

Lab 1: Getting started with Thonny

Course: CS65 - Introduction to Computer Science (Fall 2022)

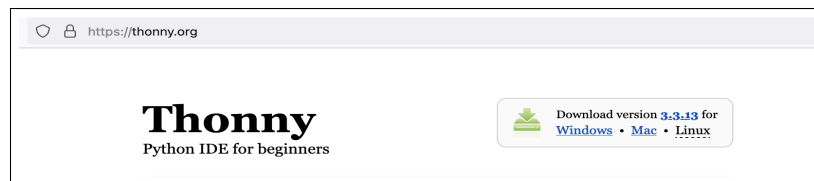
Instructor: Md Alimoor Reza, Assistant Professor of Computer Science, Drake University

Due: Thursday September 08, 11:50 PM

Part 1: Installing our favorite IDE ‘Thonny’

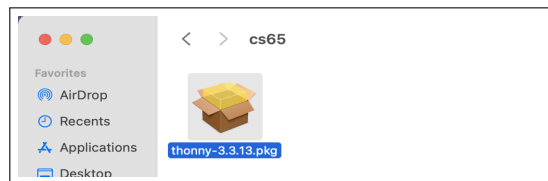
An IDE – Integrated Development Environment – is a tool that programmers use to create, run, and test new programs. It contains a *text editor*, a *compiler/interpreter* for translating the code into machine understandable instructions, and an *executable environment* for showing the result of the program. We will be using the Python programming language along with an IDE for creating Python programs. We recommend *Thonny* as an IDE which is a very user friendly tool and freely available online.

Step 1: Download and installation: Open your favorite web browser, eg, *Mozilla Firefox*, *Google Chrome*, etc. and go to the website <https://thonny.org>. On the upper-right corner you will notice the download link as shown below. Click on the *Mac* button and save the **.pkg* file.

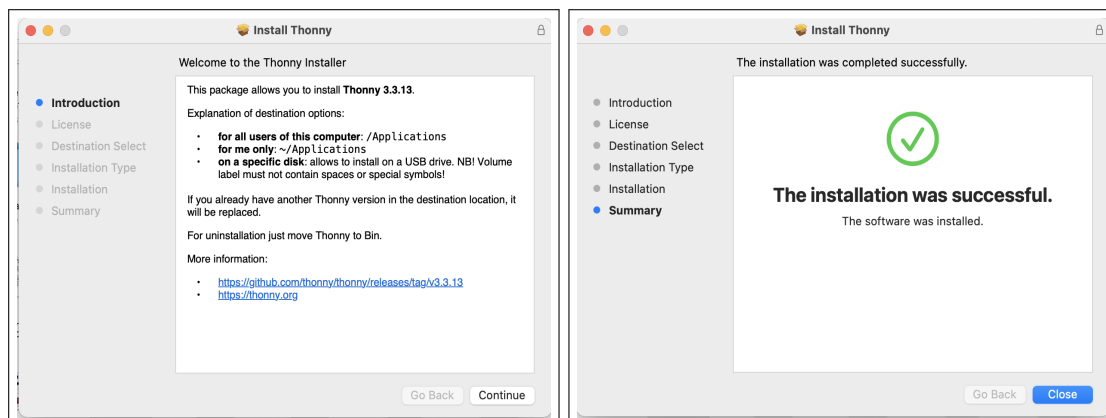


Click on the *Download* button for Mac

Double-click on the *'thonny-3.3.13.pkg'* and it will pop up a window for installation. Follow the steps as instructed. To help you guide the installation process, we are attaching the relevant snapshots during the installations.

