EECS 1015: LAB #6 – Lists, Dictionaries, and Tuples
Assigned: Nov 3, 2020
Due date: Nov 13, 2020 [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 – `lab6.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 lists, dictionaries, and tuples
- To use for-loops with lists, dictionaries, and tuples
- To continue using functions and control-statements

2. LAB 6 – 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 6
# 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_Lab6.mp4

This lab has three (3) tasks, each one has associated functions that need to be implemented.

See the explanation of the lab on the next page.

Lab 6 – Manipulating list of list

STARTING CODE LINK

This lab starts with skeleton code that you can find here: https://trinket.io/library/trinkets/277a60e993

The starting code defines several global variables (see link above):

```
# For task 1
rList = [[1, 10, 9, 4, 50],
         [3, 40, 99, 37, 5, 1],
        …]

# For task 2
encodedData1 = [[(9, ' '), (1, '.'), …], [(9,' '), (number, character),..]]
encodedData2 = [[(number, character), (number, character), …], [(number, character), (number, character),..]]

# For task 3
stringData = "1 H Hydrogen,2 He Helium,3 Li Lithium,4…"
```

We will explain how these variables will be used below for each Task.

TASK 1 – Printing and sorting a ragged list

A "ragged list" is a list of lists where the length of the lists nested inside the main list is not of the same size.

Variable `rList` is a ragged list that has already been defined for you (see above).

For Task 1, you need to implement and call the following two (2) functions:

`1) printRaggedList(param: list) -> no return`

```
Loop through each item in the ragged list (which is a list) and print out each list as follows:

Row 0: [item1, item2, item3, …, itemN]
Row 1: [item1, item2, …, itemN]
```

In Python, you can print a list after a formatted string as follows: `print("format string".format(arg1, arg2), list)`

`2) sortRaggedList(param: list) -> no return (but mutates list)`

This function will sort each list in the ragged list. The function should be passed the variable `rList`.

TASK 1 ACTION:
(1) Print the ragged list by passing `rList` as a parameter to `printRaggedList()`.

(2) Sort the ragged list by passing `rList` as a parameter to `sortRaggedList()`.

(3) Print the ragged list again after sorting using `printRaggedList()`.
-We should see that the contents of the lists are sorted.

TASK 2 – Print an encoded ASCII Art

This task processes the data in the variables `encodedData1` and `encodedData2.`
These variables are bound to a "list of lists of tuples". Specifically, each item in the list is another list, that list stores several tuples.
The tuples have two values, the first is a number, the second is a single character.
The tuple is encoding a "run" of characters. That is the number tells you how many times you could repeat the character.

The idea is that you should "decode" the list of tuples to construct a string that you can print out. Each list represents a single line of "Test Art" (or what we call ASCII Art).

See example here:

```
[[(5, '*')],                      # row 0      decodes to: *****  A run of 5 '*'
 [(2, ' '), (1, '*')],            # row 1      decodes to:   *    A run of 2 ' ', 1 '*'
 [(2, ' '), (1, '*')],            # row 2      decodes to:   *    A run of 2 ' ', 1 '*'
 [(1, ' '), (3, '*')]]            # row 3      decodes to:  ***   A run of 1 ' ', and 3 '*'
```

To perform this task, define two functions:

1) `decodeTupleList(param: list of tuples) -> string`

This will take a list of tuples in the form [(number, character), (number, character), …].

You should "decode" the list and its tuples to build a ***single*** string.
For example:

```
decodeTupleList( [ (5, '.'), (3, '-'), (5,'.') ] ) returns ".....---....."
 "."+"."+"."+"."+"."  + "-"+"-"+"-"  + "."+"."+"."+"."+"." -> ".....---....."
   5 "." chars           3 "-" chars      5 "." chars            final string
```

2) `printEncodedAsciiImage(param: list) -> no return`

This function will print the ASCII art encoded in lists bound to variables `encodedData1` and `encodedData2`.

When you pass the variable to this function, you should loop through the list. Recall that each item in the list is another list of tuples. Call `decodeTupleList(item)` to decode the string. `decoupleTupleList()` returns a string. Print the returned string. This will print the Ascii Art encoded one line at a time. When you are done, you should have a nice picture.

In the lab, I will only show you the result of `encodedData1` , you have to implement the program to see `encodedData2`.

Task 2 ACTIONS

(1) call `printEncodedAsciiImage(encodedData1)`

-An example of the output of this is shown in the video and in the output below.

