EECS 1015: LAB #3 – If-statements and loops
Assigned: Oct 4, 2021 [Monday]
Due date: Oct 18, 2021, 11:59 pm [Monday] (extra time due to reading week)

#Important reminder for your third lab

1) You must submit your lab via web-submit.

2) Please make sure you correctly submit your file(s).

3) Please follow the instructions carefully – read the lab carefully to understand everything you need to do; this lab has multiple parts.

1. GOALS/OUTCOMES FOR LAB

- To practice if-statements and while/for loops
- To continue practicing with string input and string processing
- To write your own Python code

2. LAB 3 – TASK/INSTRUCTIONS

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

```
# Lab 3
# Author: Michael S. Brown
# Email: msb99898@aol.com
# Student ID: 10233030
# Section A or B
```

Starting code for this lab: https://trinket.io/python/4bec7c8067

This lab has four tasks. Please read each carefully. You can also watch the accompanying video linked here:
https://www.eecs.yorku.ca/~mbrown/EECS1015_Lab3.mp4

***Note**: For all tasks, you can assume the input from the user will be correct, you do not need to check, unless the task explicitly asks you to check. For example, Task #4 requires you to ensure the number entered by the user is between 5 and 20.*

Task 1 [Computing the Fare Price]

Goal: This task is to compute the ticket price for a person based on the type of fare (one way or round trip) and their age group (under 12, 12-64, 64, and older).

1. Your program should first ask the person which type of fare they want: (1) one way or (2) round trip.
2. Your program should then ask the person to select the correct again group: (1) under 12; (2) 13-64; or (3) 65 or older.
3. You then should compute the fare as follows:
   One way is $1.80; round trip is $3.50.
   If the person is under 12 or 65 or older, the fair should be reduced by 50%.
4. After the input, print out the computed fare in the following format:

```
Total amount due: $1.80 [one way (full fare)]
Total amount due: $3.50 [round trip (full fare)]
Total amount due: $0.90 [one way (reduced fare)]
Total amount due: $1.75 [round trip (reduced far])]
```

See output for several examples (user input is in red)

```
--- Task 1 ---: Compute Fare
(1) One way or (2) round trip?
Enter 1 or 2: 2
Select Age Range:
(1) Under 12
(2) 13-64
(3) 65 or older
Enter 1, 2, or 3: 1
Total amount due: $1.75 [round trip (reduced fare)]

--- Task 1 ---: Compute Fare
(1) One way or (2) round trip?
Enter 1 or 2: 1
Select Age Range:
(1) Under 12
(2) 13-64
(3) 65 or older
Enter 1, 2, or 3: 2
Total amount due: $1.80 [one way (full fare)]

--- Task 1 ---: Compute Fare
(1) One way or (2) round trip?
Enter 1 or 2: 1
Select Age Range:
(1) Under 12
(2) 13-64
(3) 65 or older
Enter 1, 2, or 3: 3
Total amount due: $0.90 [one way (reduced fare)]
```

Task 2 [Print each character in a string. Compute a reverse string with spaces removed]

Goal: This task prints out each character in a string and then prints it in reverse with spaces removed.

1. Ask the user to input a long string.
2. Print out each character in the string as shown below. If the character is a space (i.e., a " "), print out the word "SPACE."
3. Finally, print a new string with the input characters in reverse order with spaces removed.

See output for several examples (user input is in red)

```
--- Task 2 ---: Parse string
Input a string: Hello World.
str[0] = H
str[1] = e
str[2] = l
str[3] = l
str[4] = o
str[5] = SPACE
str[6] = W
str[7] = o
str[8] = r
str[9] = l
str[10] = d
str[11] = .
Reverse no spaces: .dlroWolleH

--- Task 2 ---: Parse string
Input a string: Another    test.
str[0] = A
str[1] = n
str[2] = o
str[3] = t
str[4] = h
str[5] = e
str[6] = r
str[7] = SPACE
str[8] = SPACE
str[9] = SPACE
str[10] = SPACE
str[11] = t
str[12] = e
str[13] = s
str[14] = t
str[15] = .
Reverse no spaces: .tsetrehtonA
```

Task 3 [Finding the maximum even number]

Goal: The goal is to find the maximum *even* number from a sequence of positive numbers entered by the user. Stop prompting the user for numbers after they enter a negative number.

