**EECS 1015: LAB #8 – Objects and Classes**
**Assigned: Nov 15, 2021 (Monday)**
**Due date: Nov 26, 2021 (Friday) [11.59pm Eastern Time]**

**#Important reminder**
1) You must submit your lab via web-submit.

2) Submit only a single file named `lab8.py`.

3) Please follow the instructions carefully – read the lab to understand everything you need to do. This lab requires you to implement a class with multiple methods and associated data.

## 1. GOALS/OUTCOMES FOR LAB

- To learn to use objects and classes
- To practice more general programming
- To practice list manipulation, including append, extend, and remove

## 2. LAB 8 – TASK/INSTRUCTIONS

**Task 0: [This will be the same for all labs]:** Start your code with comments that include this lab ID, your full name, email address, and student id as follows:

```
# Lab 8
# Author: Michael S. Brown
# Email: msb99898@aol.com
# Student ID: 10233030  <- please do not add spaces
# Section A or B
```

**This lab involves generating a with several methods. You will also need to keep a list of user-defined objects and pass that to a function.**

https://www.eecs.yorku.ca/~mbrown/EECS1015_Lab8.mp4

This lab has only 1 task.

See the explanation of the lab on the next pages.

## Lab 8's Task – Snail racing with the Snail Object

You are to write a program that simulates "snail racing." Specifically, we will have several Snail objects. Each Snail will move at a random speed between [0.1, 0.2, 0.3, .. , 1.0]. Each snail object will also have its own ID as two initials (a two-character string).

The program will ask the user how many snails they want to race. The user will be asked to provide initials for each snail, and your program will create a Snail object and place it in a list. After creating a list of Snail objects, a "race" will start, and each Snail will move based on its speed. You should print out each snail's "animation" each race iteration until a snail reaches the finish line. See details below. *Please watch the associated video posted with this lab*. Below is an example of the program output.

----- Output of program (user input shown in red) -------

```
Snail Race . . . .
How many snails are racing? 3
Snail 1 two initials: MB
Snail 2 two initials: AB
Snail 3 two initials: BB
Press enter to start race.
----------------------------------------Time  1
__~@                                        MB
__~@                                        AB
__~@                                        BB
----------------------------------------Time  2
_~_@                                        MB
_~_@                                        AB
_~_@                                        BB
----------------------------------------Time  3
~__@                                        MB
 ~__@                                       AB
~__@                                        BB
----------------------------------------Time  4
__~@                                        MB
 __~@                                       AB
 __~@                                       BB
----------------------------------------Time  5
_~_@                                        MB
  _~_@                                      AB
 _~_@                                       BB
----------------------------------------Time  6
 ~__@                                       MB
   ~__@                                     AB
  ~__@                                      BB
----------------------------------------Time  7
 __~@                                       MB
   __~@                                     AB
  __~@                                      BB
....
----------------------------------------Time  65
            _~_@                            MB
                                      _~_@  AB
                         _~_@               BB
----------------------------------------Time  66
            ~__@                            MB
                                       ~__@ AB
                         ~__@               BB
----------------------------------------Time  67
             __~@                           MB
                                       __~@ AB
                          __~@              BB
Snail AB WON!
Play again (Y)?
```

Notice how each snail "animates" at each time step.

At each time step of the race, the Snail's will update their output to reflect their current position. Those with a faster speed will updated their position faster. See details on how the Snail object behaves on the next page.

Jumping ahead to time step 65.

The first snail in the list to reach a position greater than 39 wins the race (there is a method to get the snail's position).

Print out the initials of the snail that won.
Ask the user if they want to play again. If yes, start all over by asking the user the number of snails, and running a new race.

## 2. Implementation Details

### Snail Class

**Your class should have the following:**

**Class data** (1)

1) A *list* that stores the animation sequence.

`"__~@", "_~_@", "~__@"`

**Instance data (4)**

| | |
|---|---|
| `name` | a *string* to represent the snail's initials |
| `speed` | a *float* between [0.1 – 1.0] |
| `frame` | an *integer* for the current animation frame 0, 1, or 2 [this will index into the Class list variable] |
| `pos` | a *float* that represents the snail's current position |

**Methods (4)**

`__init__()` single parameter for the name of the self (string of 2 characters); no return

- set name to the argument passed in the constructor (convert the string to upper case)
  if the string is more than two characters, assert with the error "Snail's initials must be 2 characters."
- set speed a random number between 1-10 that is divided by 10 (this will give the range [0.1, 0.2, .., 0.9, 1.0])
- set frame and pos to 0

`move()` no parameters; no return

- set pos equal to pos + speed
- update frame by 1 (remember, frame is an index into your current animation)
  if frame is greater than 2, set back frame back to 0

`getIntPos()` no parameters; return an *integer*

- convert the position instance variable to an integer
- you can do this using the `int()` function. For example, `int(9.4)` is 9
- return the integer

`getName()` no parameters; returns a *string*

- This is a *getter* method that returns the name instance variable

`getStr()` no parameters; returns a *string* -- see details, this is an import method

- This method will create a long string that depicts the snail moving.
- This string will always be length 46
- The string will change depending on the values of the instance variables.
- See details in Figure 1 on how describing how to compute the string.

For this example, assume the object's instance variables are as follows:

int(pos) is 21 (recall, pos is a float, so `int()` is converting it to an integer)

name is "TJ"

frame is 1

Return a string as follows:

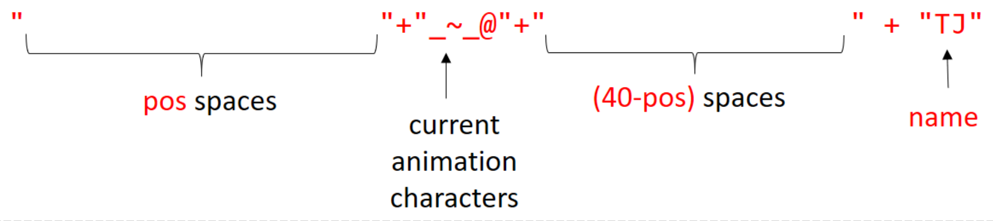


*Figure 1*

## Implementation continued

### Functions

Please implement the following functions that use the Snail objects.

`getSnailList(no param)` – *returns a list of snail objects*

1. Ask the user how many snails they want to race (let's call this N)
2. For each snail, ask the user for its 2-character initials
   - Create a Snail object with the 2-character initials and add the object to a list
3. Return the list of snail objects

`runRace(param: list of snail objects)` – *no return value*

1. Ask the user to press enter to start the race
2. Set the current time step to 1
3. Print out the current time step as shown in the example above (40 "-" + "Time " + current time step)
4. For each snail object in the snail list
   - get its "race string" and print it out
   - call its move() method
   - get snail's integer position and see if it is greater than 39 (if so, the first one to reach the finish wins)
5. Slow down the Output by calling `sleep()` for 0.2 seconds (see below*)
6. If no winner for this iteration, increment the time step, go back to 3 and repeat
7. If there is winner, print out the winner's names and end the function

`main()`

1. print out "Snail Race . . ."
2. Call `getSnailList()`
3. call `runRace()` with the snail list
4. Ask the user to play again - -if yes, go back to 1

***You'll need the `sleep()` function**

We will be simulating the animation of snails moving by printing out text on the screen. To slow down the Output, we will use the `sleep()` function from the `time` package.

Here is an example of how it works (see trinket code too: https://trinket.io/python/34fd87aaf4 )

```
import time

print("0.5s delay")
for i in range(1,6):
  print(i)
  time.sleep(0.5) # pauses for 0.5 seconds
```

The description of the Snail class and functions should be more than sufficient for you to implement the lab. Your Output should look very close to what is shown in the video.

## 3. GRADING SCHEME (Maximum number of points possible 10)

To get full marks, you need to make sure you follow the instructions correctly.   The following will be our grading scheme for the Lab components specified in Section 2 of this document.

**Task 0: (0 points, but deduction if you skip this part)**

- Filename **must** be "Lab8.py"  (all lowercase, no spaces)
- The Python comments at the beginning of your program **must** include your name, email, and York student id  (this is important for grading)
- *If your file name is incorrect, or you do not put in the required information, we will deduct -5 points (Why are we so harsh?  Because if you don't put in your name and student id, it can be very difficult for the TAs to determine whose submission this is.)*

**Main Task :**

- 2 points for a reasonable try
- 5 points for almost correct
- 10 points for a correct solution

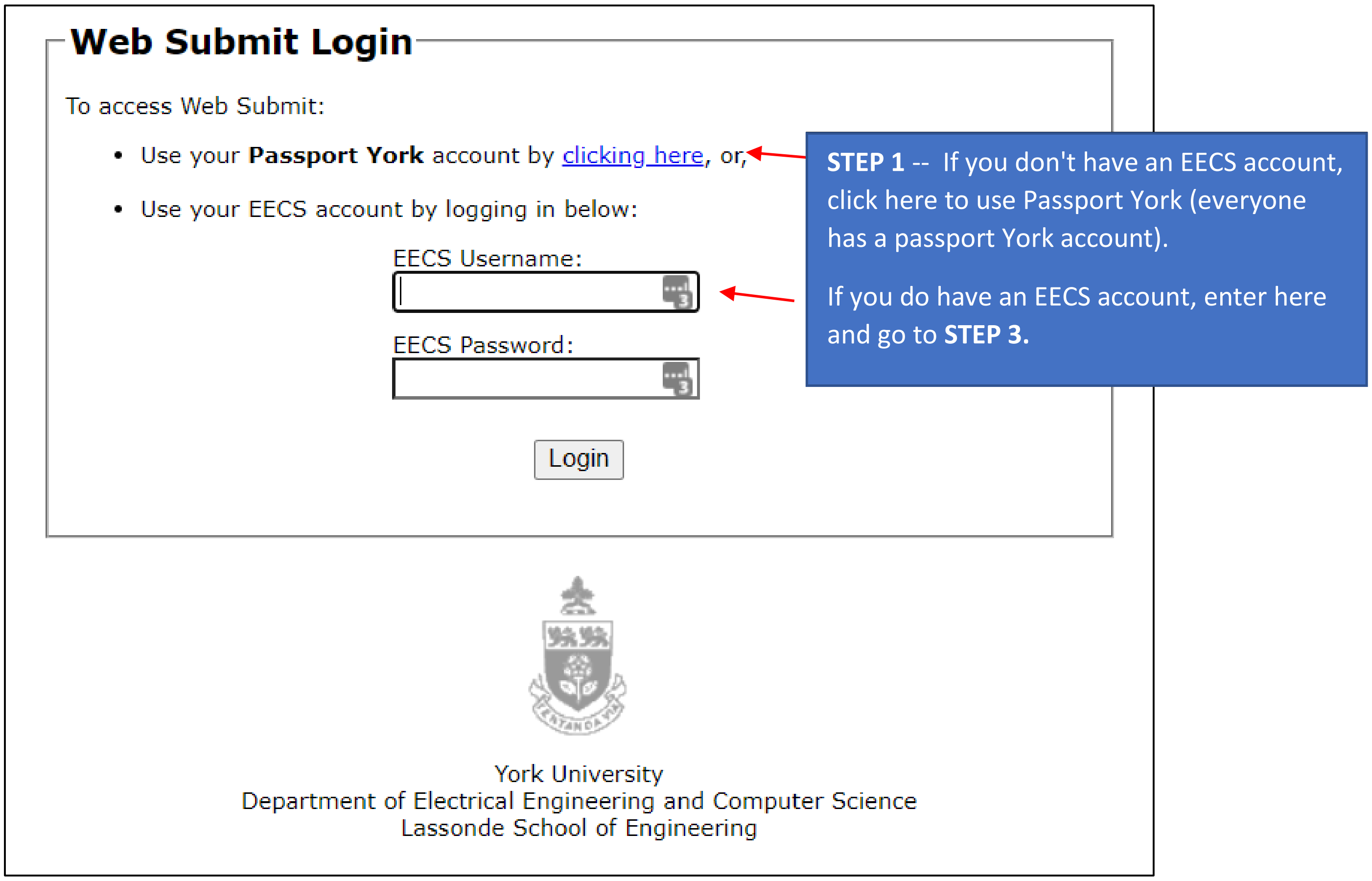
-No submission – 0 points
-Any submission up to one week after the due date is 50% off the total marks
-Any submission after one week of the due date will not be marked and treated as no submission.

