EECS 1015: Midterm programming exam (variables, expressions, strings, flow-control, functions)
Assigned: Oct 27, 2021 (Wed)
Due date: Oct 29, 2021 (Fri) [11.59 pm Eastern Time – No extensions/No late submissions!]

#Important for the midterm programming exam

1) You must submit your midterm via web-submit to the "midterm" folder (see instructions on last pages).

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

3) Please follow the instructions – read the task descriptions carefully to understand everything you need to do. There are five (5) tasks.

4) Please watch the accompanying video with this midterm. Make your output as similar as possible to what is shown in the video.

2. INSTRUCTIONS

1) You have three days to complete this take-home midterm. There are no extensions or late submissions.

2) Midterm is on Topics 2-5 from the lectures. ***The midterm does not have any questions that require Lists, Tuples, Sets, or Dictionaries.***

3) The midterm has five (5) tasks. Please complete each task.

4) This midterm is graded out of 100 points.

5) Please name your submission: midterm.py

Starting code available here: https://trinket.io/python/7a53c1ec41

A video describing the midterm is available here:
http://www.eecs.yorku.ca/~mbrown/EECS1015_midterm.mp4

See next pages for each task's description.

TASK 0 (0 points correct, -10 points deducted if not done correctly)

1) Place your information in the comments at the beginning of your python file. Please include the standard information in the comments as follows:

```
# Name: Your name
# Student ID: XXXXXXX
# Email XXXXX
# Section A or B
```

2) Function `task0()` should print out the following:

```
Midterm Exam - EECS1015
Name: Your Name
Student ID: Your York ID
email: youremail@aol.com
Section A or B
```

Why the redundant information? We have software that helps us with grading. That software looks only in the comments. The function `task0()` helps the instructor and TA to verify your name visually while grading.

TASK 1 (15 points) *[Testing: Variables and expression, strings, input, output formatted text, loop]*

Function `task1()` computes the compound return of an investment over five years for a given starting amount and fixed annual return. Your function should work as follows:

(1) Input the user's first name
(2) Input the user's last name
(3) Input the initial amount of funds from the user (assume a floating-point input)
(4) Input the annual return of the investment (assume a floating-point)
(5) Print the user's name with the first letter capitalized for the first name and all uppercase for the last name.
(6) Print out the initial deposit amount (from input (3)).
(7) Print out the compound return for years 1-5.

Each year the investment amount will increase based on the following formula:

$$amount = amount + amount \times \frac{annual\ return}{100}$$

Your output should look like the examples below. Pay attention to how the name is processed.

Example of task running (user input in red).

```
----------Task 1-----------
Your first name:    michael
Your last name:    Brown
Initial funds to invest: $515.50
Annual return percentage: 12.5
Yearly return for Michael BROWN
Initial deposit: $515.50
Year 1: $579.94
Year 2: $652.43
Year 3: $733.98
Year 4: $825.73
Year 5: $928.95
```

For the first and last name, the user may input additional spaces that you should remove.

Input for initial fund amount and annual return is a float.

Make sure to properly format the first and last name output. First name should be first letter uppercase, last name all uppercase.

```
----------Task 1-----------
Your first name: jean-philip
Your last name:  lalonde
Initial funds to invest: $100
Annual return percentage: 5.1
Yearly return for Jean-Philip LALONDE
Initial deposit: $100.00
Year 1: $105.10
Year 2: $110.46
Year 3: $116.09
Year 4: $122.01
Year 5: $128.24
```

Print out all dollar amounts with two decimal places. Make sure to verify that your code is working properly when computing the annual returns. For example, test your inputs with **100** and **5.1** to ensure you get the same results shown here.

TASK 2 (30 points) [*Testing: Variables and expression, strings, conditional-statements, loops*]

Function `task2()` emulates purchasing a soda from a vending machine. A soda costs $1.00. The user can input any five Canadian coins ($2, $1, $0.25, $0.10, and $0.05). The user continues to select coins until $1.00 is reached. Any amount over $1.00 should be returned. Your function should work as follows:

(1) Print welcome message and set current amount to 0.
(2) Print out the current amount (see below).
(3) Ask the user to input a selection between 1-5, where each selection corresponds to a Canadian coin as follows: 1 – Toonie ($2); 2 – Loonie ($1); 3 – Quarter ($0.25); 4 – Dime ($0.10); 5 – Nickel ($0.05)[1]
(5) Check the user's input to see what amount they entered. If they select a number that is not 1-5, print out "Invalid Selection." If a valid number is entered, add the corresponding amount to the current amount.
(6) If the current amount is less than $1.00, go back to (2)
(7) If the current amount is $1.00 or more, stop asking for input. Print out the total amount entered.
(8) Print out a thank you message for purchasing a soda.
(9) If the final amount is over $1.00, print "take your change $X.XX" where X.XX is the amount over $1.00.

