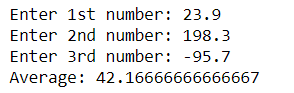
**Week 2 In-Class Exercises**

**Note:** Download the Jupyter notebook “Week 2 In-Class Exercises Starting Code (v1.0).ipynb” (which contains the starting code for the exercises below) from eLearn. Also download the file week2\_utility.py from eLearn.

**Q1: Functions for Numbers [ \* ]**

(a) You are given a function called compute\_average(). Read the docstring of the function to understand what it does. Write a piece of code that prompts the user for three numbers and displays the average of the three numbers by calling compute\_average().

A sample run of your code is shown below:



(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 , and is defined as . (See <https://en.wikipedia.org/wiki/Geometric_mean>.) (Hint: is the same as .)

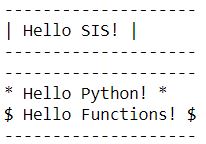
When you run our test code, you should get the following output:



**Q2: Message Printer [ \* ]**

You are given two functions, print\_a\_line() and print\_a\_message(), that are already defined for you. Read the docstrings of the two functions to understand what they do.

Write code to call these functions to produce the following output:



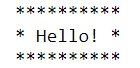
**Q3: Signage Printer [ \*\* ]**

You are now given a new function called print\_a\_customized\_line() that displays a row of a given symbol for a specified number of times. Read the docstring of the function to understand how it works.

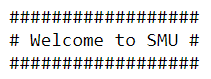
**Note:** In the body of this function, you will see a for-loop (from Line 12 to Line 13). This loop basically repeats the action of printing out a single symbol for *n* times. For now, you do not need to understand how this code works. We will learn how to write for-loops in a later week.

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



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



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: Banknotes and Coins [ \* ]**

You are given a function called calculate\_total\_amount() that has several parameters. See the Jupyter notebook for the function definition.

What do the following function calls return? You are expected to work out the answers without using your computer. Use your computer only to verify your answers.

1. calculate\_total\_amount(2, 3)
2. calculate\_total\_amount(2, 3, 4)
3. calculate\_total\_amount(2, 3, 4, 1)
4. calculate\_total\_amount(2, 3, 4, 1, 3)
5. calculate\_total\_amount(2, 3, two\_note=4)
6. calculate\_total\_amount(2, 3, two\_note=4, five\_note=1)
7. calculate\_total\_amount(3, two\_note=4, five\_note=1)

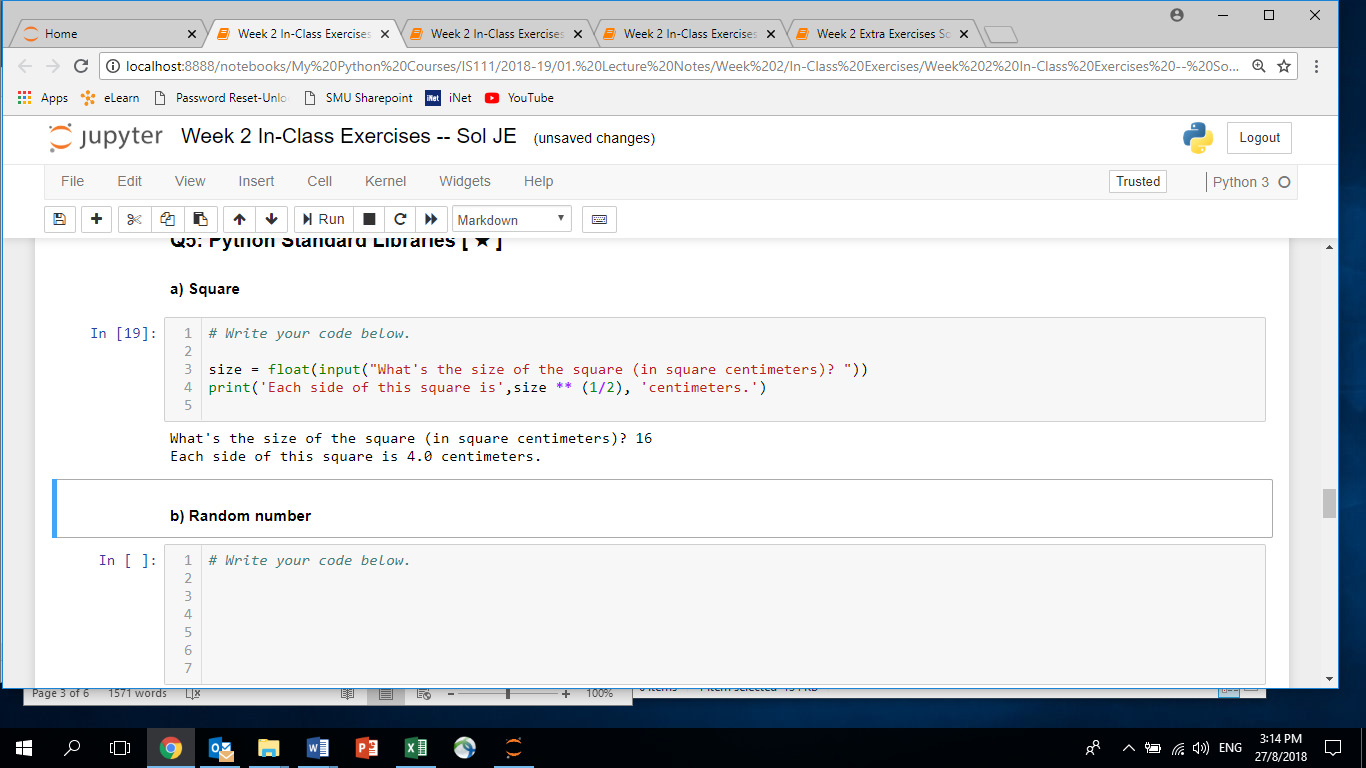
**Q5: Python Standard Libraries [ \* ]**

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

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

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

A sample run of the code is shown below:



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

1. 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 . The code then displays a random integer between 1 and (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:



**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.

**Q6: Importing a Module [ \* ]**

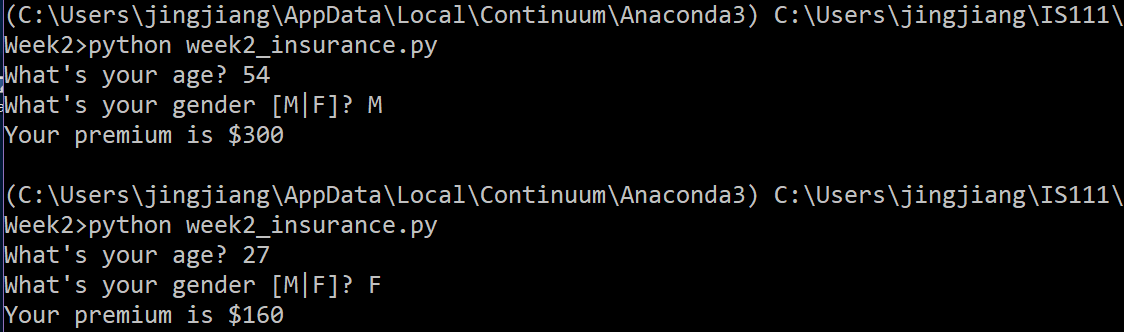
For this question, use Notepad++ to write your code and Anaconda Prompt to run your code.

You are given a Python script called week2\_utility.py. Write a Python script called week2\_insurance.py. Inside the file, do the following:

1. Prompt the user for his/her age.
2. Prompt the user for his/her gender.
3. Call a function from the given week2\_utility module to get the health insurance premium of this person.
4. 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:

**Q7: Time Calculator [ \*\* ]**

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

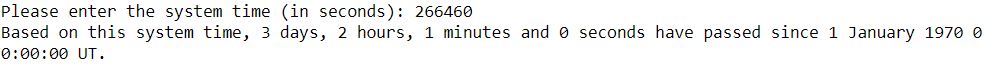
"In computer science and computer programming, system time represents a computer system's notion of the passing of time. In this sense, time also includes the passing of days on the calendar."

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: .)

Given a system time in terms of number of seconds, can you write a piece of Python code to represent it in terms of the numbers of days, hours, minutes and seconds?

We have given you some starting code in the Jupyter notebook. Complete the code in the middle so that the correct numbers are displayed.

A sample run of the code can be found below:



**Q8: Tax Calculator**

In Singapore, an individual’s personal income tax rates are progressive, which means a higher income incurs a higher tax rate. Currently, 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 2021 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.

1. [ \* ] 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.

1. [ \*\* ] The function above cannot handle an income value that is below $20,000. To make the function more flexible, now implement another function called calculate\_tax\_2 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 $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.

1. [ \*\*\* ] 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.