1. Repeatedly prompt the user to enter positives numbers.
2. Keep track of the largest even number entered by the user?

   Hint: How do you know if a number is even? Any number modulus two that results in a 0 is an even number.

   For example: `5 % 2 = 1 (odd), 4 % 2 = 0 (even), 12 %2 = 0 (even), 101%2=1 (odd)`
3. When the user enters a negative number, stop and print out the largest even number entered.
4. If the user does not enter an even number, output 0 as the output.

See output for several examples (user input is in red)

```
--- Task 3 ---: The maximum *even* number
 Keep entering positive integer
 To quit, input a negative integer
 Enter a number: 10
 Enter a number: 11
 Enter a number: 103
 Enter a number: 99
 Enter a number: 88
 Enter a number: -1
 Largest even number: 88

 --- Task 3 ---: The maximum *even* number
 Keep entering positive integer
 To quit, input a negative integer
 Enter a number: 99
 Enter a number: 109
 Enter a number: 122
 Enter a number: 44
 Enter a number: 8008
 Enter a number: 9387
 Enter a number: 33
 Enter a number: 99
 Enter a number: 806
 Enter a number: -1
 Largest even number: 8008

 --- Task 3 ---: The maximum *even* number
 Keep entering positive integer
 To quit, input a negative integer
 Enter a number: -1
 Largest even number: 0     <- No even number was provided.
```

Task 4: Better triangle draw

Goal: Draw a triangle based on a number between 5-20 that the user provides.

1. Ask the user to input a number between 5 and 20
   If the number they input is not between 5 and 20, ask again until it is.
3. Draw a triangle as shown below.
4. This task is similar to the example in the lecture notes but requires modifications.

See output for several examples (user input is in red)

```
--- Task 4 ---: Better triangle draw
Enter size between 5 and 20: 2
Enter size between 5 and 20: 21
Enter size between 5 and 20: 5
\                <- 1       a single '\'
-\               <- 2       1 '-' followed by a '\'
--\              <- 3       2 '-' followed by a '\'
---\             <- 4            .
----\            <- 5            .
-----|           <- 6       5 '-' followed by a '|'
----/            <- 7       4 '-' followed by a '/'
---/             <- 8       3 '-' followed by a '/'
--/              <- 9            .
-/               <- 10      1 '-' followed by a '/'
/                <- 11      a single '/'
```

```
--- Task 4 ---: Better triangle draw
Enter size between 5 and 20: 7
\                <- 1       a single '\'
-\               <- 2       1 '-' followed by a '\'
--\              <- 3       2 '-' followed by a '\'
---\             <- 4            .
----\            <- 5            .
-----\           <- 6            .
------\          <- 7            .
-------|         <- 8       7 '-' followed by a '|'
------/          <- 9       6 '-' followed by a '/'
-----/           <- 10      5 '-' followed by a '/'
----/            <- 11           .
---/             <- 12           .
--/              <- 13           .
-/               <- 14      1 '-' followed by a '/'
/                <- 15      a single '/'
```

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

This lab is more challenging than lab 2. However, the notes and trinkets examples are all sufficient to help you do this lab. 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)

- File name **must** be "lab3.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, your or 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.)*

Task 1-4: (2.5 points each)

- Each task should prompt the user correctly and compute the required output correctly.
- Please watch the accompanying video.

