EECS 1015: LAB #7 – Writing testing code for use with PyTest

Assigned: Nov 8, 2021
Due date: Nov 19, 2021 [11.59pm Eastern Time]

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

2) Please make sure you correctly submit your file (only a single file please – `lab7.py`).

3) Please follow the instructions carefully – read the lab carefully to understand everything you need to do. This lab requires you to implement multiple functions. Each function uses or processes lists, dictionaries, and tuples.

1. GOALS/OUTCOMES FOR LAB

- To practice using assert
- To be more familiar with type hinting
- To write test function to test your code using Pytest

2. LAB 7 – 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 7
# Author: Michael S. Brown
# Email: msb99898@aol.com
# Student ID: 10233030
```

This lab involves generating several functions. Please read carefully. A video of this lab running is available here.
https://www.eecs.yorku.ca/~mbrown/EECS1015_Lab7.mp4

This lab has two (2) tasks. You will also need to install Pytest – see course notes.

See the explanation of the lab on the next pages.

Lab 7 – Writing test cases for code

STARTING CODE LINK: https://www.eecs.yorku.ca/~mbrown/Lab7_start.py

The starting code is also shown here. There are three functions of importance (shown in blue)

```
import pytest
from typing import List

# Accepts a list of integers
def initializeMinMaxList(myList: List[int]) -> None:   # given
    myList.sort()

def insertItem(myList: List[int], item: int) -> None:  # given
    myList.append(item)
    myList.sort()

def getMinMax(myList: List[int], minormax: str) -> int:   # given -- but requires additional assert
    assert minormax.upper()=="MAX" or minormax.upper()=="MIN", "2nd argument must be 'Min' or 'Max' "
    result: int
    if minormax == "MAX":
        result = myList[-1]
        del myList[-1]
    else:
        result = myList[0]
        del myList[0]
    return result

# Main function is given.
def main():
    aList = [10, 11, 99, 1, 55, 100, 34, 88]
    print("Starting List: ", aList)
    initializeMinMaxList(aList)
    min1 = getMinMax(aList, "MIN")
    print("1st min: %d" % (min1))
    min2 = getMinMax(aList, "MIN")
    print("2nd min: %d" % (min2))
    max1 = getMinMax(aList, "MAX")
    print("1st max: %d" % (max1))
    max2 = getMinMax(aList, "MAX")
    print("2nd max: %d" % (max2))

    print("Insert %d %d %d %d" % (min1 - 1, min2 - 1, max1 + 1, max2 + 1))
    insertItem(aList, min1 - 1)
    insertItem(aList, min2 - 1)
    insertItem(aList, max1 + 1)
    insertItem(aList, max2 + 1)

    min1 = getMinMax(aList, "MIN")
    print("1st min: %d" % (min1))
    min2 = getMinMax(aList, "MIN")
    print("2nd min: %d" % (min2))
    max1 = getMinMax(aList, "MAX")
    print("1st max: %d" % (max1))
    max2 = getMinMax(aList, "MAX")
    print("2nd max: %d" % (max2))

    print("DONE.  Please Enter to exit.")
    input()

if __name__ == "__main__":
    main()
```

What does the code provided do?

The code provides a set of functions that implements a "minmax list" of integers. There are three functions we will use to with the minmax list, namely `initializeMinMaxList()`, `insertItem()`, and `getMinMax()`.

The minmax-list works as follows.

(1) Assume you have a list of integers.
e.g., x = [20, 99, 88, 1, 100, 0]

(2) Call function `initializeMinMaxList( List[int] )` -> None
e.g., `initializeMinMaxList(x)`

(3) After it has been initialized, you can insert items into the list using `insertItem( List[int], int)` -> None
e.g., `insertItem(x, 10), insertItem(x,-1)`

(4) To exact the min or max from the list, call function `getMinMax( List[int], str="MIN" or "MAX") -> int`

```
minItem = getMinMax( x, "MIN")
maxItem = getMinMax(x, "MAX")
```

The function `getMinMax()` will return the minimum or maximum item from the list, and delete the item from the list.

`getMinMax(List[int], str="MIN" or "MAX")` has two pre-conditions.

(1) The string must be either `"MIN"` or `"MAX"` – if not, the code will raise an assertion error.

(2) The `List[int]` must not be empty. If it is, the code should raise an assertion error.

See `main()` to see example usage of the minmaxList functions.

Running the code provided to you will give the following output.