See the pages below on how to submit your lab code.
MAKE SURE TO SELECT **Lab8** with websubmit

**4. SUBMISSIONS (**EECS web-submit)

You will submit your lab using the EECS web submit.

Click on the following URL: https://webapp.eecs.yorku.ca/submit



**Web Submit Login**

To access Web Submit:

- Use your **Passport York** account by clicking here, or,
- Use your EECS account by logging in below:

EECS Username:

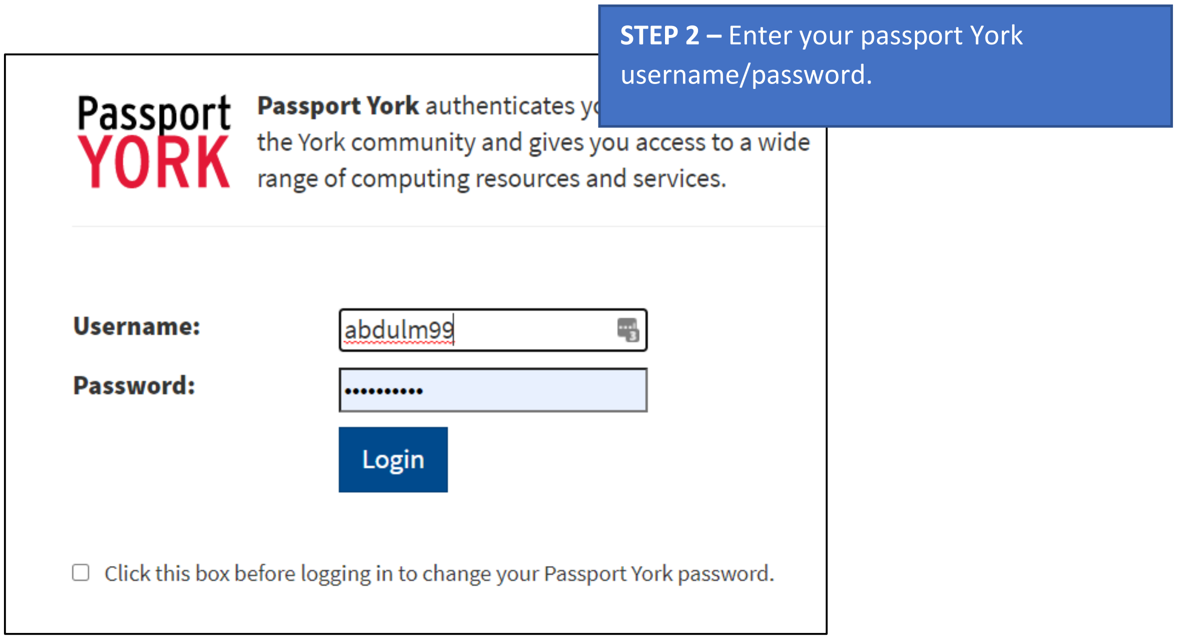
EECS Password:

Login

York University
Department of Electrical Engineering and Computer Science
Lassonde School of Engineering