(2) call `printEncodedAsciiImage(encodedData2)`

- You need output this too, but it is not shown (you have to implement the task to see it!)

TASK 3 – Element string to a dictionary

This task processes the data in the string variable `stringData.`

`stringData` is a long string that encodes the information on the first 25 elements from the periodic table.

To perform this task, define two functions:
`1) buildElementDictionary(param: string) -> dictionary`

This function processes the **stringData** to build a dictionary. The string has the following form.

```
"1 H Hydrogen,2 He Helium,3 Li Lithium,4 Be Beryllium,5 B Boron,6 C Carbon,…"
```

Split the string to get a list of each element as follows:
`["1 H Hydrogen", "2 He Helium", "3 Li Lithium, …]`
Now, for each string in this list, split it to get
`"1", "H", "Hydrogen"`

Add this information to your dictionary as follows:
`key='H', value = ['Hydrogen' ,'1']  i.e. {'H', ['Hydrogen', '1']}`
`key='He', value = ['Helium', '2']  i.e. {'He', 'Helium', '2']}`

Process each element and return the final dictionary with all 25 elements.

2) `printElements(param: dictionary) -> no return`

This function takes the dictionary created by `buildElementDictionary()` as a parameter.
Print out the dictionary as follows:

```
H [Hydrogen] #1
He [Helium] #2
Li [Lithium] #3
Be [Beryllium] #4
```

'H' is the key to the dictionary. Hydrogen and '1' are the 1st and 2nd items in the list paired with the key.

Task 3 ACTIONS

(1) Call `buildElementDictionary(stringData)`

- This will generated the dictionary from the stringData that stores the elements.

(2) Print the dictionary using `print()`.

- Print the dictionary out so its contents and verify that it is OK.

(3) Call `printElements()` by passing your dictionary.

- This will print out the contents as described above.

Finally, put all your tasks in the `main()` function.

`main(parameters: none) -> no return`

Your main function will be used to test the functionality above. The skeleton code for your main() is:

```
def main():
  print("Task 1 - Sorting and printing a ragged list ")
  print("Task 2 - Decoding Ascii Art ")
  print("Task 3 - Elements String to Dictionary ")
```

See the next page for an example output of Lab 6.

```
Task 1 - Sorting and printing a ragged list
--List before sorting--
Row 0: [1, 10, 9, 4, 50]
Row 1: [3, 40, 99, 37, 5, 1]
Row 2: [8, 11, 10, 94]
Row 3: [100, 9, 2, 88, 44]
Row 4: [4, 9, 2, 19]
--List after sorting--
Row 0: [1, 4, 9, 10, 50]
Row 1: [1, 3, 5, 37, 40, 99]
Row 2: [8, 10, 11, 94]
Row 3: [2, 9, 44, 88, 100]
Row 4: [2, 4, 9, 19]
```

Result of calling `printRaggedList()`.

Caling `printRaggedList()` again after calling `sortRaggedList()`.

```
Task 2 - Decodng Ascii Art
         .8.
         888
         888l
        j8888.
       .888888.
      .88888888.
    .d8888888888b.
  .d88888888888888b.
 .8888888888888888888b.
.888888888888888888888
88888888888888888888888
888P""4888
`P'     .    .     `q'
 `-..____:  :____..-'
         :  :
         :  :
         :  :
         :  :
         :  :
       \(/\)\/ mh
```

Result of calling `printEncodedAsciiImage(encodedData1)`.

Each line of the "image" was produced by calling `decodeTupleList()`.

IMPORTANT. You also need to include the output of `printEncodedAsciiImage(encodedData2)`.