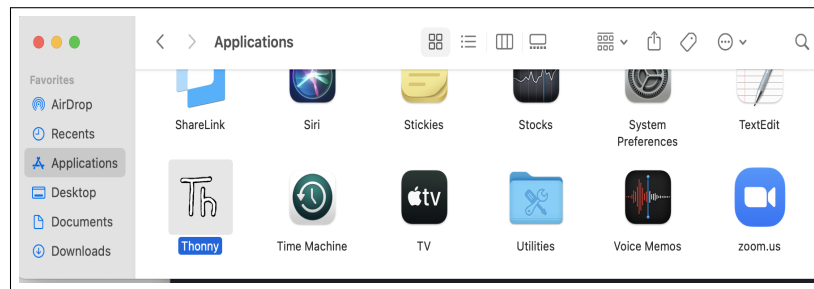


Double-click on the *'thonny-3.3.13.pkg'*.



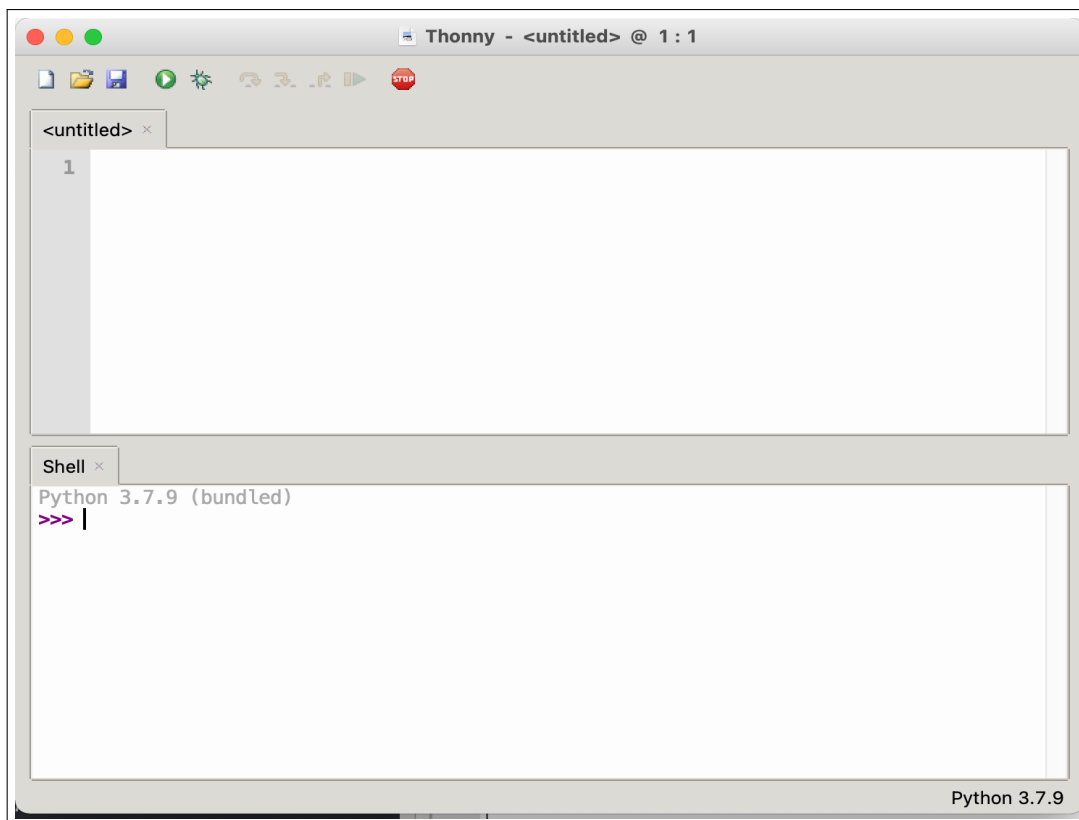
Follow the installation instruction as shown above.

Step 2: Opening up Thonny If you correctly follow the installation process, Thonny will be installed in your *Applications* folder as shown below.

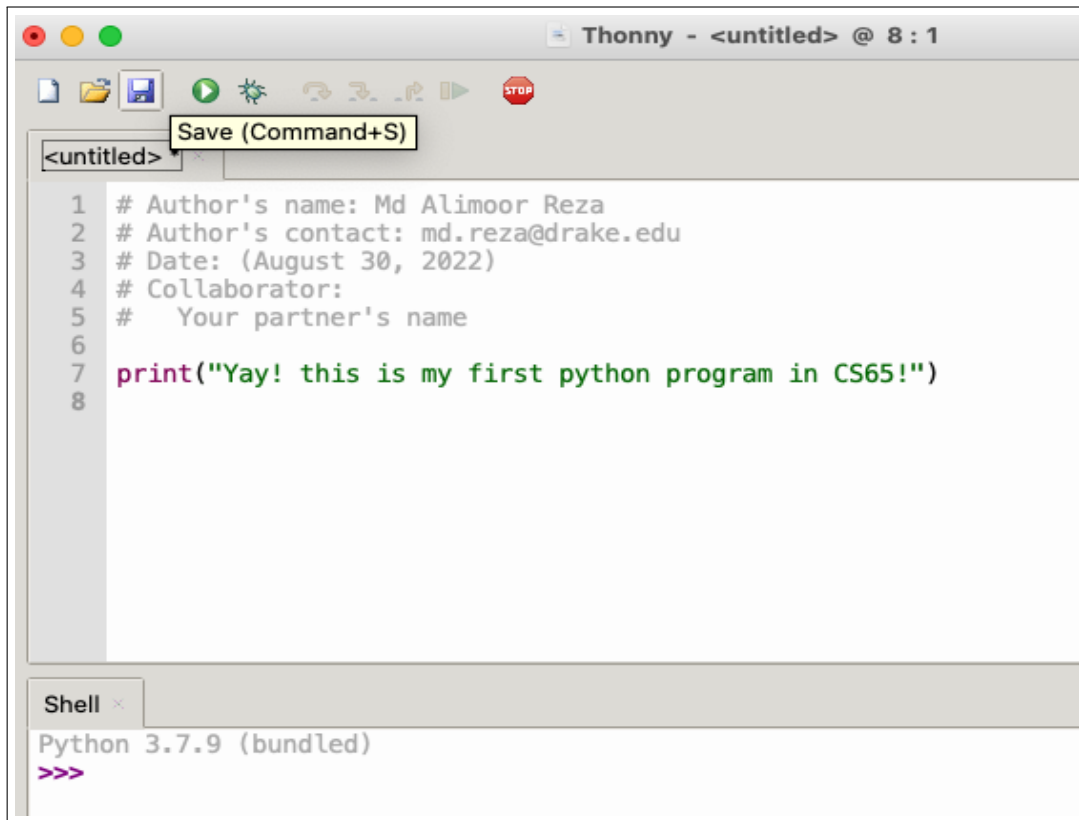


Click on the *Thonny* icon to launch it.

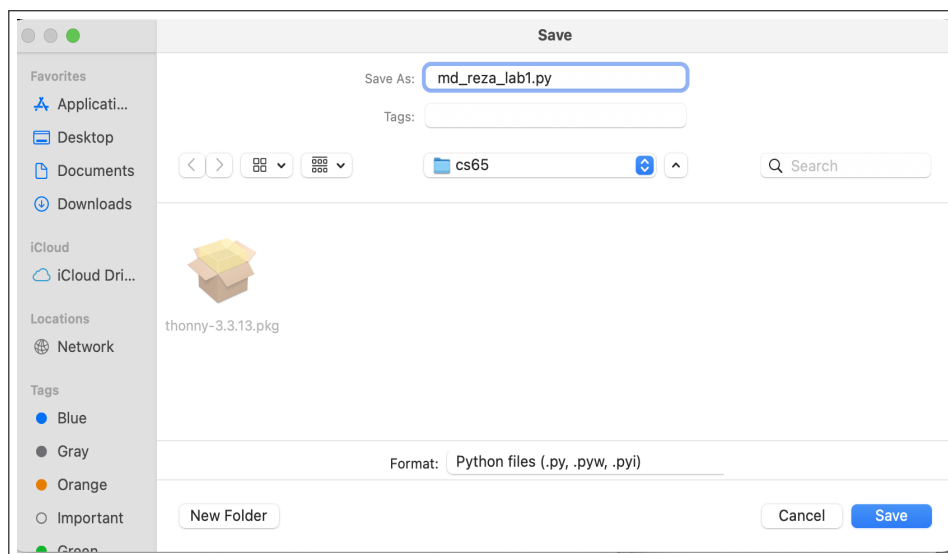
Finally, we will be able to see the Thonny interface as shown below. We will be writing all our assignments, labs, and other practice programming in Python here. *Yay!!! We installed it successfully and ready to write program in Python!.*

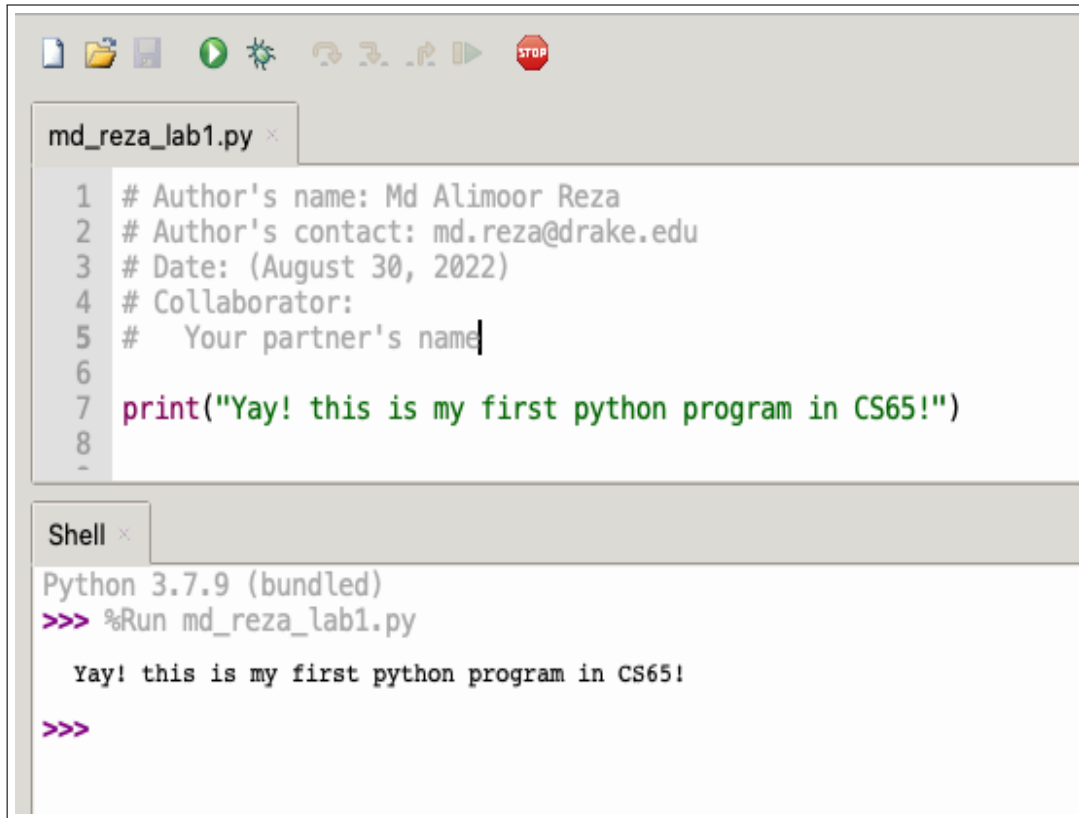


Step 3: Writing your *first* python program in Thonny We will be writing a very simple program. We will create a python file (with a *.py* extension) where our python instructions will be saved. By default, Thonny opens an unnamed file in the a *text editor* (top text pane). It is a very good practice to write a formal header and suppress it as comments. A line of code with a preceding *#* is considered as comment in Python. You should type in the necessary parts as shown below eg, *your name, your contact email, description, etc..* Once you finish typing in, click on the *Save* button as shown below.