```
Starting List:  [10, 11, 99, 1, 55, 100, 34, 88]
1st min: 1
2nd min: 10
1st max: 100
2nd max: 99
Insert 0 9 101 100
1st min: 0
2nd min: 9
1st max: 101
2nd max: 100
DONE.  Type Enter to exit.
```

See next page for Lab 7's TASKS

Lab 7's TASKs

TASK 1 – Checking Pre-condition for empty list [modify `getMinMax()`] (2 points)

Modify the function `getMinMax(List[int], str)` to include an **assert** statement to check the pre-condition that the list is not empty. There is already an **assert** checking if the str is either "MIN" or "MAX".

TASK 2 – Test functions for our code. [write five (5) Pytest functions]

Modify the starting code to include the following five (5) test functions for use with Pytest.
These functions should appear at the end of the current code.
Please name your functions exactly as shown below.

(1) `def test_getMinMaxCase1():`
This function will test a standard use case for our MinMaxList.
(a) Create a list with two items that are different.
(b) Call `initializeMinMaxList()` with (a).
(c) Use `getMinMax()` to get the mimimum item. Use an assert statement to check if this is correct. Error message should be "Min should be **x**", where **x** is the minimum item in the list specified in (a).
(d) Use `getMinMax()` to get the maximum item. Use an assert statement to check if this is correct. Error message should be "Max should be **y**", where **y** is the maximum item in the list specified in (a).

(2) `def test_getMinMaxCase2():`
This function will test an edge case where the list only has a single item.
(a) Create a list with only 1 item, let's call this item **y**.
(b) Call `initializeMinMaxList()` with (a).
(c) Use `getMinMax()` to get the minimum item (which is **y**). Use an assert statement to check if this is correct. Error message should be "Min should be **y**", where **y** is the single item in your list in (a).
(d) Use `insertItem()` to insert the same item **y** back into the list in (a).
(e) Use `getMinMax()` to get the maximum item (which is **y**). Use an assert statement to check if this is correct. Error message should be "Max should be **y**", where **y** is the maximum item in the list specified in (a).

(3) `def test_getMinMaxCase3():`
This function will test an edge case where the list starts out empty.
(a) Create an empty list.
(b) Call `initializeMinMaxList()` with (a).
(c) Insert an item **x** into (a) using `insertItem()`.
(d) Insert an item **y** into (a) using `insertItem()`. *Item **y** should be larger than **x**.*
(e) Use `getMinMax()` to get the minimum item. Use an assert statement to check if this is correct. Error message should be "Min should be **x**", where **x** is the minimum item inserted into (a).
(f) Use `getMinMax()` to get the maximum item. Use an assert statement to check if this is correct. Error message should be "Max should be **y**", where **y** is the maximum item inserted into (a).

(4) `def test_getMinMaxRequestError()`
This function will test to see if `getMinMax()` properly causes an assertion error when the string argument is not correct.

(a) Create a list with 3 items.
(b) Call `initializeMinMaxList()` with (a).
(c) Call `getMinMax()` with a, but using `"MID"` instead of "MIN" or "MAX". This will cause `getMinMax()` to raise an AssertionError.
(d) Check if the AssertionError was raised. Assert on this condition, if the condition was not rasied, your error should be: "Should raise AssertionError!"

Continues on next page.

(5) `def test_getMinMaxEmptyError():`

This function will test to see if `getMinMax()` properly causes an assertion error when the list is empty.

(a) Create an empty list.
(b) Call `initializeMinMaxList()` with (a).
(c) Call `getMinMax().` If you did Task 1 correctly, this will cause `getMinMax()` to raise an AssertionError.
(d) Check if the AssertionError was raised. Assert on this condition, if the condition was not rasie, your error should be:
"Should raise AssertionError!"

VERIFYING YOU TEST FUNCTIONS WITH PYTEST

Remember to first install Pytest using the pip command as described in the notes. Now, verify that your test functions work by using "pytest lab7.py" from PyCharm's terminal.
To the best of our knowledge, the code provided to you does not have any bugs, so the 5 test should all pass if written correctly.

The expected output would look as follows.

