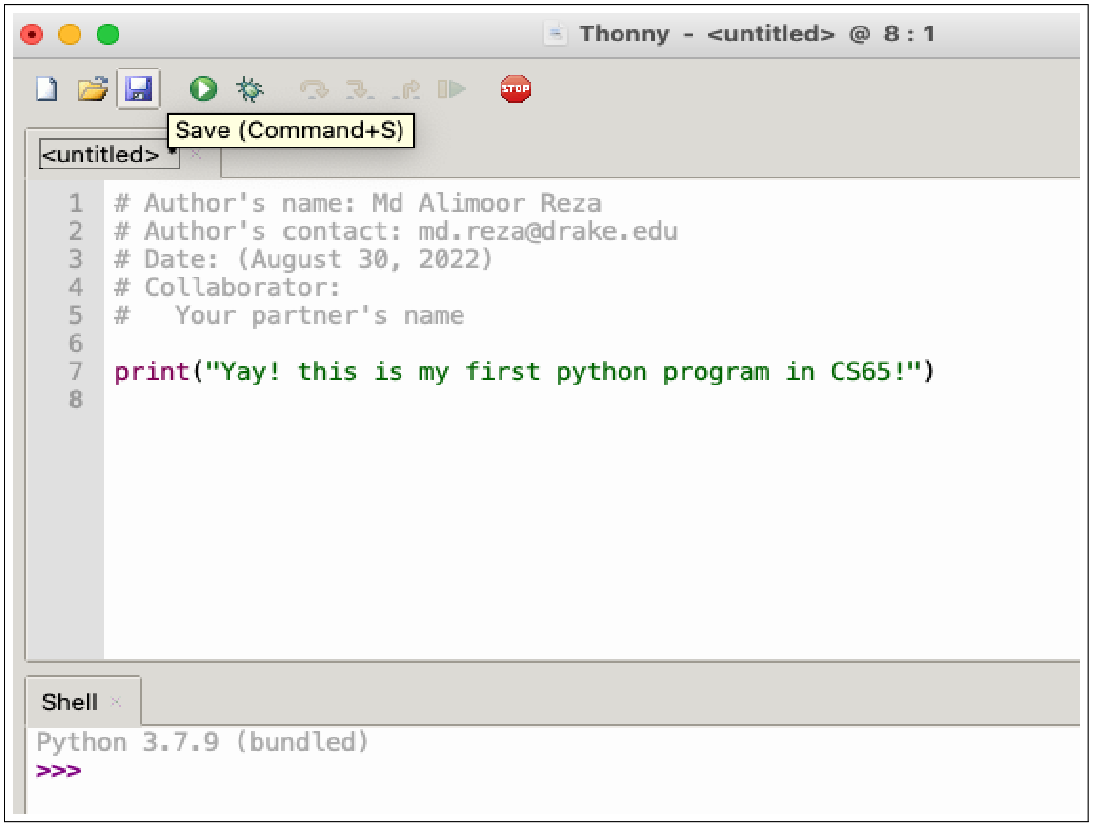
Lab 2: Playing with functions and variables

Course: CS65 - Introduction to Computer Science (Fall 2022)
Instructor: Md Alimoor Reza, Assistant Professor of Computer Science, Drake University
Due: Saturday, September 17, 11:50 PM

Playing with functions and variables

You will be creating a python file for each task separately (with a **.py* extension), where you will save your python instructions. By default, Thonny opens an unnamed file in the a *text editor* (top text pane). Writing a formal header and suppressing it as comments is an excellent practice. A line of code with a preceding # is considered comment in Python. You should type in the necessary parts as shown below, e.g., *your name, your contact email, description, etc.*. Once you finish typing in, click on the *Save* button as shown below.



