CSC120H Lab 1

Congratulations on getting started with the CSC120 labs. Now it’s time for you to do some programming. To earn your lab marks, you must submit your work by Friday September 15th at 9:59pm.

Please go to your Thursday lab to get help!

1 Objectives

1. Learn how to program in pairs as driver and navigator
2. Define driver and navigator
3. Practise working with expressions in the shell
4. Practise working with variables
5. Practise defining functions in the editor
6. Practise submitting a lab/exercise/assignment

2 Driver and navigator

Throughout the term, you will have the option to work alone or in pairs for the labs. We encourage pair programming as it helps when you are stuck. We will use the terms driver and navigator. Here are the definitions of the two roles:

driver: The person typing at the keyboard.
navigator: The person watching for mistakes, and thinking ahead.

Here is the most important rule for this and all future labs:

The navigator must not touch the keyboard or mouse. The driver’s role is to work with the computer, and the navigator’s is to think about language issues and upcoming issues related to the problem being solved. If the navigator interferes with the driver, the group loses its view of what is coming up and the problem may become harder to solve and the driver will find it harder to learn the material.1

In every lab handout, we’ll call you two s1 and s2, and s1 will be the first driver.

3 Numerical expressions in the shell

Now that the administrative details are out of the way, we can get started with the fun part of the lab! Many of you will be programming for the first time; that’s exciting, but may also be a bit overwhelming. Don’t hesitate to ask your TA or your labmates for help. You may also refer to your notes and the textbook.

To begin, click the WING IDE icon in the “Start” menu. In this lab, we’ll start by working in the Python shell. You TA can help you with using the Python shell in the Wing IDE.

1 Analogously, if you’re driving with a friend and looking at the map, do you think it’s wise to reach over and turn the steering wheel when it’s time to turn the corner? Especially if they’re just learning to drive?
Python has many calculator-like features. Try evaluating a few expressions in the shell. Before you hit the return (Enter) key, after typing each expression, try to first think what you expect the result to be.

- $34 + 8$
- $29 / 3$
- $29 // 3$
- $18 - 2.9$
- $24 \% 5$
- $5.5 * 4.0$
- $28 / 3.0$
- $28 // 3.0$
- $14 * 3 + 2$
- $14 * (3 + 2)$
- etc...

Checking your understanding:

- Which of the above expressions you would not see on a calculator? Explain to your partner why that is the case. Now have your partner explain it to you.
- Explain the values calculated, watching out for decimal points and what comes after them. Check that your answers agree with your mental arithmetic.

4 Variables

The values in the expressions that we are working with often have more meaning than the numbers alone reveal. For example, the expression $0.5 * 10 * 5$, might be the area of a triangle, where the value $10$ is the base and $5$ is the height. To associate a name with those values, we use variables.

The general form of an assignment statement is: \textit{variable} = \textit{expression}

Switch roles: \textit{s2} drives and \textit{s1} navigates.

- Assign the value 10 to a variable named \textit{base} and the value 5 to a variable \textit{height}.
- Now rewrite the expression $0.5 * 10 * 5$ so that it uses variables \textit{base} and \textit{height}.
- Next, assign the expression to a variable named \textit{area}.
- Reassign variables \textit{base} and \textit{height} the values 12 and 17, respectively. What is the value of \textit{area} now? Type \textit{area} and the return key to see \textit{area}'s value. If the value of \textit{area} is not what you expected, and you can't figure out why, call your TA over for help.
- Verify the current variable values by typing each variable in the shell followed by the return key. Now recalculate the area (using the variables) and check its value.
- \textit{Wing 101 Shell Tip: Try using the up/down arrows of your keyboard. See what happens if you press any one of them multiple times.}
5 A function

In the previous section, we calculated the area of a triangle. Let’s define a function, called calculate_area, so that we can reuse the code we wrote. The general form of a function is:

```python
def function_name(parameters):
    body
```

Switch roles: s1 drives and s2 navigates.

Create a new file called `triangle.py`. In this file, define a function `triangle_area`.

- parameters: `base` and `height`
  (Any legal variable names would do, but it’s best to choose something meaningful.)
- The function should calculate the triangle’s area for the given base and height, and return the area.

Switch roles: s2 drives and s1 navigates.

Click the Run button (green triangle). In the shell, call the function as follows:

```python
triangle_area(10, 12)
triangle_area(3.4, 5.2)
triangle_area(base, height)
```

6 Practise submitting an assignment

This section is very important!! If you leave figuring out how to submit your work until you really need to do it, you won’t be able to, because that’s how the world works.

The assignments in this course will be submitted electronically. You will need to go to MarkUs by visiting this website:

https://markus.teach.cs.toronto.edu/csc120-2017-09

to submit your labs, exercises and assignments.

Switch roles: s1 drives and s2 navigates.

The driver logs into MarkUs at the website above. s1 should select the assignment L1: Lab 1 and submit your file `triangle.py`.

Switch roles: s2 drives and s1 navigates: s2 should also log into MarkUs and submit `triangle.py`. 