```
(venv) C:\Users\mbrown\PycharmProjects\pythonProject1>pytest Lab7.py
==================== test session starts ====================
platform win32 -- Python 3.8.5, pytest-6.1.2, py-1.9.0, pluggy-0.13.1
rootdir: C:\Users\mbrown\PycharmProjects\pythonProject1
collected 5 items

Lab7.py .....
[100%]

==================== 5 passed in 0.04s ====================
```

NOTE: You can force a failure of your test cases by commenting out one of the assert statements in the function `getMinMax().`

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 "lab7.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 Tasks :

- 2 points for Task 1 (adding assert)
- 8 points for the Task 2 (adding in the test functions)

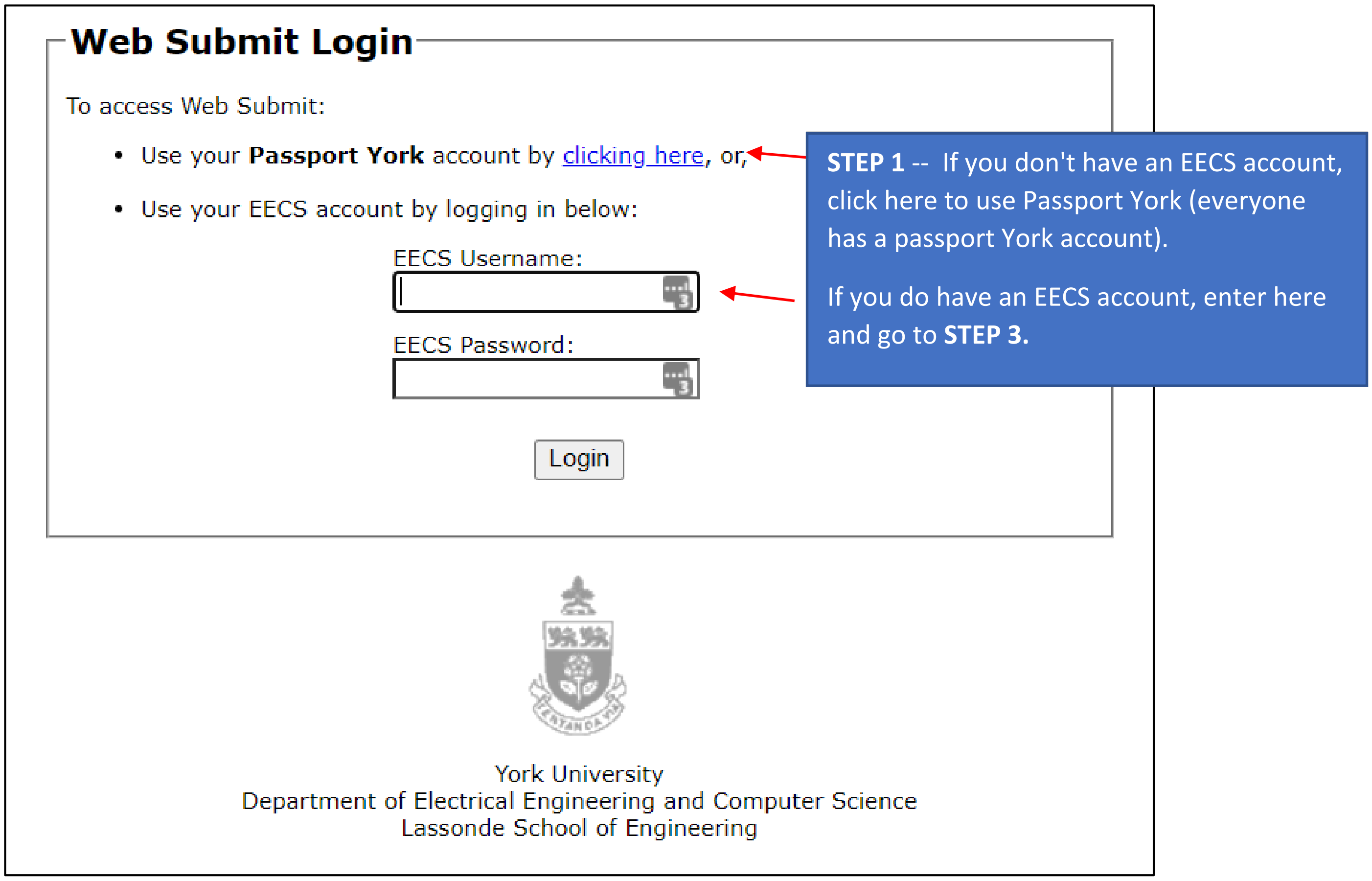
-No submission – 0 points
-Any submission up to 1 week after the due date 50% off the total marks
-Any submission after 1 week of the due date will not be marked and treated as no submission.

See pages below on how to submit your lab code.
MAKE SURE TO SELECT **Lab7** 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:

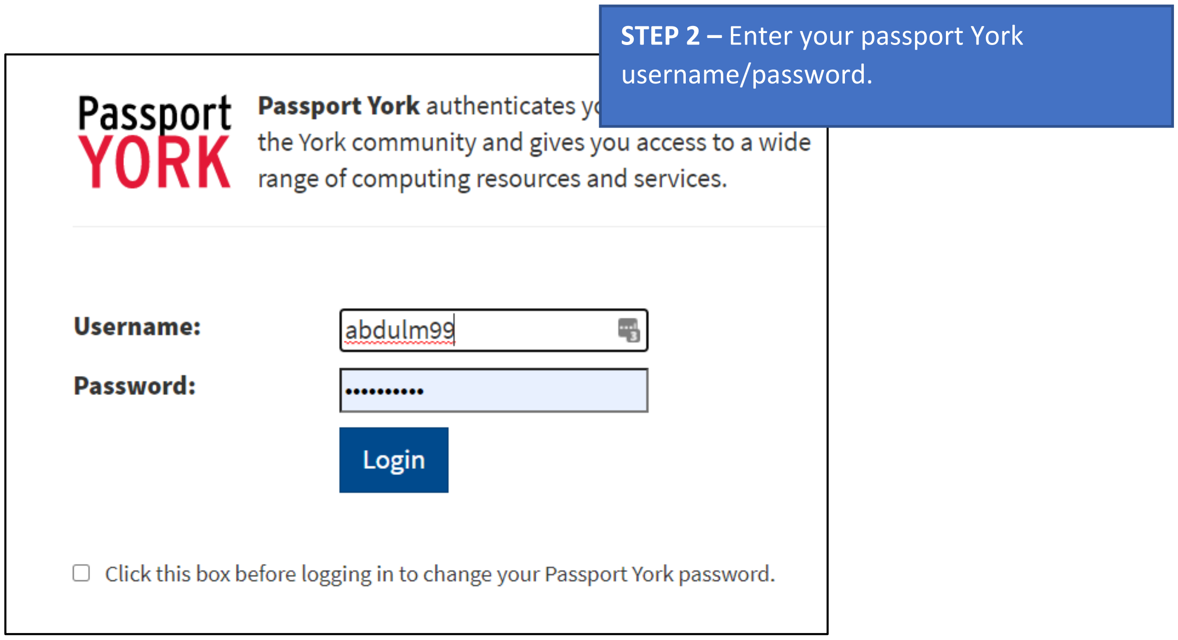
EECS Username:

EECS Password:

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.**



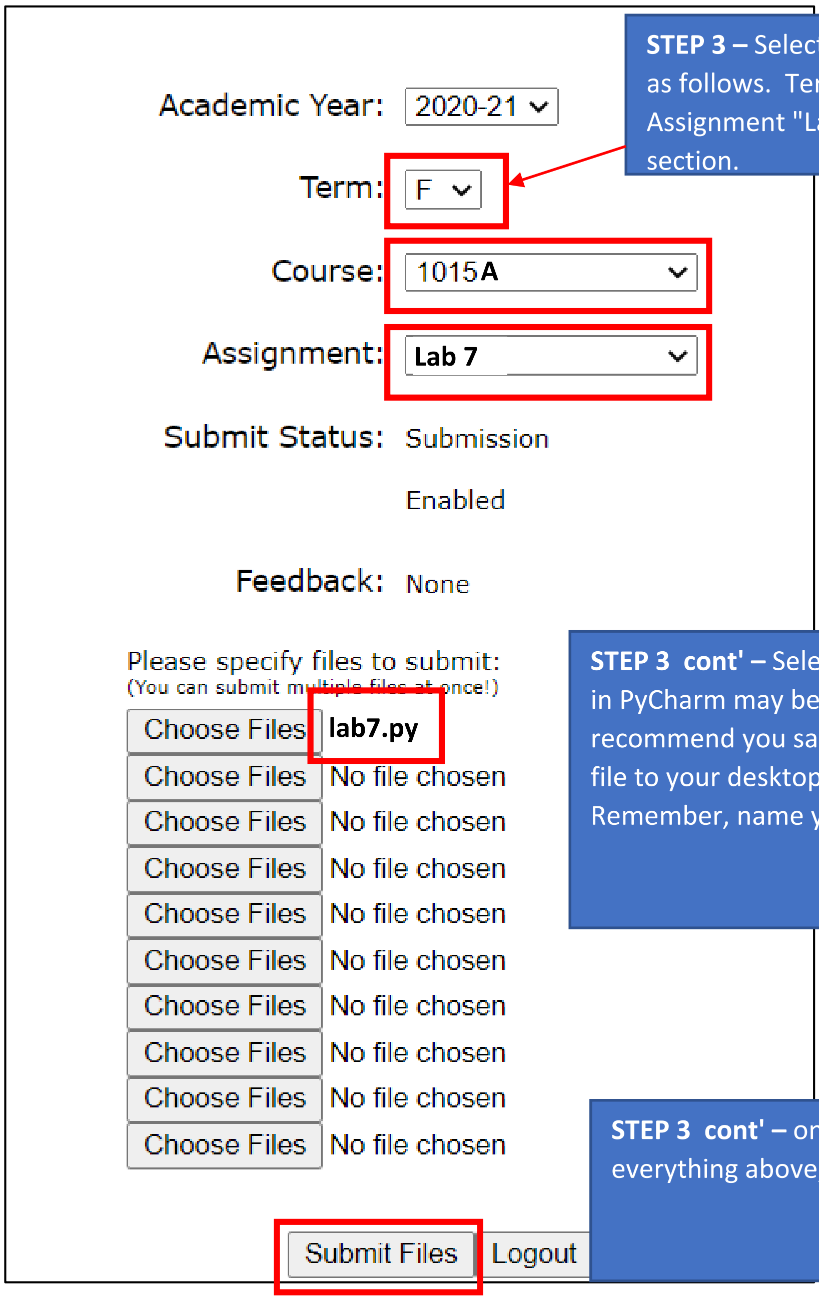
STEP 2 – Enter your passport York username/password.

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

Username: abdulm99

Password: ••••••••••

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



Academic Year: 2020-21

Term: F

Course: 1015A

Assignment: Lab 7

Submit Status: Submission Enabled

Feedback: None

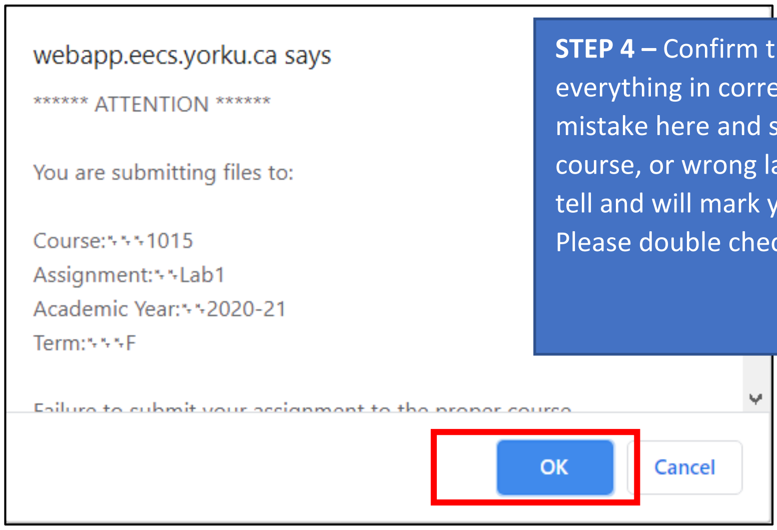
Please specify files to submit:
(You can submit multiple files at once!)