**STEP 1** -- If you don't have an EECS account, click here to use Passport York (everyone has a passport York account).

If you do have an EECS account, enter here and go to **STEP 3.**



**STEP 2 –** Enter your passport York username/password.

Passport YORK

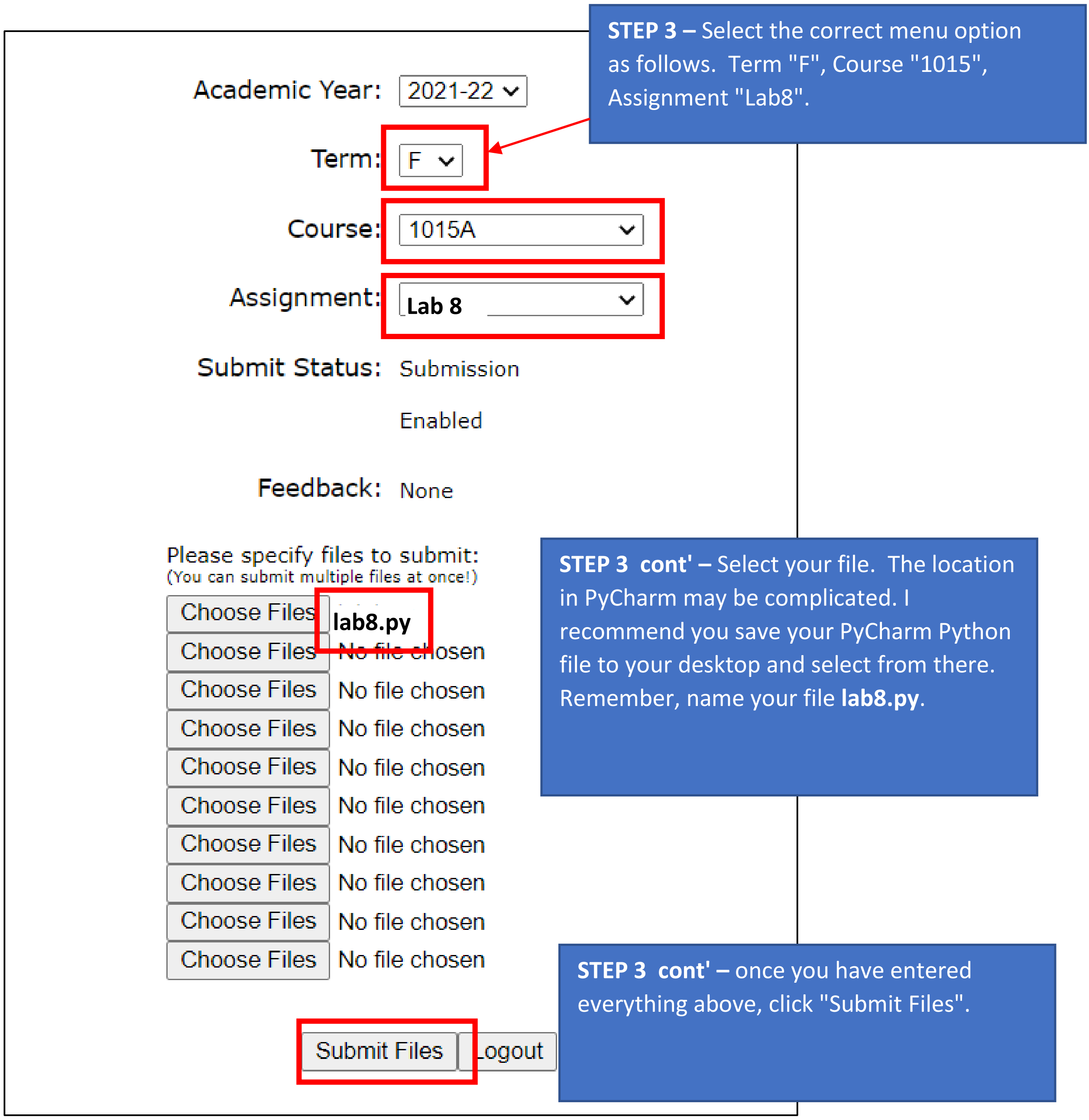
**Passport York** authenticates y the York community and gives you access to a wide range of computing resources and services.

**Username:** abdulm99

**Password:** ••••••••••

Login

☐ Click this box before logging in to change your Passport York password.