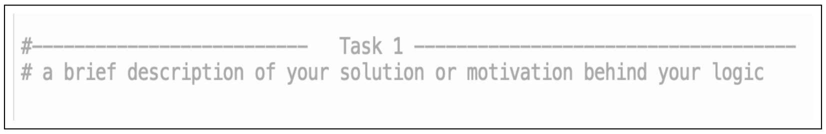
```
# Author's name: Md Alimoor Reza
# Author's contact: md.reza@drake.edu
# Date: (August 30, 2022)
# Collaborator:
#   Your partner's name

print("Yay! this is my first python program in CS65!")
```

```
Python 3.7.9 (bundled)
>>>
```

Now, save the file using the format as follows *firstname_lastname_lab2_task1.py* (all in lowercase letters). For example, I saved my file as *md_reza_lab2_task1.py*. You should follow the same file naming format for the rest of the semester. You will be prompted to save the file in a specific directory. For example, I saved it in a new folder called 'cs65' in the 'Desktop.' Feel free to save it in other places of your choice.

Driver and navigator: Now, you will be doing collaborative work for this lab. These exercises should be completed with your assigned partner. Both of you will take turns playing the role of i) the driver and ii) the navigator. The driver's responsibility is to type into Thonny when completing each exercise. The navigator's responsibility is to support the driver by reading the exercises, looking up relevant readings, and identifying errors.

Tasks and file submission: This lab consists of several small tasks. We expect that you submit separate file for each task (a python file with a *.py* extension). In your python file, add comments to briefly explain what you are doing. For example you could just write:



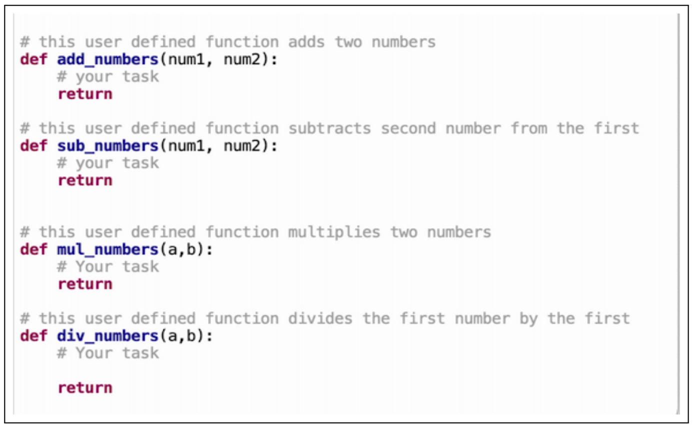
```
#---------------------------  Task 1 ------------------------------------
# a brief description of your solution or motivation behind your logic
```

You should submit your individual copy by acknowledging your partner's name at the beginning of your python file (inside a comment); if you could manage to finish it during class, that great! If you cannot complete the lab during class today, you can complete it individually (or together) sometime before next Tuesday.

Functions

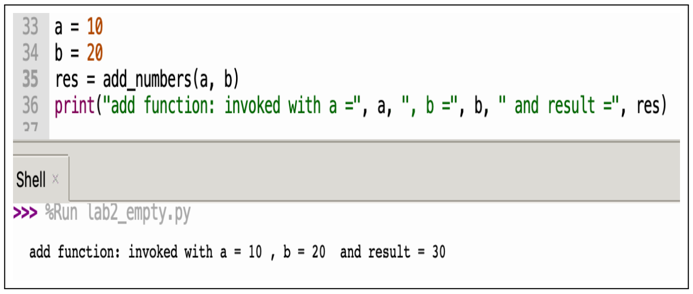
Task 1: writing simple arithmetic operations with functions

In this task, you will be implementing a python program that calculates simple arithmetic operations such as *add, subtract, multiply,* and *divide* between two numbers. Instead of writing a line of expression in your python file, you should create a separate function for each of the above-mentioned operations. For example, here are the empty function headers for the four arithmetic operations. Your goal is to finish the function body with Python *expressions* as necessary. Recall that a fragment of Python code, which calculates a new value, is called an *expression*.



```python
# this user defined function adds two numbers
def add_numbers(num1, num2):
    # your task
    return

# this user defined function subtracts second number from the first
def sub_numbers(num1, num2):
    # your task
    return


# this user defined function multiplies two numbers
def mul_numbers(a,b):
    # Your task
    return

# this user defined function divides the first number by the first
def div_numbers(a,b):
    # Your task

    return
```

