Download and extract the files from the **Lab2\_Starting\_Code.zip**. Save the files into a folder of your choice – you should obtain several \*.py files. Enter your code into the respective \*.py file to complete the exercises.

### Q1 [ \* ]: Code Tracing

Given the following codes in q1.py, could you **predict** what is going to be the **output** of the code without running the code?

You may want to use a **sheet of paper** to note down the various **values** of the **variables** stored in memory as well as what is **displayed** on the **screen** (if applicable).

Run the code in q1.py provided to **check** whether your guess is correct.

|  |
| --- |
| def perform\_magic\_1(x):  x = x + 1    def perform\_magic\_2(x):  x = x + 10  return x    def perform\_magic\_3(y):  return (y + 3)  def perform\_magic\_4(z):  z = z + 1  z = perform\_magic\_2(z)  return (z + 1)  x = 0  print(x)  perform\_magic\_1(x)  print(x)  perform\_magic\_2(x)  print(x)  x = perform\_magic\_3(x)  print(x)  x = perform\_magic\_4(x)  print(x) |

### Q2 [ \* ]: Function Documentation and Function Call

You are given a function display\_numbers in q2.py that displays all the integers ranging from 𝑚 to 𝑛.

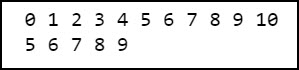
**Note:** You do not need to understand how the function is implemented. In other words, you can ignore Line 14 to Line 16 of the function, but you should read the description of the function, i.e., the docstring.

1. You can use the **help()** function to get the **description** of a function (which is in the triple quotes and which is called **docstring**).  
   help(display\_numbers)has been provided in line 25. Run the file and observe what happens.
2. Uncomment the following code in line 32 and run the file again.

display\_numbers(3, 5)

Is the actual output the same as what you have expected?

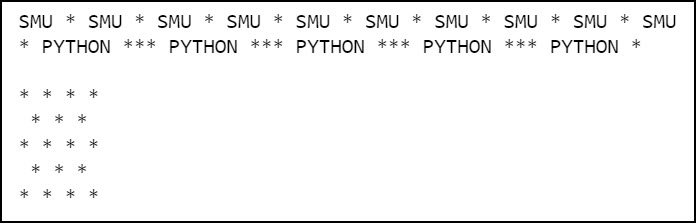
1. Write a piece of Python code from line 36 that calls the display\_numbers function to display the following output:



### Q3 [ \*\* ]: Pattern Printer

You are given a function called print\_pattern in q3.py.  
Note that you **do not need to understand how the function is implemented**, (i.e., you do not need to understand Line 16 to Line 20). We will study it a bit later. However, do read the docstring provided.

Write a piece of **Python code** that uses this function to display the following output:



### Q4 [ \*\* ]: Interest Calculator

So far you have practiced how to use a given function. In this exercise you will practice **how to define** your **own function**.

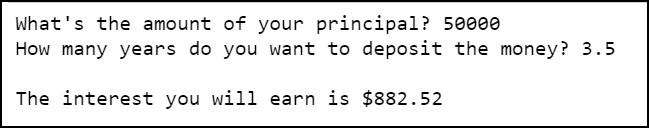
Please refer back to ***Lab1 - Q5*** and define a function called calculate\_interest that takes in 4 parameters:

* principal: the amount of principal that's deposited.
* annual\_interest\_rate: the annual interest rate.
* frequency\_of\_compounding: the number of times the interest is compounded per year.
* deposit\_period: the deposit time (in years)

and returns the **total interest** earned by the end of the deposit period, **rounded** with 2 decimal places. (Use the built-in function round(x, 2) to round x to 2 decimal places.)

Next, write a piece of code that prompts the user for his/her **principal** and the **deposit time**, and use the calculate\_interest  function that you have written to help you **calculate** and **display** the **interest earned** at the end of the deposit period.

Your code should produce the following output (similar to previous lab1):



### Q5: Shopping Cart

**Part 1 [ \*\* ]**: Implement a function in q5.py called calculate\_price\_after\_discount that takes in the following parameters:

* `unit\_price`: the unit price of an item
* `quantity`: the quantity of that item
* `discount`: the discount rate offered by the store for that item (e.g. 10 % off...)

and returns how much the customer has to pay for buying that particular quantity of items (after discount).

For example, if the unit price of a ruler is **$1.50**, the quantity is **5** and the discount is **10%**, then the final price to pay will be 1.5 × 5 × (1 – 10 / 100), which is **$6.75**.

**Part 2 [ \* ]**: Given the following items in the shopping cart with their unit price, quantity and discount, call your function in Part 1 and calculate the value of the **total payment after discount**:

* milk: $5.95, 2, 10%
* rice: $6.50, 1, 5%
* eggs: $2.40, 2, 0%
* kaya: $3.95, 3, 15%

Your code should produce the following sample output:



### Q6 [ \* ]: Phone Bill

You're given the function compute\_phone\_bill in q6.py that computes the **amount on a monthly mobile phone bill**. The function takes the following parameters:

* `base`: the base premium
* `data\_limit` (in Gb): the monthly limit of data usage
* `amount\_data` (in Mb): the actual amount of data used for the month
* `num\_extra\_sms`: the number of extra SMS sent, with default set to 0
* `num\_minutes\_extra\_calls`: the number of minutes of extra outgoing local calls, with default set to 0

Now the business rules for computing the monthly bills are as follows:

* For each 1Mb of extra data usage, the charge is $4.5
* For each extra SMS, the charge is $0.05
* For each extra minute, the charge is $0.15

Based on the function compute\_phone\_bill provided, can you figure out the returned **values** of the following **function calls** without running the code?

Run the code in q6.py to **check** your answers.

1. compute\_phone\_bill(35.5, 2, 800)
2. compute\_phone\_bill(35.5, 3)
3. compute\_phone\_bill(22.5)
4. compute\_phone\_bill(35.5, data\_limit=2, amount\_data=800)
5. compute\_phone\_bill(base=35.5, data\_limit=2, amount\_data=800)
6. compute\_phone\_bill(data\_limit=2, amount\_data=800, base=20)
7. compute\_phone\_bill(35.5, amount\_data=800, data\_limit=2)
8. compute\_phone\_bill(amount\_data=800, data\_limit=2)
9. compute\_phone\_bill(32, 2, 800)
10. compute\_phone\_bill(35.5, 2, 2050)
11. compute\_phone\_bill(35.5, 2, 1900, 10, 20)
12. compute\_phone\_bill(35.5, 2, num\_minutes\_extra\_calls=100)
13. compute\_phone\_bill(35.5, 2, num\_minutes\_extra\_calls=0=100, num\_extra\_sms=100)

### Q7 [ \*\* ]: String Manipulation

* In “lab\_2\_util.py”, there is a function called “get\_common\_characters” that takes in two strings and returns a string that contains the common characters that the two strings share.
* You are also given the starting code of a script called “q7.py”. Inside the file, there are two functions that need to be implemented: q7\_1 and q7\_2.

Read the instructions provided in the docstring of these functions to know what is expected from those functions.

When q7.py is run, the following output is expected:

