

CSC108H/A08H Functions Lab

Welcome back to the CSC108H/A08H lab! To earn your lab marks, you must actively participate in the lab and make significant progress. At the end of the lab, please show your work to your TA and return your handout. The handout will be posted on the course website at the end of the week.

Your TA will assign you to a group, and you will pair program with your partner. Throughout the lab, you will see directions to “switch”. Whenever you encounter this, you and your partner should exchange roles (the driver becomes the navigator and vice versa). Remember:

driver: The person typing at the keyboard.

navigator: The person watching for mistakes **and** thinking ahead.

Both roles are active! The navigator should be helping the driver avoid mistakes, and the driver should be narrating what he or she is doing while typing.

1 Objectives

- Learn to use `if __name__ == "__main__"`.
- Use the debugger to trace through assignments and function calls.
- Understand the difference between `print` and `return`.
- Practice defining and calling functions.
- Practice importing modules and using functions from imported modules.

2 Tracing

In this exercise, we will practice using the debugger to understand how programs are executed by the computer. Download `cube.py` and `calc.py` from the Labs page on the course website.

1. **Open** `cube.py` in Wing. You should see two functions and a `__main__` block. The two functions compute the same value but use them in different ways. With your partner, discuss what the code does and, as a pair, write down what you expect to see when you run the program.
2. Click the green arrow (**Run**) button to execute the entire program `cube.py`. Is the output what you expected? If not, what differed?
3. Now, click **Step Into**, so that you can run the program one line at a time and watch what happens. Click **Step Into** again. Note how the function definition is executed. Now python knows what you want when you ask it to perform `cube`! Click **Step Into** again and the second function definition is executed.
4. Keep executing **Step Intos**, one at a time. After each line, look at in the **Stack Data** tab, which shows variable names and values at that point in the execution. Pay close attention to how `print` and `return` differ in terms of how they change program state.

Switch driver and navigator.

5. Now, **Open** `calc.py` and peruse the contents. This python file uses `cube.py` by importing it and calling functions from it. Considering what you know about “`if __name__ == __main__:`”, what do you think you will see when you run the program?

6. Use the debugger to explore `calc.py`. Be prepared to answer the following questions when you show your work to the TA:
 - **Step Over** the `import` statement. How does this differ from **Step Into**?
 - Why do you think it is important to use `__main__` blocks?
 - Is there any relationship between `x_cubed` in `calc.py` and the same name in `cube.py`? What about `x` in the function and `x_cubed`?

Switch driver and navigator.

3 Functions

It's time for you to write some functions of your own! Download `functions.py` and a sample picture `BahenSmall.jpg` from the Labs page on the course website.

The file `functions.py` contains a series of function definitions with docstrings but no real body — they just say `pass`, which does nothing, so that Python won't complain that they have no body. For each function, you should implement it according to its docstring. You should **handwrite** your solution. (This is great practise for the midterm questions that you will solve on paper.) Once you and your partner agree on a solution, replace the `pass` with the code you have written. In the `__main__` section, add a call to your function to verify that it works as you expect. If it does not, use Wing's debugger to trace through execution and see where the computer's state diverges from your expectations.

We will start with some simple functions and use them to build more complex operations.

1. Begin by implementing the function `intensity`. The intensity of a pixel is the average of its red, green, and blue components.
2. Use the `intensity` function you just wrote to create the function `grayscale`. A picture appears in grayscale if the red, green, and blue components of each pixel are all set to the intensity of that pixel. You must not change the original picture; you will find `media`'s `copy` function useful.

Switch driver and navigator.

3. Implement the `avg_intensity` function. The average intensity of a picture is the average of the intensities of all of its pixels.
4. Implement the `red_cross` function. Use Python's `help` function to learn about the function `media.add_rect_filled`. It may help to draw a picture on paper of the desired image and label it with (x, y) coordinates.

Congratulations! Once you're finished, call your TA to demonstrate your work and to receive credit for the lab. See you next week!