Now, save the file using the format as follows *firstname_lastname_lab1.py* (all in lowercase letters). For example, I saved my file as *md_reza_lab1.py*. To maintain consistency, we will 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. Hit the 'Run' button as shown below to execute your program. If everything goes well, in the 'Shell' menu (the bottom text pane), you will see the line *"Yay! this is my first python program in CS65!"* as shown below.





The screenshot shows the Thonny IDE interface. At the top, there is a toolbar with icons for file operations, running, and stopping. Below the toolbar, a tab labeled 'md_reza_lab1.py' is open, displaying the following Python code:

```
1 # Author's name: Md Alimoor Reza
2 # Author's contact: md.reza@drake.edu
3 # Date: (August 30, 2022)
4 # Collaborator:
5 #   Your partner's name
6
7 print("Yay! this is my first python program in CS65!")
8
-
```

Below the code editor, a 'Shell' tab is open, showing the execution of the script. The shell prompt is 'Python 3.7.9 (bundled)'. The user has entered the command '%Run md_reza_lab1.py', and the output is 'Yay! this is my first python program in CS65!'. The shell prompt '>>>' is visible again, indicating the end of the execution.

Please, take a screenshot of this last step so that we could verify that you have successfully executed your first python program.

Part 2: Playing with Thonny

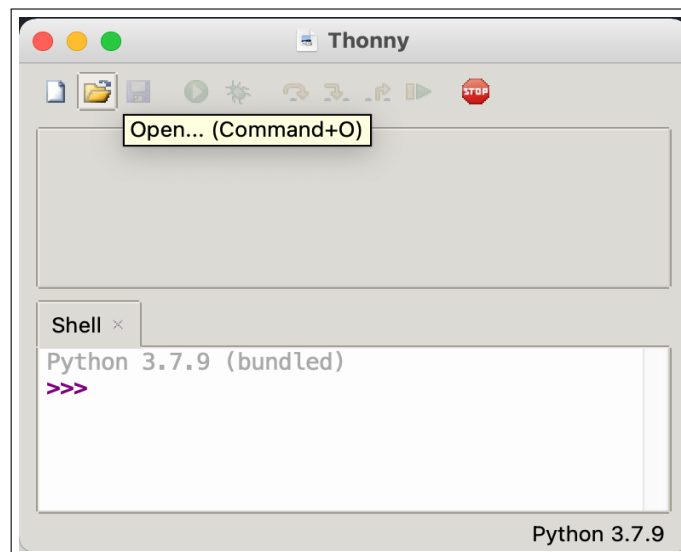
Driver and navigator: Now, you will be doing collaborative work for this lab. These exercises should be completed with your assigned partner. Both of you will take turns playing the role of i) the driver and ii) the navigator. The driver's responsibility is to type into Thonny when completing each exercise. 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 a file (a python file with a `.py` extension) that should have answers to each of these tasks. In your python file, denote the tasks using comments. For example you could just write:

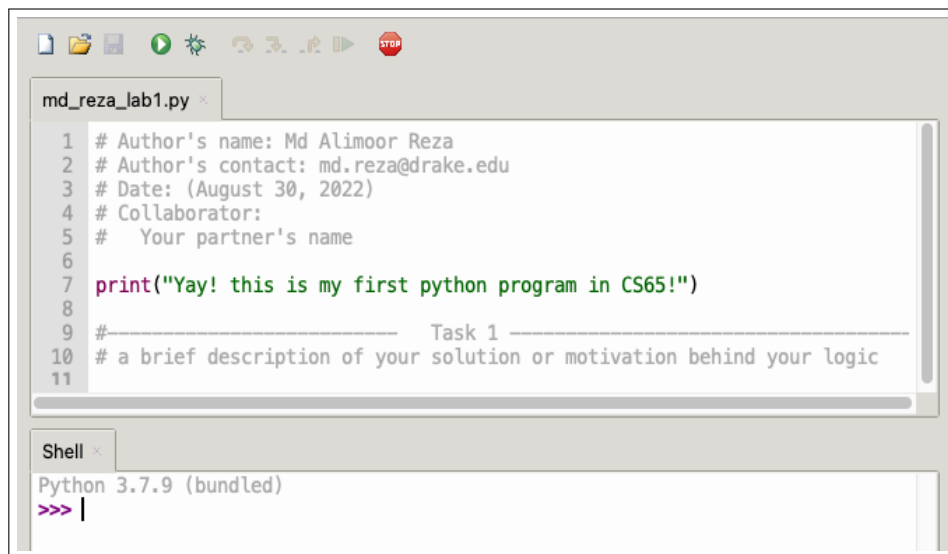
```
#----- Task 1 -----  
# a brief description of your solution or motivation behind your logic
```