Once you **define** a function, you need to **call** them in order to see its execution. For example, if the definition is complete for the function *add_numbers(...)*, I could the **call** the function with the following python lines. Hit the Run button to execute your program and see the outputs.



```python
a = 10
b = 20
res = add_numbers(a, b)
print("add function: invoked with a =", a, ", b =", b, " and result =", res)
```

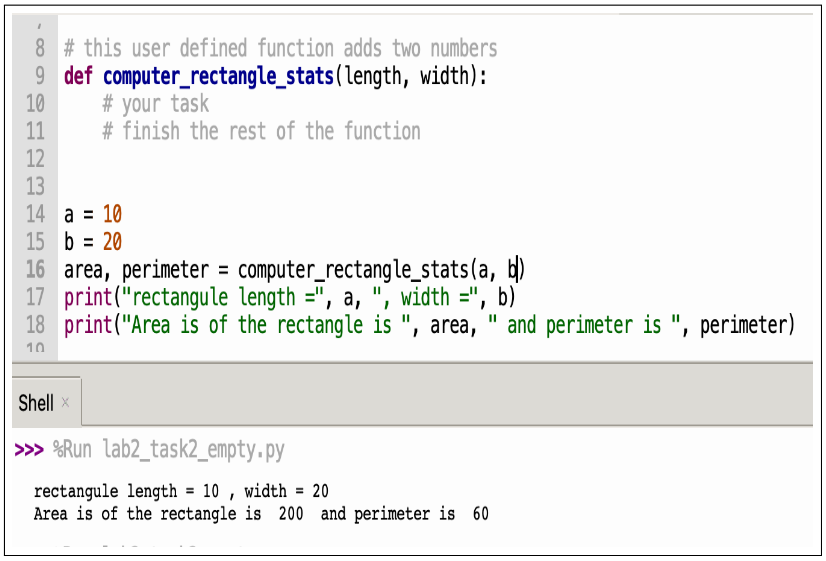
```
Shell
>>> %Run lab2_empty.py

 add function: invoked with a = 10 , b = 20  and result = 30
```

Define the four functions and call them as instructed. You should call the functions multiple times with different values.

Task 2: writing functions for different shapes

In this task, you will be computing various statistics of some basic shapes e.g., *rectangle, square, triangle* and *circle*. More specifically, you should compute two statistics for each shape: 1) area and 2) perimeter. Also you could assume that for each shape necessary *parameters* to calculate these two statistics are given. In order to **define** a function for *rectangle*, you will need two *parameters* namely its *length* and *width*. Similarly, in order to **call** your function, you can instantiate two *variables* (of course you should give them *meaningful names*). Here are the steps you should follow:

- **Step 1:** First, **define** the function with a meaningful name and necessary *parameters*.
- **Step 2:** finish *area* and *perimeter* computation.
- **Step 3:** return the results (you may need to return two variables).
- **Step 4:** finally, **call** the function with appropriate *arguments*.



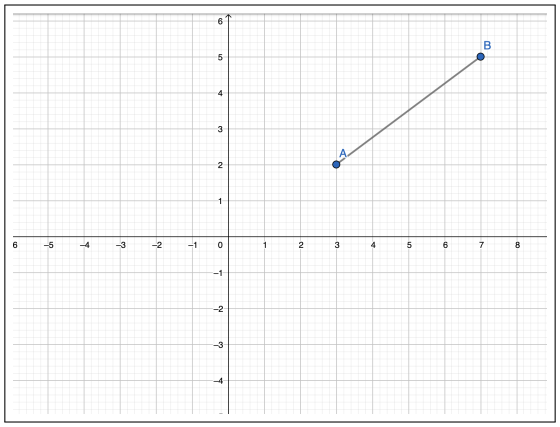
```
# this user defined function adds two numbers
def computer_rectangle_stats(length, width):
    # your task
    # finish the rest of the function


a = 10
b = 20
area, perimeter = computer_rectangle_stats(a, b)
print("rectangule length =", a, ", width =", b)
print("Area is of the rectangle is ", area, " and perimeter is ", perimeter)
```

```
>>> %Run lab2_task2_empty.py

  rectangule length = 10 , width = 20
  Area is of the rectangle is  200  and perimeter is  60
```