Example of task running (user input in red).

```
----------Task 2-----------
Soda Vending Machine
Current amount $0.00 out of $1.00
Insert Coin
1. Toonie  ($2.00)
2. Loonie  ($1.00)
3. Quarter ($0.25)
4. Dime    ($0.10)
5. Nickel  ($0.05)
Selection [1-5]? 3
Current amount $0.25 out of $1.00
Insert Coin
1. Toonie  ($2.00)
2. Loonie  ($1.00)
3. Quarter ($0.25)
4. Dime    ($0.10)
5. Nickel  ($0.05)
Selection [1-5]? 4
Current amount $0.35 out of $1.00
Insert Coin
1. Toonie  ($2.00)
2. Loonie  ($1.00)
3. Quarter ($0.25)
4. Dime    ($0.10)
5. Nickel  ($0.05)
Selection [1-5]? 1
Total amount provided: $2.35
Thank you for your purchase.
Please take your change $1.35
```

Print welcome and current amount. All amounts should be shown with two decimal places.

Print coin selection menu as shown. Input the user's selection.

Add the selected amount to the current amount. If this is still less than $1.00, keep asking for more inputs. Each time, show the updated amount.

When the current amount is $1.00 or more, stop asking for input. Print out the total amount provided. Print a thank you message as shown. Any amount over $1.00 should be returned as change. Print a message to the user to take their change as shown.

[1] For those of you who are new to Canada, these are the names of the Canadian coins. There is a $0.50 coin, but it no longer used.

Task 2 – examples continued

```
----------Task 2-----------
Soda Vending Machine
Current amount $0.00 out of $1.00
Insert Coin
1. Toonie  ($2.00)
2. Loonie  ($1.00)
3. Quarter ($0.25)
4. Dime    ($0.10)
5. Nickle  ($0.05)
Selection [1-5]? 3
Current amount $0.25 out of $1.00
Insert Coin
1. Toonie  ($2.00)
2. Loonie  ($1.00)
3. Quarter ($0.25)
4. Dime    ($0.10)
5. Nickle  ($0.05)
Selection [1-5]? 9
Invalid selection!
Current amount $0.25 out of $1.00
Insert Coin
1. Toonie  ($2.00)
2. Loonie  ($1.00)
3. Quarter ($0.25)
4. Dime    ($0.10)
5. Nickle  ($0.05)
Selection [1-5]? 3
Current amount $0.50 out of $1.00
Insert Coin
1. Toonie  ($2.00)
2. Loonie  ($1.00)
3. Quarter ($0.25)
4. Dime    ($0.10)
5. Nickle  ($0.05)
Selection [1-5]? 3
Current amount $0.75 out of $1.00
Insert Coin
1. Toonie  ($2.00)
2. Loonie  ($1.00)
3. Quarter ($0.25)
4. Dime    ($0.10)
5. Nickle  ($0.05)
Selection [1-5]? 3
Total amount provided: $1.00
Thank you for your purchase.
```

If input is not 1-5, then print "`Invalid selection!.`" Note that the amount will not be updated.

When exactly $1.00 is provided, you do not need to output "Please take your change" as shown in the previous example above.

TASK 3 (25 points) [*Testing: Variables and expression, strings, conditional-statements, loops*]

Function `task3()` is a simple dice game that roles a dice ten times. The user wins if they have more than 35 points. The number of points is the sum of all the dice rolls and a bonus of 10 points if exactly two of the dice rolls are ones.

Implement your dice game as follows:
(1) Print a welcome message
(2) Output the result of 10 dice rolls using `randint(1,6)`—print roll # and dice value as shown below.
(3) Sum up the values of rolls 1-10.
(4) If two of the rolls were the number 1, then print out "+10 Bonus for snake eyes [1][1]!".
Add 10 points to the sum—[Note that more than two "ones" do not get a bonus].
(5) If the final points (including bonus) is greater than 35, the user wins; otherwise, they lose. Print out the appropriate message as shown below.
(6) Ask the user to input 'Y' to play again.
(7) If the user inputs either 'Y' or 'y', return to (1) and play the game again. Otherwise, the function is done.

Example of task running (user input in red).

```
----------Task 3-----------

Dice Game
Rolling Die 10 times
Roll 1: [6]
Roll 2: [2]
Roll 3: [6]
Roll 4: [6]
Roll 5: [5]
Roll 6: [1]
Roll 7: [1]
Roll 8: [1]
Roll 9: [3]
Roll 10: [6]
Total 37 -- OVER 35 POINTS [YOU WIN!]
Enter 'Y' to play again: y
Dice Game
Rolling Die 10 times
Roll 1: [6]
Roll 2: [1]
Roll 3: [4]
Roll 4: [5]
Roll 5: [2]
Roll 6: [1]
Roll 7: [4]
Roll 8: [3]
Roll 9: [1]
Roll 10: [2]
Total 29 -- TOO FEW POINTS [YOU LOSE!]
Enter 'Y' to play again: y
```

Print the dice values as shown. Print roll # (starting from 1) and the dice value in [].

