Lab 1 – Functions

Q1: Functions for Numbers [*]

(a) Write a function called `compute_average()` which calculates the average of 3 numbers. Then, outside of the function, prompt the user for three numbers and call the `compute_average()` function.

A sample run of your code is shown below:

```
Enter 1st number: 23.9
Enter 2nd number: 198.3
Enter 3rd number: -95.7
Average: 42.16666666666667
```

(b) Define a function called `compute_geometric_mean()`. This function should take in three numbers and return the geometric mean of the three numbers. The geometric mean of three numbers x, y and z is defined as $\sqrt[3]{x \times y \times z}$. (See https://en.wikipedia.org/wiki/Geometric_mean.) (Hint: $\sqrt[3]{x \times y \times z}$ is the same as $(x \times y \times z)^{\frac{1}{3}}$.)

When you run your test code, you should get the following output:

```
The geometric mean of 2, 4 and 6 is: 3.634241185664279
```

Q2: Message Printer [*]

Write two functions. The first function `print_a_line()` prints a line of dashes. The second function `print_a_message()` prints a given message, starting and ending with a given symbol.

Call these functions to produce the following output:

```
--------------------
| Hello SIS! |
--------------------
--------------------
* Hello Python! *
$ Hello Functions! $
--------------------
```

Q3: Signage Printer [**]

You are now given a new function called `print_a_customized_line()` (download **q3_help.py** from eLearn) that displays a row of a given symbol for a specified number of times. Read the comment above the function to understand how it works.

Note: In the body of this function, you will see a for-loop (from Line 5 to Line 9). This loop basically repeats the action of printing out a single symbol for *n* times. For now, you do not need to understand how this code works. We will learn how to write for-loops in a later week.

Making use of this `print_a_customized_line()` function together with the `print_a_message()` function earlier, can you define a Python function called `print_signage()` that takes in a text message and a symbol, and prints out the message surrounded by the symbol?

For example, if "`Hello!`" and "`*`" are passed to the function, the function should print out

```
**********
* Hello! *
**********
```

If "`Welcome to SMU`" and "`#`" are passed to the function, the function should print out

```
##################
# Welcome to SMU #
##################
```

We have given you some code to test your implementation.

Hint: You will need to figure out how many characters the given message contains in order to figure out how many times you need to print out the symbol in the line above the message and the line below the message. Use the Python built-in function `len()` to help you with this.

Q4: Python Standard Libraries [*]

Let's try to use Python standard libraries to solve some problems.

a) Write a piece of code that prompts the user for the area of a square (in cm^2). The code then displays the side of the square (in cm).

   For example, if the area of a square is 16.0 cm^2, then each side of the square is 4.0 cm.

   A sample run of the code is shown below:

```
What's the size of the square (in square centimeters)? 16
Each side of this square is 4.0 centimeters.
```

   Note: You can use the `sqrt()` function from the Python `math` module.

b) Write a piece of code that prompts the user for a positive integer. (You can assume that the user is always going to input a positive integer.) Call this integer n. The code then displays a random integer between 1 and n (both inclusive).

For example, if the user enters 10, then the code displays a random integer between 1 and 10.

A sample run of the code is shown below:

```
Enter a positive integer: 10
7
```

Note: There are two functions from the Python `random` module that you can use: `randrange()` and `randint()`.

`random.randrange(a, b)` return a random integer *N* such that `a <= N < b`.

`random.randint(a, b)` return a random integer *N* such that `a <= N <= b`.

Q5: Importing a Module [*]

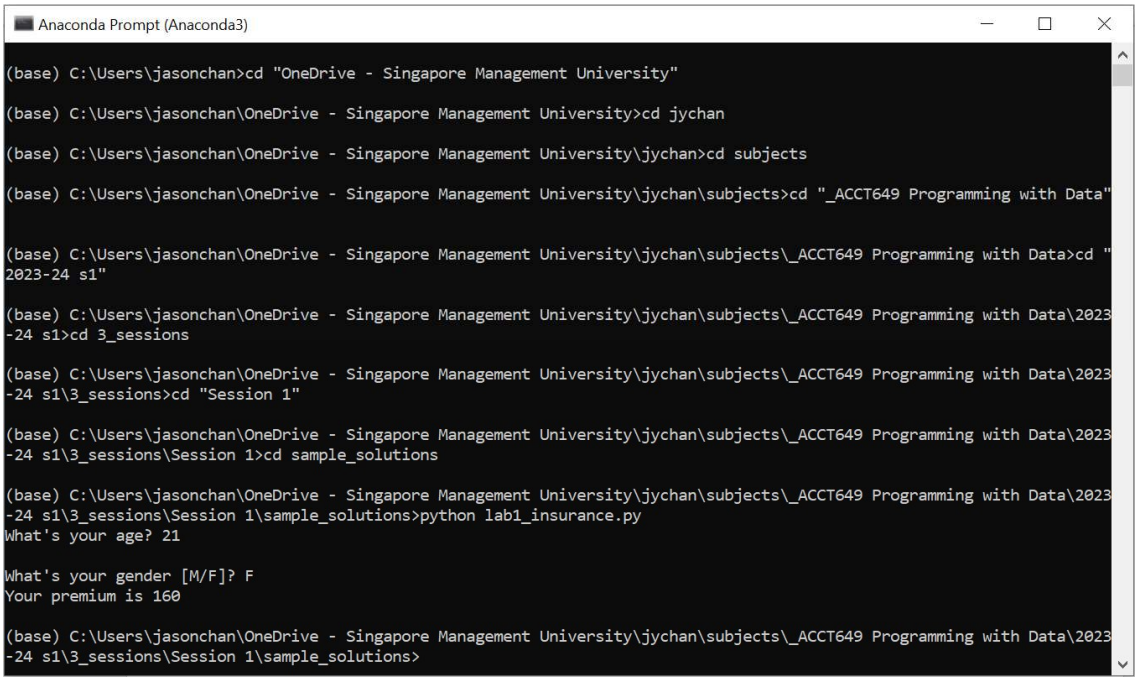
For this question, use Notepad++ to write your code and Anaconda Prompt to run your code.

You are given a Python script called `lab1_utility.py`. Write a Python script called `lab1_insurance.py`. Inside the file, do the following:

a) Prompt the user for his/her age.
b) Prompt the user for his/her gender.
c) Call a function from the given `lab1_utility` module to get the health insurance premium of this person.
d) Print out the amount.

Note: You do not need to understand the implementation of the function given to you. You just need to know how to correctly call the given function.

A sample run of the program should look like the following:



Note that in a new Anaconda Prompt, you will need to use the “cd” command to first go into the folder where your work is saved. Then, type “python lab1_insurance.py” to run your program.

Q6: Time Calculator [**]

Read the following definition of ***system time*** from Wikipedia:

"In computer science and computer programming, system time represents a computer system's notion of the passing of time. In this sense, time also includes the passing of days on the calendar."

In Unix systems, the system time is encoded as the number of seconds elapsed since 1 January 1970 00:00:00 UT. For example, if the system time is 3600, it represents exactly 1 hour since 1 January 1970 00:00:00 UT, because 3600 seconds are equivalent to 1 hour. If the system time is 266460, it represents 3 days, 2 hours and 1 minute since 1 January 1970 00:00:00 UT, because 266460 seconds are equivalent to 3 days, 2 hours and 1 minute. (It can be calculated as follows: $60 \times 60 \times 24 \times 3 + 60 \times 60 \times 2 + 60 \times 1 = 266460$.)

Write a function that is given a system time in terms of number of seconds, and then prints out the numbers of days, hours, minutes and seconds as below:

```
Please enter the system time (in seconds): 266460
Based on this system time, 3 days, 2 hours, 1 minutes and 0 seconds have passed since 1 January 1970 0
0:00:00 UT.
```

Q7: Tax Calculator

In Singapore, an individual’s personal income tax rates are progressive, which means a higher income incurs a higher tax rate. For 2017, the following income tax rates are applied to Singapore residents:

Chargeable Income	Income Tax Rate (%)	Gross Tax Payable ($)
First $20,000 Next $10,000	0 2	0 200
First $30,000 Next $10,000	- 3.50	200 350
First $40,000 Next $40,000	- 7	550 2,800
First $80,000 Next $40,000	- 11.5	3,350 4,600
First $120,000 Next $40,000	- 15	7,950 6,000
First $160,000 Next $40,000	- 18	13,950 7,200
First $200,000 Next $40,000	- 19	21,150 7,600
First $240,000 Next $40,000	- 19.5	28,750 7,800
First $280,000 Next $40,000	- 20	36,550 8,000
First $320,000 In excess of $320,000	- 22	44,550

(To simplify the calculation, we ignore income reliefs, tax rebates, etc.)

For example, suppose a person's annual taxable income in 2017 is $65,000. According to the table above, for the first $40,000, the tax is $550. For the remaining ($65,000 - $40,000 = $25,000) of the taxable income, a tax rate of 7% is applied, which amounts to an additional tax of ($25,000 * 0.07 = $1,750). So the total tax is ($550 + $1,750 = $2,300) for this person.

a) [*] Implement a function called `calculate_tax_1` that takes in a float value which represents a person's taxable income. The function returns the total tax the person needs to pay. In this function, it is assumed that the taxable income passed to the function is **between $20,000 and $30,000** (both inclusive).

   After you implement this function, call `calculate_tax_1(25000.0)` and check whether the function returns 100.0.

b) [**] The function above cannot handle an income value that is below $20,000. To make the function more flexible, implement another function called `calculate_tax_2` that takes in a float value which represents a person's taxable income and returns the total tax the person needs to pay. In this function, it is assumed that the taxable income passed to the function is **between $0 and $30,000** (both inclusive).

After you implement this function, call `calculate_tax_2(25000.0)` and check whether the function returns `100.0`. Also call `calculate_tax_2(10000.0)` and check whether the function returns `0.0`.

Hint: We have not talked about if/else, and you do not need to use if/else in your solution. You can consider using the built-in function `max()` to help you.

c) [***] Now let us further improve the function. Let us implement a third function called `calculate_tax_3` that takes in a float value which represents a person's taxable income. The function returns the total tax the person needs to pay. In this function, it is assumed that the taxable income passed to the function is **between $0 and $40,000** (both inclusive). Note that you need to consider two possible tax rates (2% and 3.5%).

After you implement this function, call `calculate_tax_3(25000.0)` and check whether the function returns `100.0`. Also call `calculate_tax_3(10000.0)` and check whether the function returns `0.0`. Call `calculate_tax_3(35000.0)` and check whether the function returns `375.0`.

Can you further improve the function so that it can handle any amount of taxable income as its argument?

Note: This question can be solved without using if/else. Again, use `max()` to help you with this question.

d) Finally, improve the function again so that it can handle any amount of taxable income. Test it using a variety of inputs.