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.

Task 1: arithmetic operations Open up the file you have created in Part 1 of this lab. You could use the Open button in Thonny to locate your saved file.



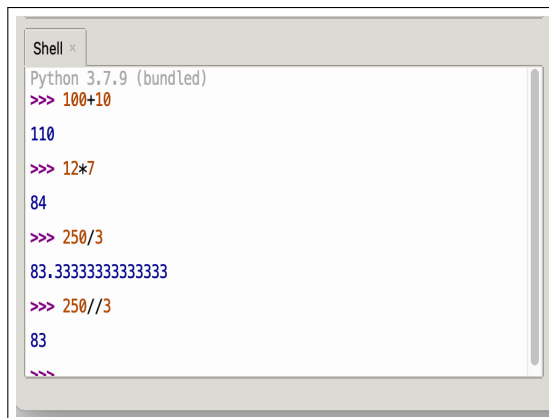
Then modify the comment section in your file with your partner's name as follows:



```
md_reza_lab1.py x
1 # Author's name: Md Alimoor Reza
2 # Author's contact: md.reza@drake.edu
3 # Date: (August 30, 2022)
4 # Collaborator:
5 # Your partner's name
6
7 print("Yay! this is my first python program in CS65!")
8
9 #----- Task 1 -----
10 # a brief description of your solution or motivation behind your logic
11

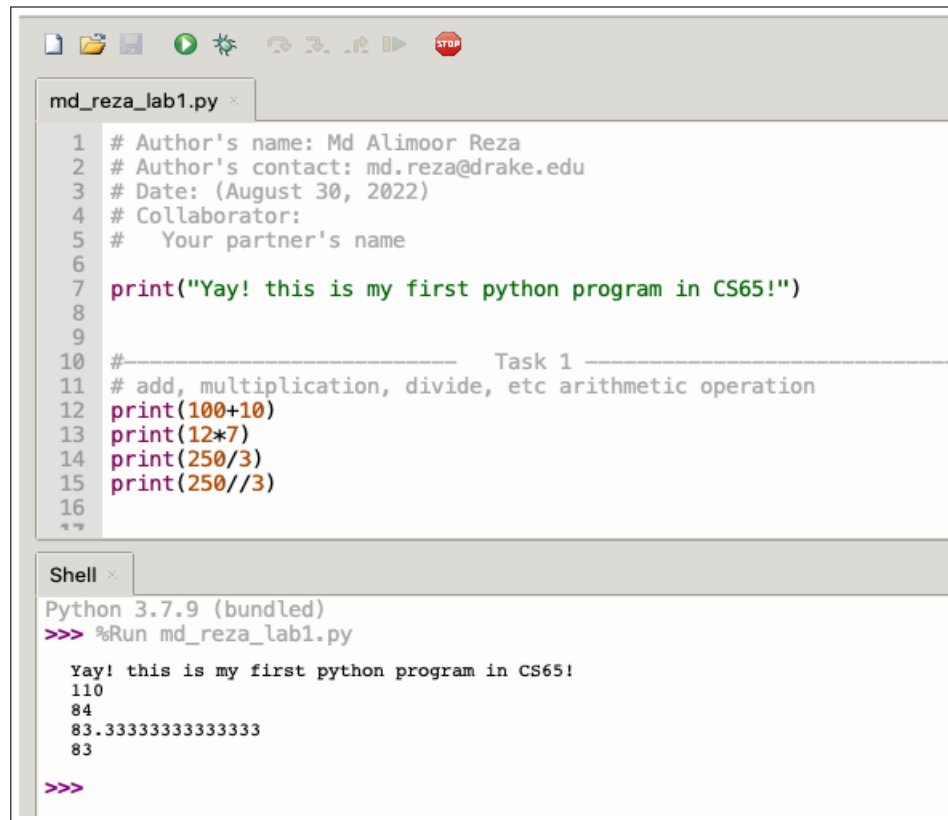
Shell x
Python 3.7.9 (bundled)
>>> |
```