Choose Files	lab7.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 – Select the correct menu option as follows. Term "F", Course "1015", Assignment "Lab7". Selection course section.

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 **lab7.py**.

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



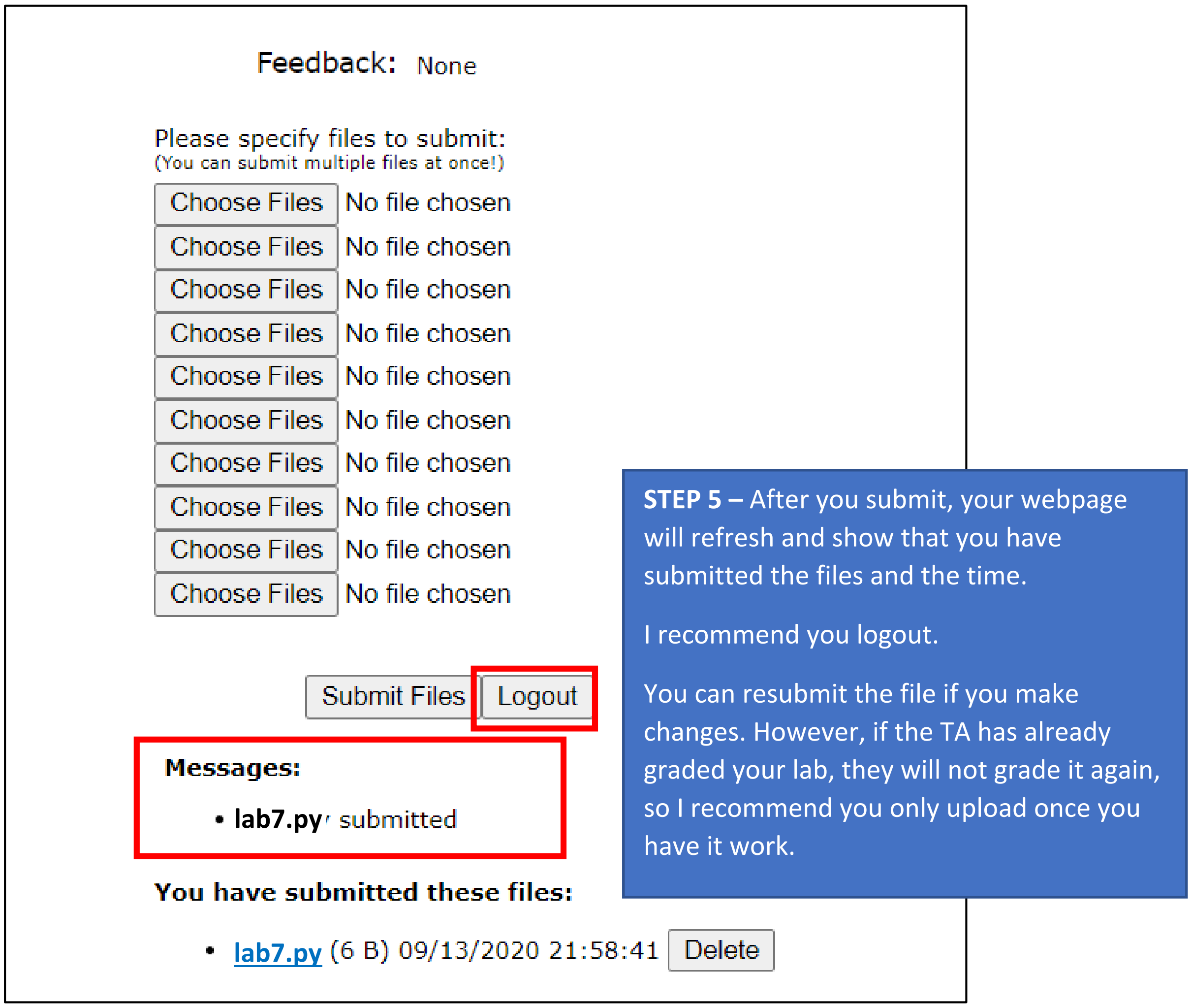
webapp.eecs.yorku.ca says

****** 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

Messages:

- **lab7.py** submitted

You have submitted these files:

- lab7.py (6 B) 09/13/2020 21:58:41 Delete

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