**STEP 3 –** Select the correct menu option as follows. Term "F", Course "1015", Assignment "Lab8".

Academic Year: 2021-22

Term: F

Course: 1015A

Assignment: Lab 8

Submit Status: Submission Enabled

Feedback: None

Please specify files to submit:
(You can submit multiple files at once!)

Choose Files lab8.py

Choose Files No file chosen

Choose Files No file chosen

Choose Files No file chosen

Choose Files No file chosen

Choose Files No file chosen

Choose Files No file chosen

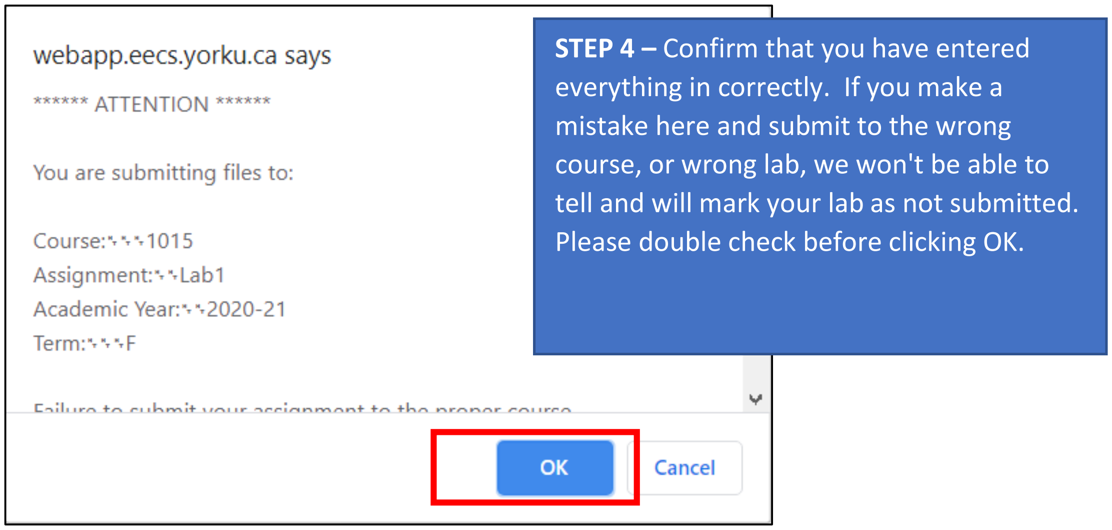
Choose Files No file chosen

Choose Files No file chosen

Choose Files No file chosen

**STEP 3 cont' –** Select your file. The location in PyCharm may be complicated. I recommend you save your PyCharm Python file to your desktop and select from there. Remember, name your file **lab8.py**.