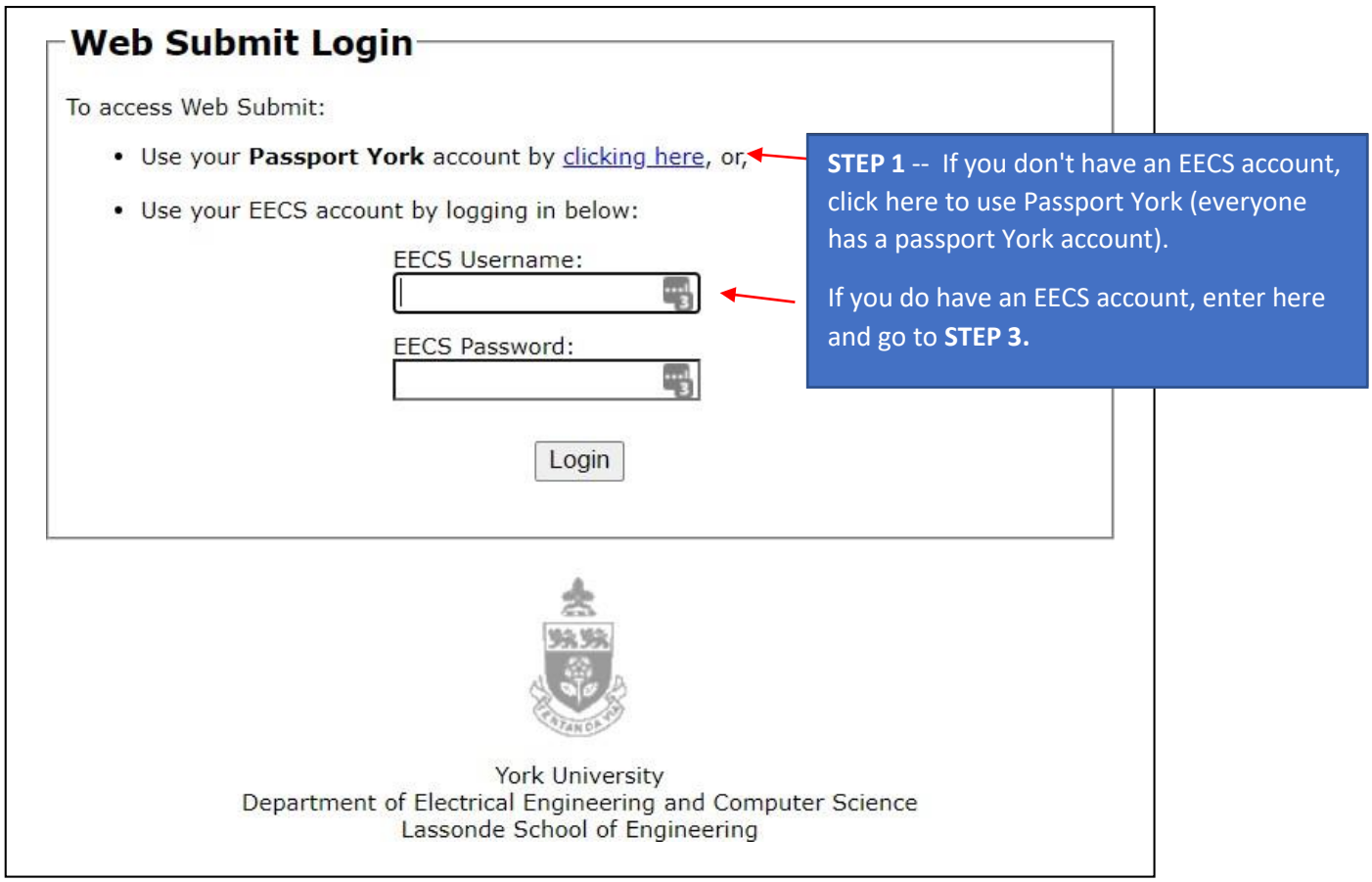
-No submission – 0 points
-Any submission 1 week after the due date 50% off the total marks
-Any submission 2 weeks after the due date will not be marked and treated as no submission.