Please call this function again after the first `printEncodedAsciiImage(encodedData1)`.

```
Task 3 - Elements String to Dictionary
{'H': ['Hydrogen', '1'], 'He': ['Helium', '2'], 'Li': ['Lithium', '3'], 'Be': ['Beryllium', '4'], 'B': ['Boron', '5'],
'C': ['Carbon', '6'], 'N': ['Nitrogen', '7'], 'O': ['Oxygen', '8'], 'F': ['Fluorine', '9'], 'Ne': ['Neon', '10'], 'Na':
['Sodium', '11'], 'Mg': ['Magnesium', '12'], 'Al': ['Aluminum', '13'], 'Si': ['Silicon', '14'], 'P': ['Phosphorus', '15'],
'S': ['Sulfur', '16'], 'Cl': ['Chlorine', '17'], 'Ar': ['Argon', '18'], 'K': ['Potassium', '19'], 'Ca': ['Calcium', '20'],
'Sc': ['Scandium', '21'], 'Ti': ['Titanium', '22'], 'V': ['Vanadium', '23'], 'Cr': ['Chromium', '24'], 'Mn': ['Manganese',
'25']}
---First 25 Elements---
H [Hydrogen] #1
He [Helium] #2
Li [Lithium] #3
Be [Beryllium] #4
B [Boron] #5
C [Carbon] #6
N [Nitrogen] #7
O [Oxygen] #8
F [Fluorine] #9
Ne [Neon] #10
Na [Sodium] #11
Mg [Magnesium] #12
Al [Aluminum] #13
Si [Silicon] #14
P [Phosphorus] #15
S [Sulfur] #16
Cl [Chlorine] #17
Ar [Argon] #18
K [Potassium] #19
Ca [Calcium] #20
Sc [Scandium] #21
Ti [Titanium] #22
V [Vanadium] #23
Cr [Chromium] #24
Mn [Manganese] #25
```

Dictionary produced by calling function: `buildElementDictionary(stringData)`.

Print the dictionary out.

Result of calling `printElements()`.

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 "lab6.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 :

- 3 Tasks [-5 points for each that doesn't work properly]
- main function [-2 if the main doesn't work]
- You can't receive below a 0.

-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.

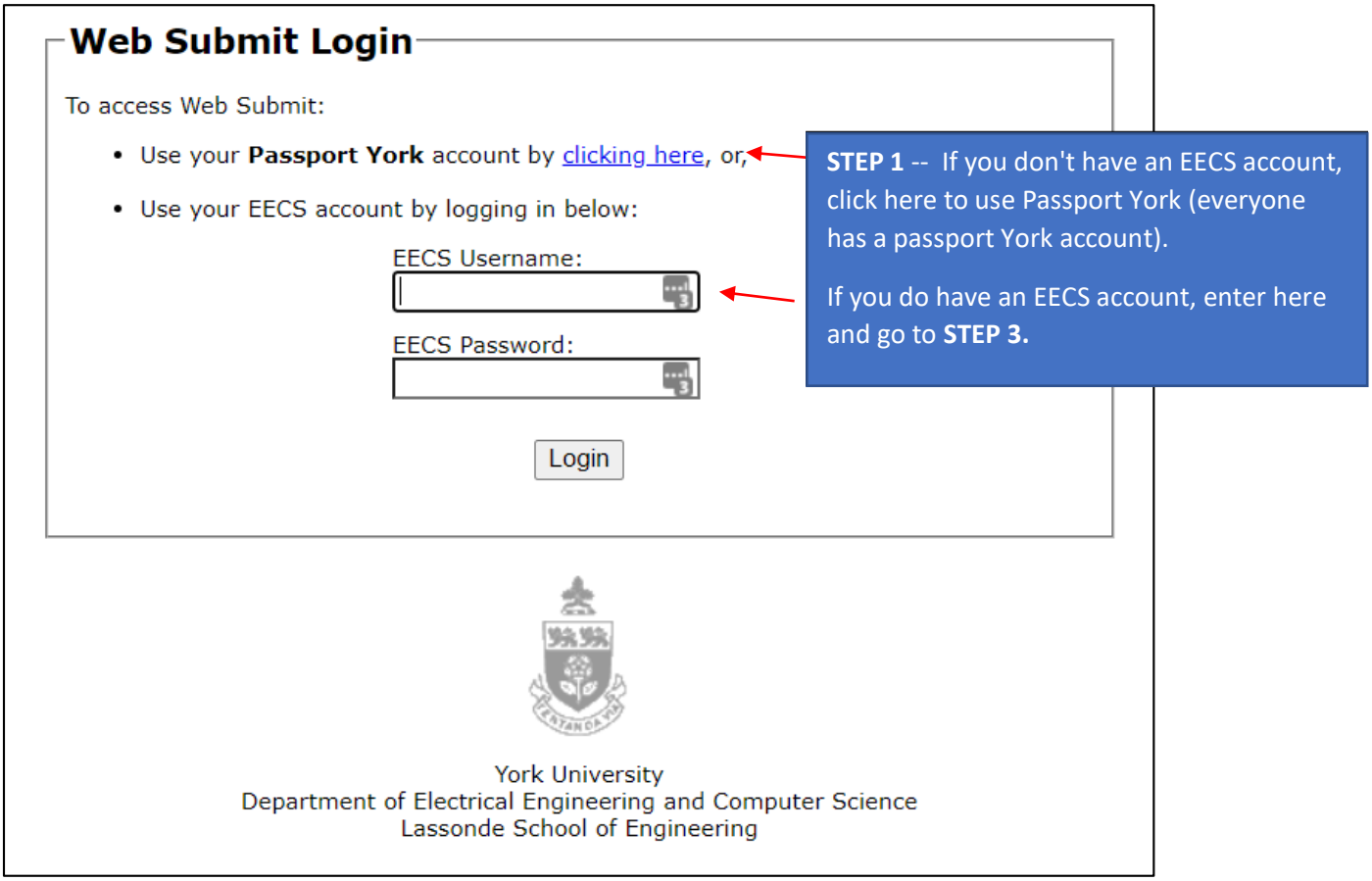
MAKE SURE TO SELECT **Lab6** with websubmit