In the Python shell environment, let's do some arithmetic operations such as addition, multiplication, division, etc. The shell environment allows you to play interactively with Python. It is a Read, Evaluate, Print, Loop (REPL). Every line you type into the shell is immediately executed, and any value it produces is then printed to the display. I tried the following operations.



```
Shell x
Python 3.7.9 (bundled)
>>> 100+10
110
>>> 12*7
84
>>> 250/3
83.33333333333333
>>> 250//3
83
>>>
```

Try out these operations with different numbers of your choice and see what happens. In your python file, do the same operations but instead of typing them, put the operations inside `print()` function as follows:



The screenshot shows the Thonny IDE interface. The top toolbar contains icons for file operations, running, and stopping. The main editor window displays a Python script named 'md_reza_lab1.py' with the following code:

```
1 # Author's name: Md Alimoor Reza
2 # Author's contact: md.reza@drake.edu
3 # Date: (August 30, 2022)
4 # Collaborator:
5 #   Your partner's name
6
7 print("Yay! this is my first python program in CS65!")
8
9
10 #----- Task 1 -----
11 # add, multiplication, divide, etc arithmetic operation
12 print(100+10)
13 print(12*7)
14 print(250/3)
15 print(250//3)
16
```

Below the editor is a 'Shell' window titled 'Python 3.7.9 (bundled)'. It shows the command prompt with the command '%Run md_reza_lab1.py' and the following output:

```
>>> %Run md_reza_lab1.py
Yay! this is my first python program in CS65!
110
84
83.33333333333333
83
>>>
```

Hit the [Run](#) button to execute your program and see the outputs. Explain the difference between the two division operations. You could do this as a comment in your python file. Recall that you can add a comment in your python file with the help of a `#` symbol. Try out arithmetic operations with two more operators: i)% and ii)**. Observe the results and find out the functions of these two operators.

Task 2: variables and expressions You can create a variable through the use of an assignment operator `=`. In other words, you can store something (numbers or textual data) in *variables*; later use them for other operations. A sequence of characters is referred to as a *string*. A string is denoted by being enclosed by double quotes `" "` or single quotes `' '` in Python. Let's initialize some *variables* with different numbers and strings.

```

18 #----- Task 2 -----
19 # variables
20
21 time_sec = 60
22 temp_degree = 27
23
24 mile_to_kilometer = 1.609
25 price_in_dollars = 1500.89
26
27 first_name = 'Fiona'
28 last_name = "Johnson"
29

```

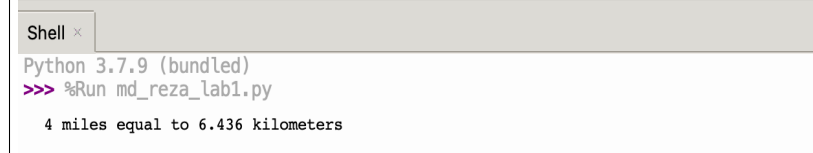
Type these lines in your Python file and initialize the variables accordingly.