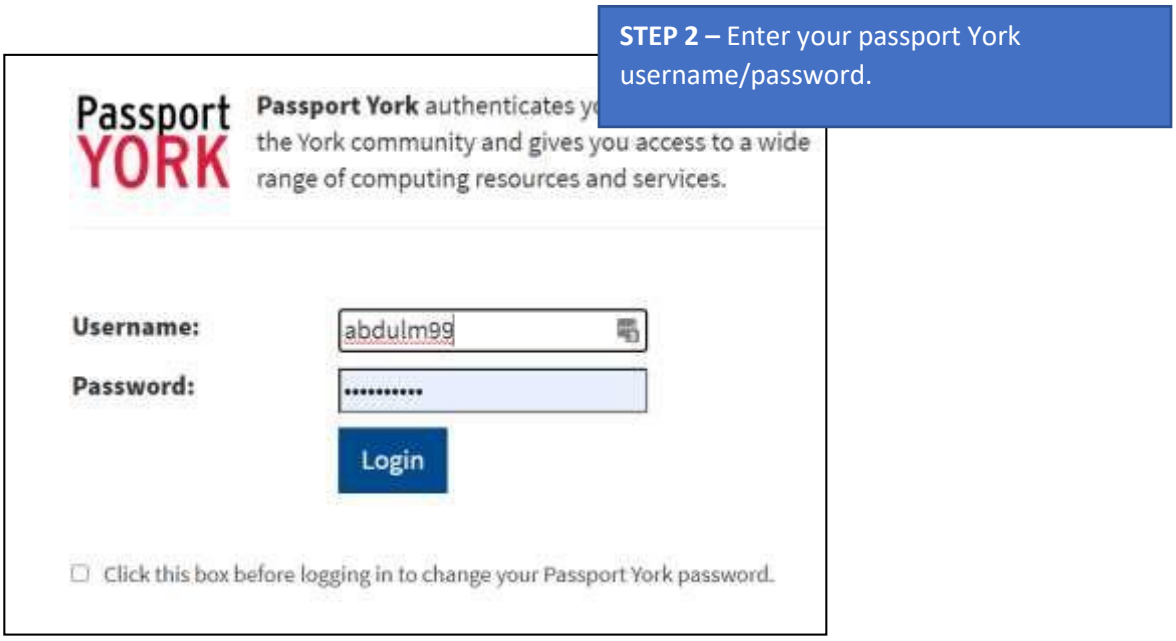
See pages below on how to submit your lab code.
MAKE SURE TO SELECT **Lab3** with websubmit

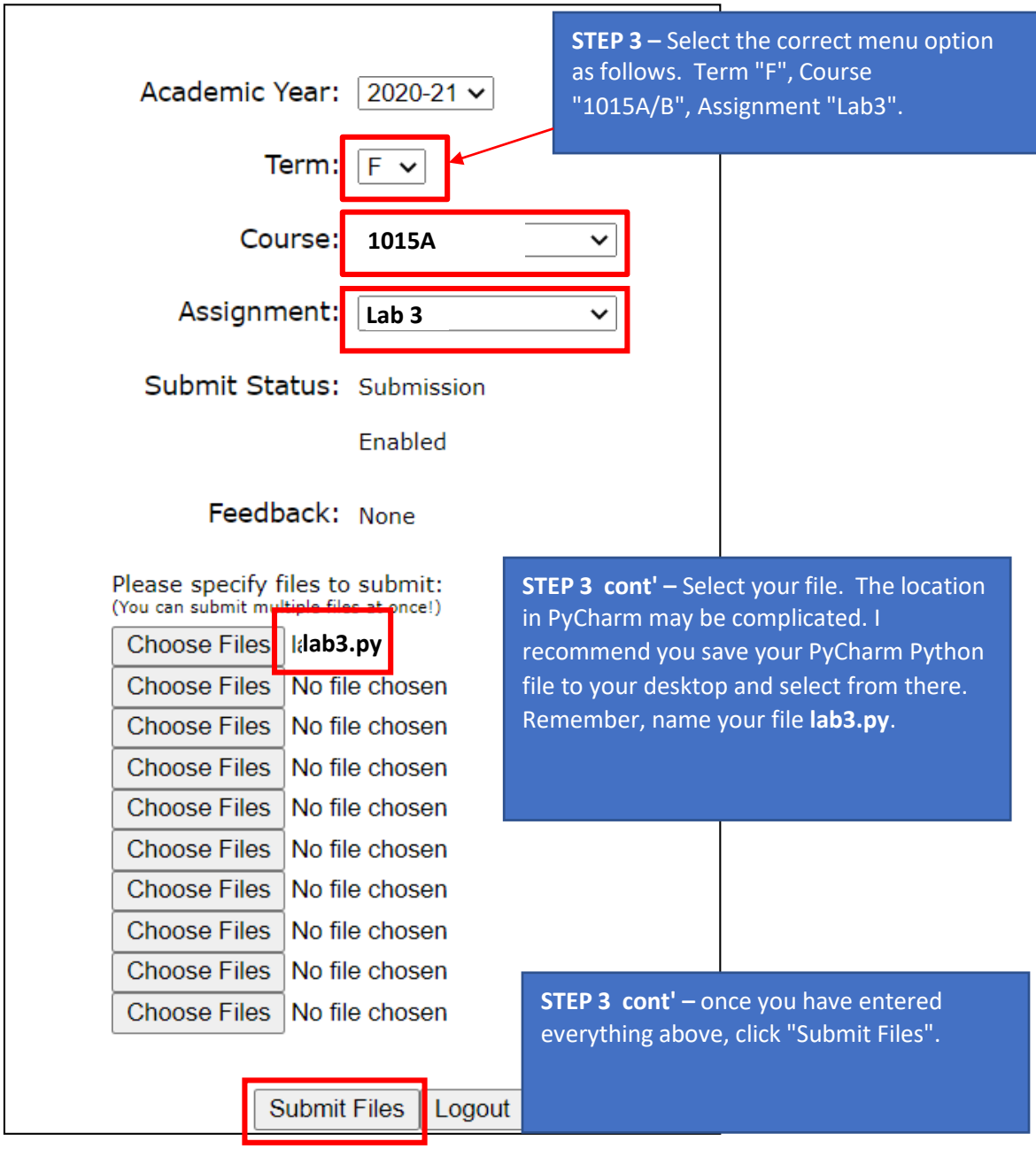
4. SUBMISSION (EECS web-submit)