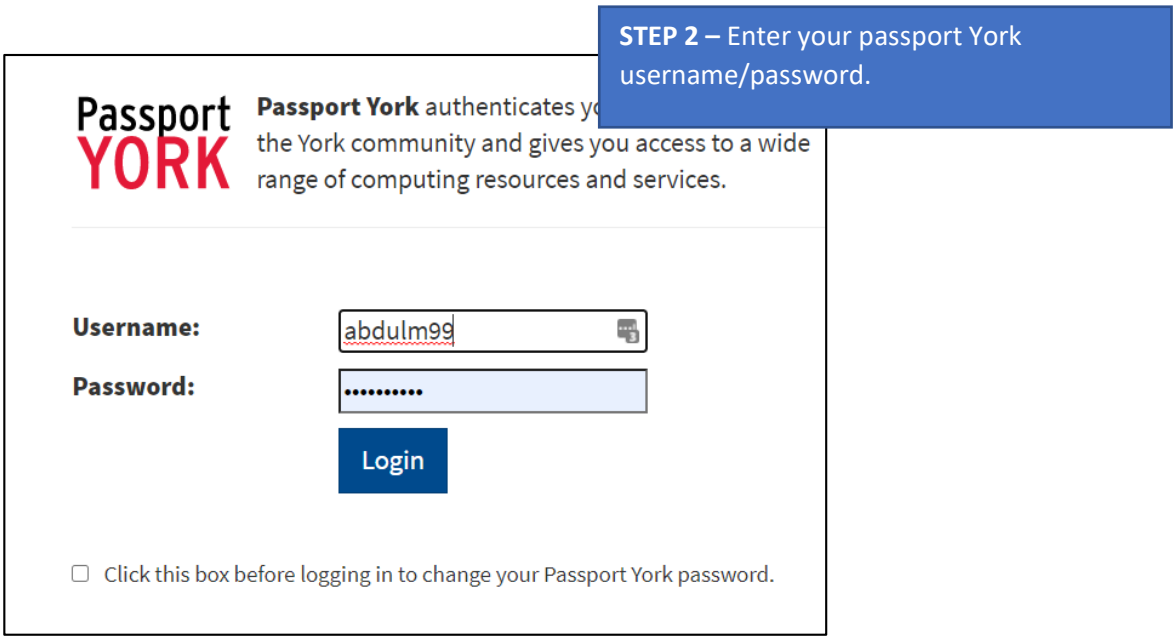
Note, if you use the new experimental testing platform it can perform websubmit for you!