**STEP 3 cont' –** once you have entered everything above, click "Submit Files".



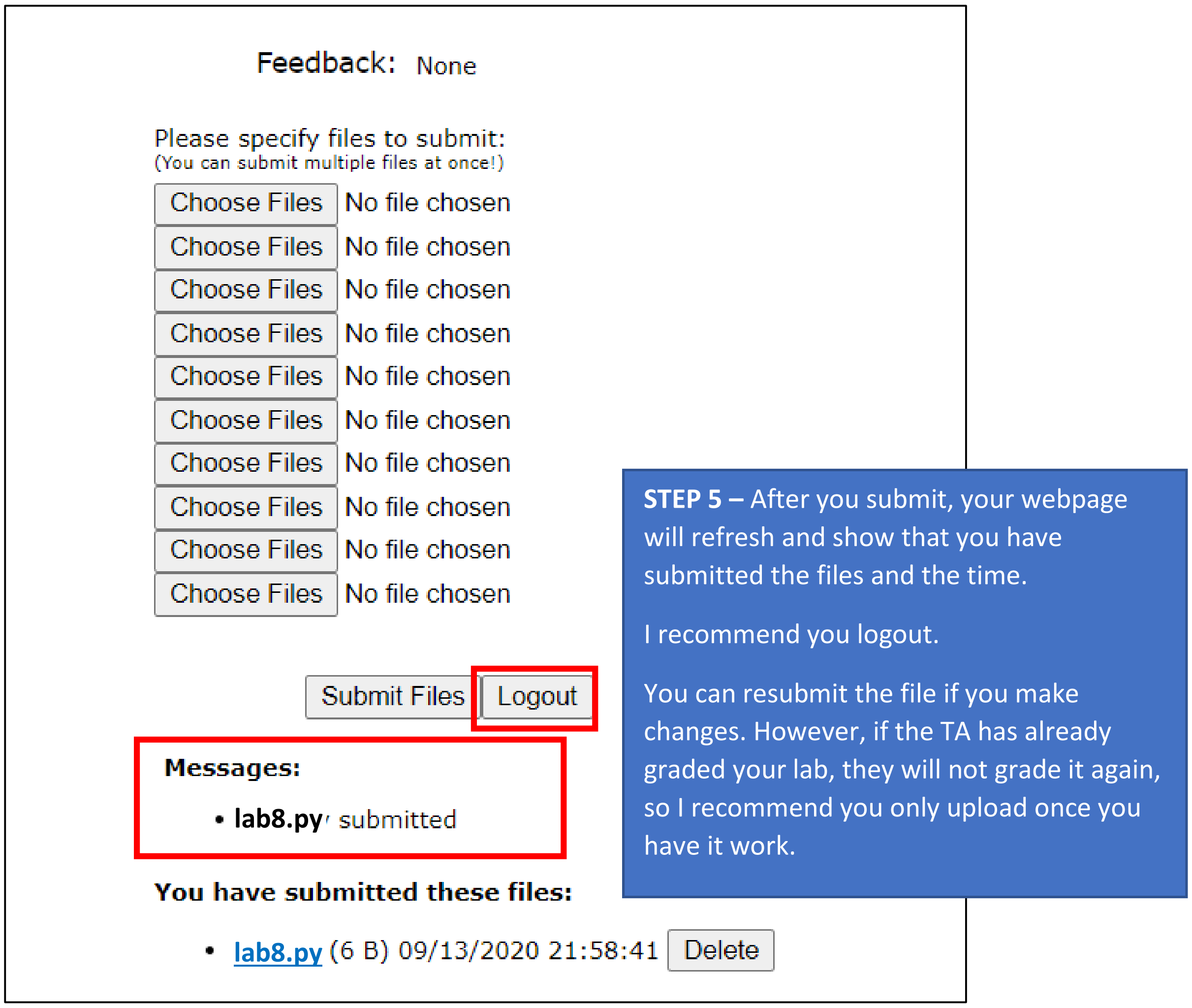
webapp.eecs.yorku.ca says

****** ATTENTION ******

You are submitting files to:

Course: 1015
Assignment: Lab1
Academic Year: 2020-21
Term: F

**STEP 4 –** Confirm that you have entered everything in correctly. If you make a mistake here and submit to the wrong course, or wrong lab, we won't be able to tell and will mark your lab as not submitted. Please double check before clicking OK.


Feedback: None
Please specify files to submit:
(You can submit multiple files at once!)
Choose Files No file chosen
Choose Files No file chosen
Choose Files No file chosen
Choose Files No file chosen
Choose Files No file chosen
Choose Files No file chosen
Choose Files No file chosen
Choose Files No file chosen
Choose Files No file chosen
Choose Files No file chosen
Submit Files
Logout
Delete

**Messages:**

- **lab8.py** submitted

**You have submitted these files:**

- lab8.py (6 B) 09/13/2020 21:58:41

**STEP 5 –** After you submit, your webpage will refresh and show that you have submitted the files and the time.

I recommend you logout.

You can resubmit the file if you make changes. However, if the TA has already graded your lab, they will not grade it again, so I recommend you only upload once you have it work.

For more details on *websubmit*, see EECS department instructions:

https://wiki.eecs.yorku.ca/dept/tdb/services:submit:websubmit