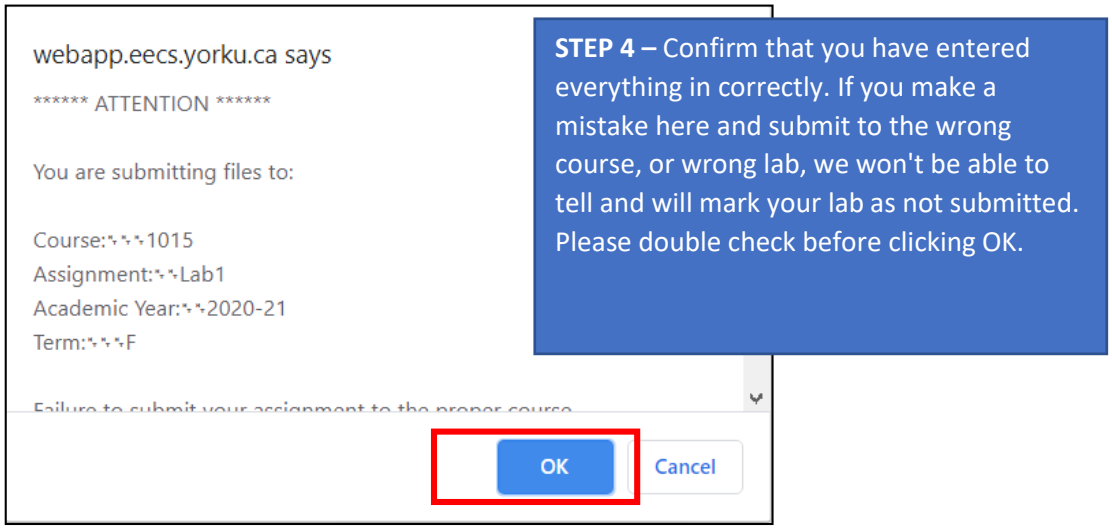
You will submit your lab using the EECS web submit.