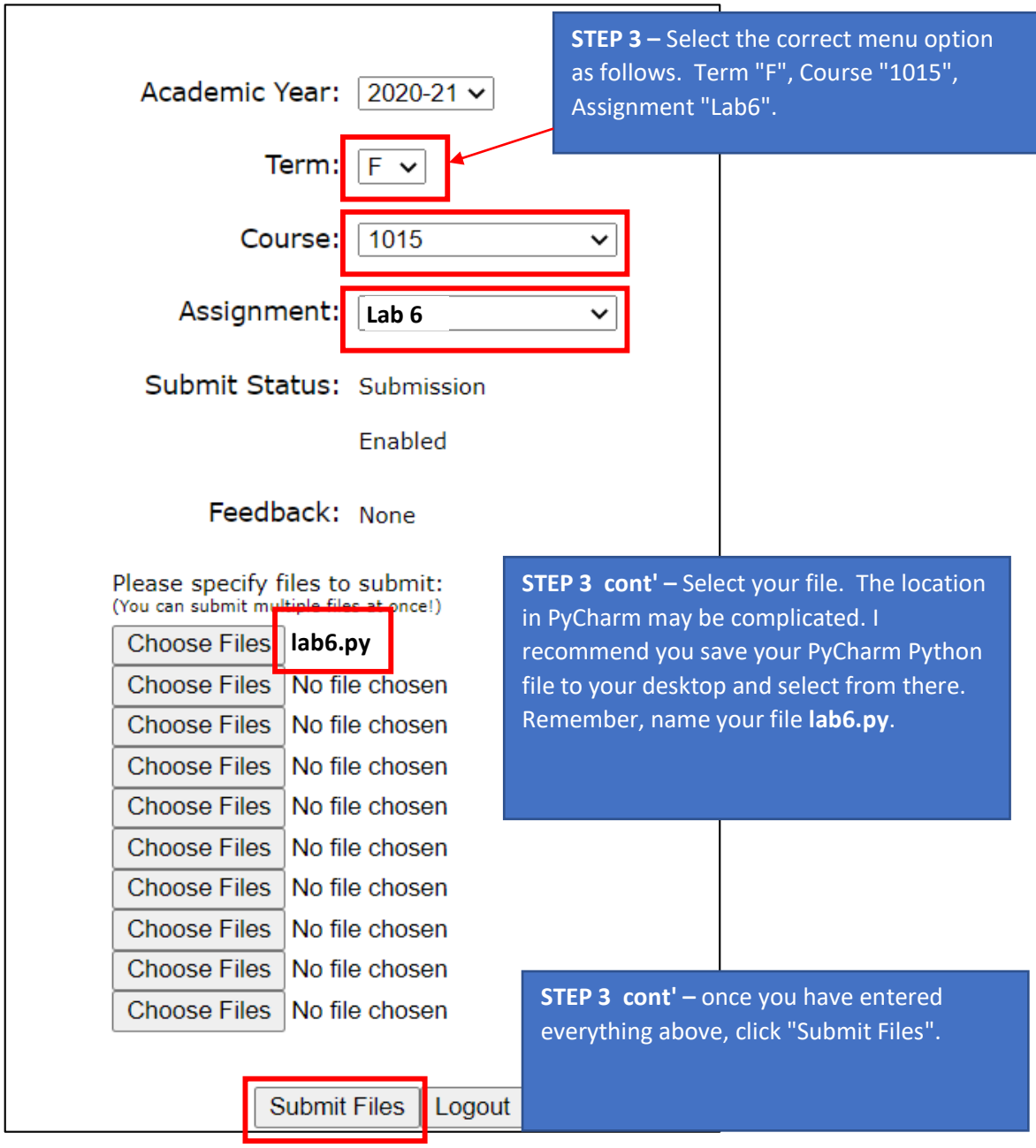
4. SUBMISSIONS (EECS web-submit)

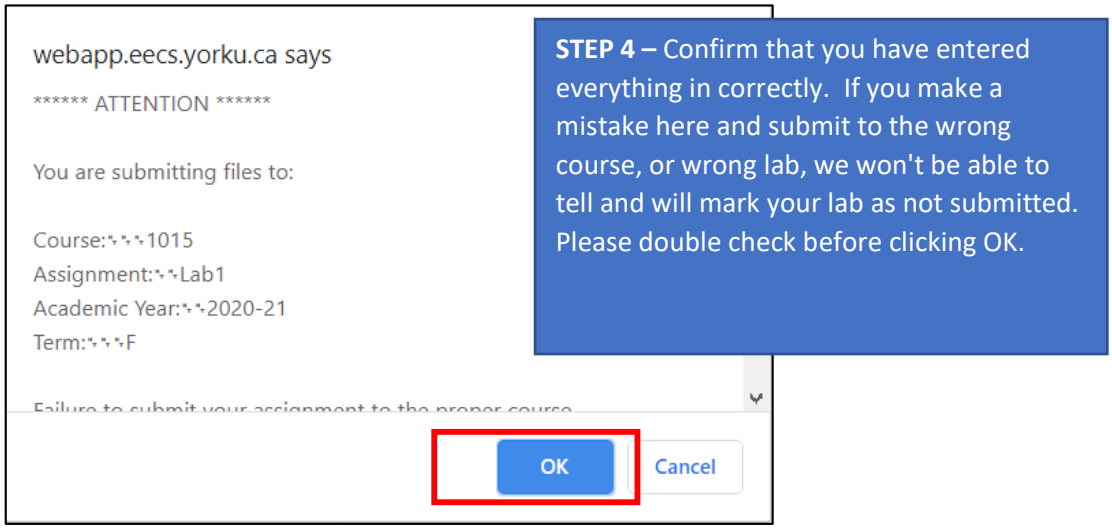
You will submit your lab using the EECS web submit.

