, the distribution of muscle and bone mass, etc., it is not appropriate to use this as the only or final indication for diagnosis.

BMI Table for Adults:

This is the World Health Organization's (WHO) recommended body weight based on BMI values for adults. It is used for both men and women, age 18 or older.

Category	BMI range - kg/m^2
Severe Thinness	<= 16
Moderate Thinness	>16 - 17
Mild Thinness	>17 - 18.5
Normal	>18.5 - 25
Overweight	>25 - 30

Obese Class I	>30 - 35
Obese Class II	>35 - 40
Obese Class III	>40

Example output is as follows. Note that *'%0.2f\tSevere Thinness'* should be used.

NOTE: You must strictly follow the input and output format.

Running examples:

```
C:\INF1002\Lab1\BMICalculator>python BMICalculator.py metric 1.80 78
24.07   Normal

C:\INF1002\Lab1\BMICalculator>python BMICalculator.py metric 1.78 48
15.15   Severe Thinness

C:\INF1002\Lab1\BMICalculator>python BMICalculator.py metric 1.60 126
49.22   Obese Class III

C:\INF1002\Lab1\BMICalculator>python BMICalculator.py imperial 68.90 154.32
22.85   Normal

C:\INF1002\Lab1\BMICalculator>python BMICalculator.py imperial 85.63 135.68
13.01   Severe Thinness

C:\INF1002\Lab1\BMICalculator>python BMICalculatorTest.py abc
Your input is invalid!
```

Instructions to submit and auto grade your code to Gradescope:

1. Download the skeleton code file "BMICalculator.py" from the "Python\Lab\Lab1" xSiTe folder.
2. Add your code to the function "*def BMICalculator()*".
3. Do not change file name or function names.
4. Drag and drop your locally tested code to Gradescope. You can submit and auto graded any number of times. Last submission counts.
5. There are 5 test cases and maximum score for this task is 5.

Task 3: Weekly Payment Calculator

Task Description:

Develop one employee weekly payment calculation program as we have discussed in Lecture 2. The program requirement is as follows:

1. Allow users to run your program with three input arguments by passing three values to the program: the number of working hours, input normal rate and input the overtime rate.
2. Your program will read the three arguments and calculate the user's salary using the following two formulas:
    a. *Payment of the normal hours = normal rate * normal hours*
    b. *Payment of the overtime hours = overtime rate * overtime hours*

    NOTE: the working hours within 40 belong to normal hours and those beyond 40 hours are considered as overtime hours.

3. After user inputs all the numbers, if the input numbers are invalid, you need to present an error message "Your input is invalid!". Otherwise, you need to print out the employee's payment of normal hours, his payment of overtime hours and total payment. The output payment requires to have 2 precisions. For instance, if payment is 2300.456, it should print 2300.45. If payment is 2300, it should print 2300.00.

NOTE: You must strictly follow the input and output format.

Assume your program is named as WeeklyPaymentCalculator.py. Example output is as follows:

Running examples:

```
C:\ICT1002\Lab1\WeeklyPaymentCalculator>python WeeklyPaymentCalculator.py
20 30 100
Normal Salary:600.00, Extra Salary:0.00, Total Salary:600.00


C:\ICT1002\Lab1\WeeklyPaymentCalculator>python WeeklyPaymentCalculator.py
60 30 200
Normal Salary:1200.00, Extra Salary:4000.00, Total Salary:5200.00


C:\ICT1002\Lab1\WeeklyPaymentCalculator>python WeeklyPaymentCalculator.py
10000 10 200
Your input is invalid!
```

Instructions to submit and auto grade your code to Gradescope:

1. Download the skeleton code file "WeeklyPaymentCalculator.py" from the "Python\Lab\Lab1" xSiTe folder.
2. Add your code to the function "*def WeeklyPaymentCalculator()*".
3. Do not change file name or function names.
4. Drag and drop your locally tested code to Gradescope. You can submit and auto graded any number of times. Last submission counts.
5. There are 5 test cases and maximum score for this task is 5.

Final submission and auto grading all three tasks together:

<u>After testing all the above three tasks, submit (drag and drop) all three files *AverageCalculator.py, BMICalculator.py* and</u>

WeeklyPaymentCalculator.py together and auto graded for a maximum score of 15 for all the 3 tasks.

Note: It is not necessary to zip these 3 files.