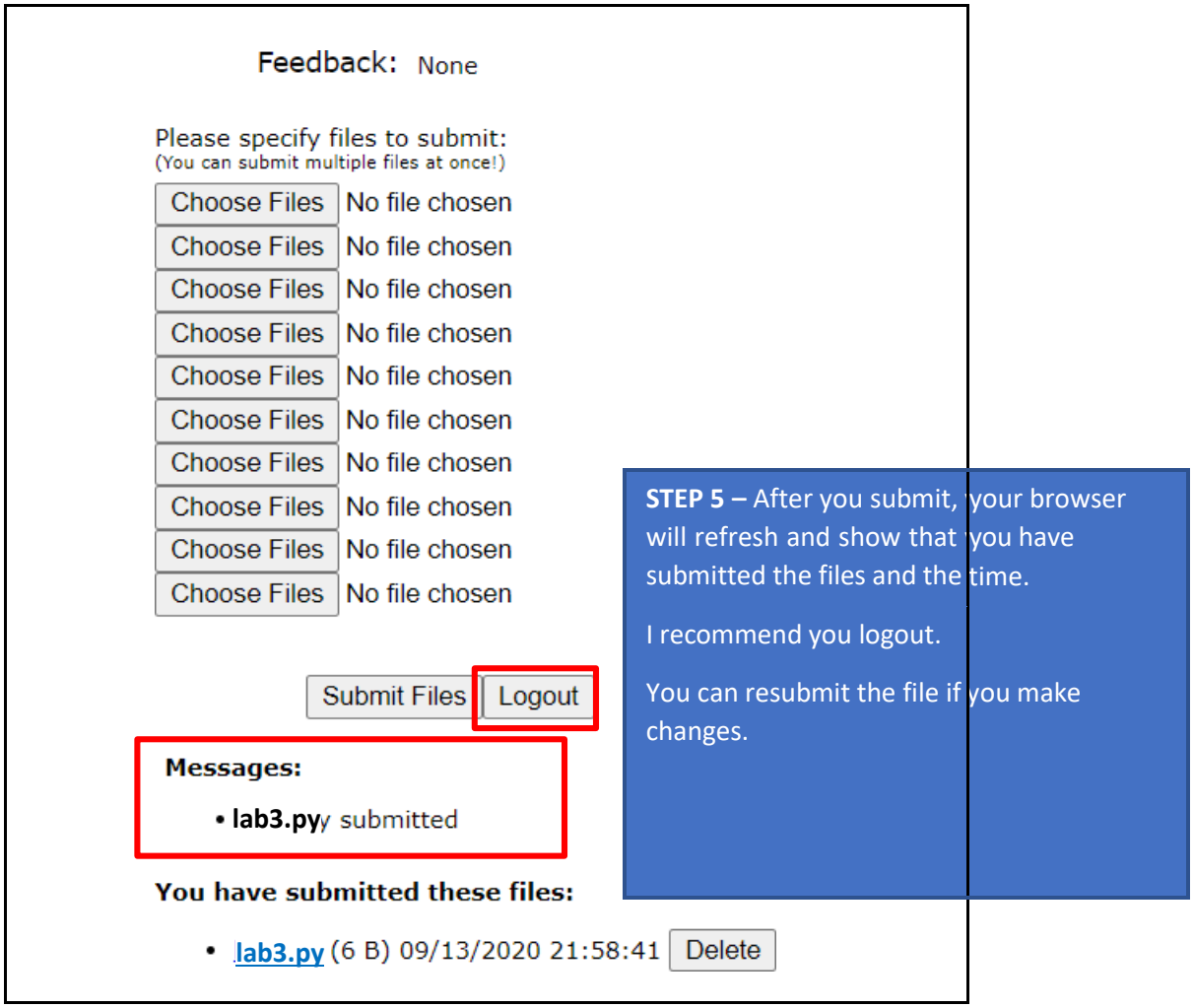
Click on the following URL: https://webapp.eecs.yorku.ca/submit






Academic Year: 2020-21
Term: F
Course: 1015A
Assignment: Lab 3
Submit Status: Submission Enabled
Feedback: None
Please specify files to submit:
(You can submit multiple files at once!)
Choose Files lab3.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
Submit Files
Logout
STEP 3 – Select the correct menu option as follows. Term "F", Course "1015A/B", Assignment "Lab3".
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 lab3.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
OK
Cancel
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.



For more details on websubmit, see EECS department instructions:

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