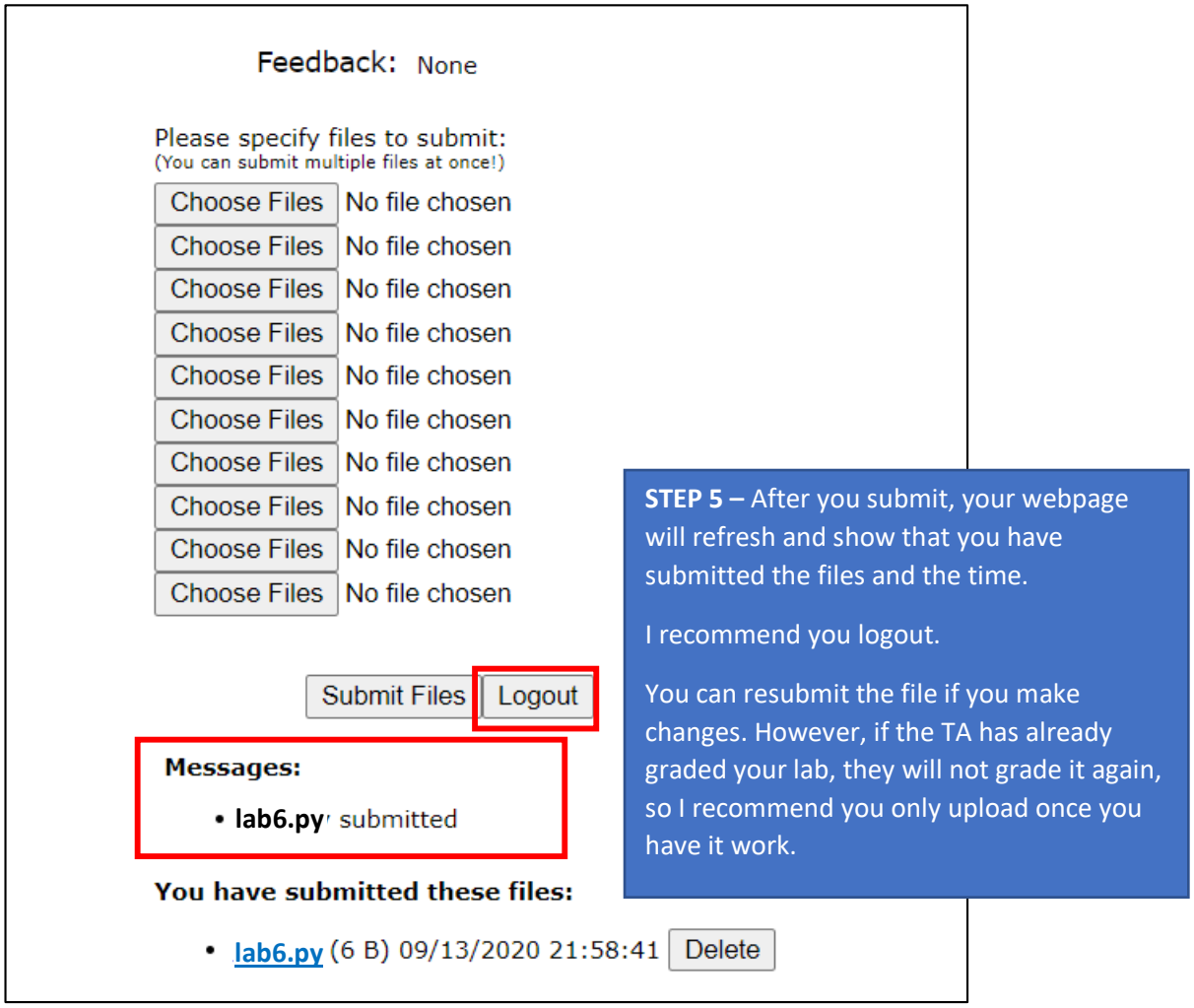
Click on the following URL: https://webapp.eecs.yorku.ca/submit






Academic Year: 2020-21
Term: F
Course: 1015
Assignment: Lab 6
Submit Status: Submission Enabled
Feedback: None
Please specify files to submit:
(You can submit multiple files at once!)
Choose Files lab6.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 "1015", Assignment "Lab6".
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 lab6.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