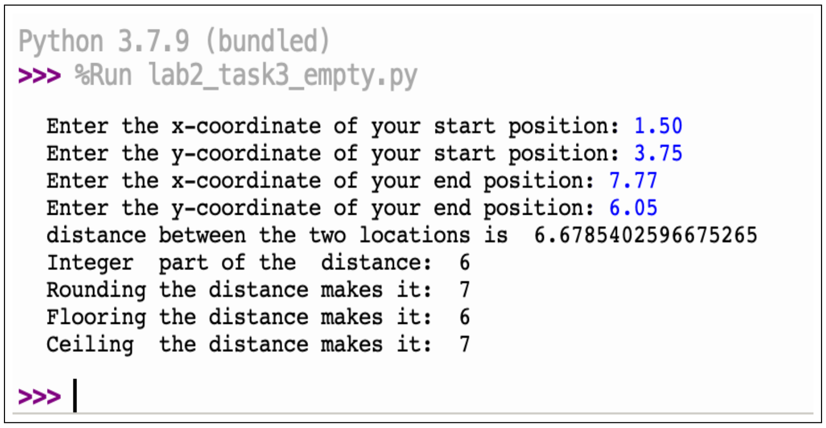
Now, finish the above steps for remaining three shapes for *square, triangle*, and *circle*. First, determine how many *parameters* you will need for each shape. For example, for *circle* you will need only one *parameter*, which is the radius of the circle. Then finish the function *definition* and *calling*.
Define the four functions and call them as instructed. You should call your functions multiple times with different values.

Task 3: How far is Hubbell Dining Hall?

Imagine you have a device that can give your location inside the Drake campus. You can record your locations in various places on campus with this device. For example, you recorded your location at **Collier-Scripps Hall** first. Then you headed towards **Hubbell Dining Hall**. Once you reached there, you recorded your location one more time. The locations are defined in a 2D coordinate system. For example, two locations are shown in the figure below. Location A has a x-coordinate of 3 units and y-coordinate of 2 units; hence A can be denoted by (3,2). Similarly, location B has a x-coordinate of 7 units and y-coordinate of 5 units; hence B can be denoted by (7,5). The straight line distance between them is **5 units**. Your goal for this task is to calculate the straight line distance between two locations defined in a 2D coordinate system, as discussed before. Additionally, you also need to show various other quantities of the calculated distance.



This task is more open-ended! Unlike the previous two tasks, you are not given any initial empty function header or python code for calling it. The goal is to write Python code from scratch. You should get the 2D coordinates of the start location from the user using built-in *input()* function. Similarly, also get the 2D coordinates of the end location from the user. Besides showing the distance with fractions, you also need to show the following outputs. By using various mathematical operations such as **floor(), ceiling(), round()** you either need to chop off the fractional part of the distance or add in an additional fraction to make it an integer number. You have used several other built-in functions by now, such as *print(), input(), int(), etc.*. There are several other Python built-in functions in **math** library. You may find it useful to use them.



```
Python 3.7.9 (bundled)
>>> %Run lab2_task3_empty.py

  Enter the x-coordinate of your start position: 1.50
  Enter the y-coordinate of your start position: 3.75
  Enter the x-coordinate of your end position: 7.77
  Enter the y-coordinate of your end position: 6.05
  distance between the two locations is  6.6785402596675265
  Integer  part of the  distance:  6
  Rounding the distance makes it:  7
  Flooring the distance makes it:  6
  Ceiling  the distance makes it:  7

>>> 
```

Define the necessary functions and call them. You should call the functions multiple times with different values.

What to turn in:

Please submit a zip file containing your source code (three python files with *.py* extension with proper naming convention described above). Please make sure that your code runs prior to submission. Log in to Blackboard and find Lab 2 and submit your zip file like Lab 1.