You can *print* the *variables* to see contents stored inside them. A fragment of Python code that calculates a new value called an **expression**. In the code above, for example, you can convert miles into kilometers using the following **expression**:

```

30 num_of_miles = 4
31 num_of_kilometers = num_of_miles*mile_to_kilometer
32 print(num_of_miles, "miles equal to", num_of_kilometers, "kilometers")
33

```



```

Shell x
Python 3.7.9 (bundled)
>>> %Run md_reza_lab1.py

4 miles equal to 6.436 kilometers

```


Python evaluates this **expression** as follows. The multiplication operator `*` has been used to calculate a new value. The values stored inside the two variables *num_of_miles* and *mile_to_kilometer* are retrieved back during the multiplication process. Finally, we initialized a new variable *num_of_kilometers* with the newly calculated value.

Follow the same instructions in your python file and replicate the result. Now create a new variable *full_name* so that it is initialized with the full name? You should use the existing *variables* *first_name* and *last_name* in order to accomplish this goal so that when you print *full_name* you the following output:

```

37 print(full_name)
38
39

```



```

Shell x
Python 3.7.9 (bundled)
>>> %Run md_reza_lab1.py

Fiona Johnson

```

Hints: You can also apply the operator `+` in between *strings*. It will concatenate the two *strings*. Within the print statement, use commas between the variables and strings.

Task 3: receiving input from the user The built-in function *print()* shows the output in the shell environment. There is another built-in function in Python that allows us to receive input from the user. You can invoke the following line of Python code to accomplish this:


```

41 #----- Task 3 -----
42 # user input
43
44 height = input("Enter the height of the wall (in inches): ")
45 print("The wall is ", height, "inches tall")
46
47

```

Shell x

Python 3.7.9 (bundled)

```

>>> %Run md_reza_lab1.py

```

Enter the height of the wall (in inches): 60
The wall is 60 inches tall

```

>>> |

```

Type in the above two lines in your python file and try to receive different numbers from the user. You need to press return in order to see the output in the shell.

Now try to do arithmetic operations on the entered number. Let's try to convert the unit of the height *variable* from inches to feet. If you try to directly manipulate the variable *height* with an arithmetic operator, you will encounter an error message. The variable *height* is a *string* and you can't divide a string. Hint: convert the variable to a number first. Your output should look like this:

Shell x

Python 3.7.9 (bundled)

```

>>> %Run md_reza_lab1.py

```

Enter the height of the wall (in inches): 60
The wall is 5 feet tall

```

>>> |

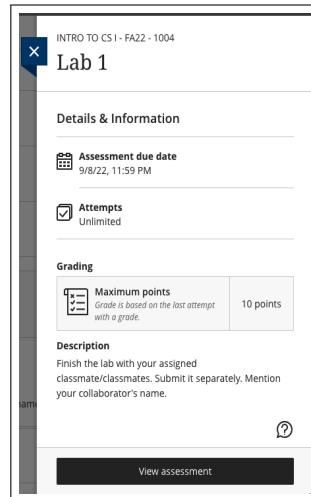
```

What to turn in:

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

The screenshot shows the Blackboard course interface. On the left, there is a sidebar with 'Course Faculty' (Md Reza, INSTRUCTOR), 'Details & Actions' (Roster, Course Description, Progress Tracking, Blackboard Collaborate, Attendance, Groups, Announcements, Books & Tools), and 'Course Content' (Support Center, CS65 Course Schedule, CS65 Course Syllabus, CS65 Zoom Link). The main content area shows 'Week 1' with 'Course materials for week1. These are also available (as links) from the course schedule.' Below this, there is a 'Lab 1' entry with a due date of 9/8/22, 11:59 PM and instructions: 'Finish the lab with your assigned classmate/classmates. Submit it separately. Mention your collaborator's name.'

When you click on Lab 1, another window will pop-up. Press on the View assessment button.



Upload your file by clicking on the 'Attachment' link. Once it is uploaded, hit the submit button to finish the submission process.