More than two [1] does not result in the "snake eye" bonus of 10 points.

After showing the two rolls results, sum up the value of all the dice. If it is more than 35 the user wins

Ask the user to enter 'Y' to play again.
if 'Y' or 'y' is entered, continue to play.

Here the sum for the second game is 35 or less. The user loses.

```
Dice Game
Rolling Die 10 times
Roll 1: [5]
Roll 2: [6]
Roll 3: [1]
Roll 4: [3]
Roll 5: [3]
Roll 6: [4]
Roll 7: [5]
Roll 8: [1]
Roll 9: [6]
Roll 10: [6]
+10 Bonus for snake eyes [1][1]!
Total 50 -- OVER 35 POINTS [YOU WIN!]
Enter 'Y' to play again: n
```

If two of the dice are [1], then the user gets +10 bonus points for "snake eyes" as shown.

Make sure to print to the "snake eyes" message as shown here. At 10 points to the dice total.

Stop playing when anything other than 'Y' or 'y' is entered.

Task 4 (30 points) [*Testing: Variables and expression, strings, conditional-statements, loops, functions*]

Function `task4()` shows the results of several custom string manipulation functions you will write. For this task, you need to define the following three functions:

`countCases(param: string) returns two integer values`

This function takes a string and returns two values. The first value is the number of uppercase letters in the string. The second value is the number of lowercase letters in the string.
For example: "EECS1015 Fall 2021" has 5 uppercase (red) and 3 lowercase (green).

`flipCase(param: string) returns a string`

This function takes a string and returns a new string where the cases of the string are swapped.
For example: "EECS1015 Fall 2021" would be converted to "eecs1015 fALL 2021"

`cutQuotedText(param: string) returns a string`

This function takes a string with a *single* "word" in ***doubles*** quotes and removes all characters within the quote, including the quotes. To simplify things, we will assume that only two " characters appear in the string. If exactly two quote characters are not in the string, we will return a failure string message as shown below:
For example (proper input):
Input to function: 'I'm taking "EECS1015" this semester.'
Return string: 'I'm taking this semester.' <- quoted text has been removed.

If there isn't exactly one quoted text (i.e., only two quotes), return 'ERROR! No quoted text.'
For example (failure input):
Input to function: 'EECS1015'
Return string: 'ERROR! No quoted text.'
Another failure example:
Input to function: 'This "is" a test".' <- considered a failure because more than two "
Return string: 'ERROR! No quoted text.'

Using the three functions above, implement `task4()` function as follows:
(1) Prompt the user to enter a long string with one quoted word.
(2) Pass string to `countCases()` function.
(3) Pass string to `flipCase()` function.
(4) Pass string to `cutQuotedText()` function.
(5) Print out the results of the functions as shown below.

```
Example of task running (user input in red).

----------Task 4-----------

Enter string with one word with "quotes": This is a "test" input.
This string has 1 uppercase characters.
This string has 15 lowercase characters.
Case flip: 'tHIS IS A "TEST" INPUT.'
Quote removed: 'This is a  input.'
```

Print case flip and quote removed results with single quotes around the strings.

(more examples of task 4 on next page)

```
----------Task 4-----------
Enter string with one word with "quotes": Another "example" of good input.
This string has 1 uppercase characters.
This string has 24 lowercase characters.
Case flip: 'aNOTHER "EXAMPLE" OF GOOD INPUT.'
Quote removed: 'Another  of good input.'

----------Task 4-----------
Enter string with one word with "quotes": Example with BAD input.
This string has 4 uppercase characters.
This string has 15 lowercase characters.
Case flip: 'eXAMPLE WITH bad INPUT. '
Quote removed: 'ERROR! No quoted text.'

----------Task 4-----------
Enter string with one word with "quotes": Example with "more" than two " chars.
This string has 1 upper case characters
This string has 26 lower case characters
Case flip: 'eXAMPLE WITH "MORE" THAN TWO " CHARS.'
Quote removed: 'ERROR! No quoted text.'
```

Last two examples above shows input to the `cutQuotedText()` function that return error strings.

FINAL COMMENTS:

For tasks 2-5 you may introduce additional functions; however, it is not required and will not affect your grade either way. For Task 5, you <u>must</u> define the three functions as required in the task's description. All 5 tasks must be written in a single file named "midterm.py". All tasks should run one after the other, as shown in the accompanying video.

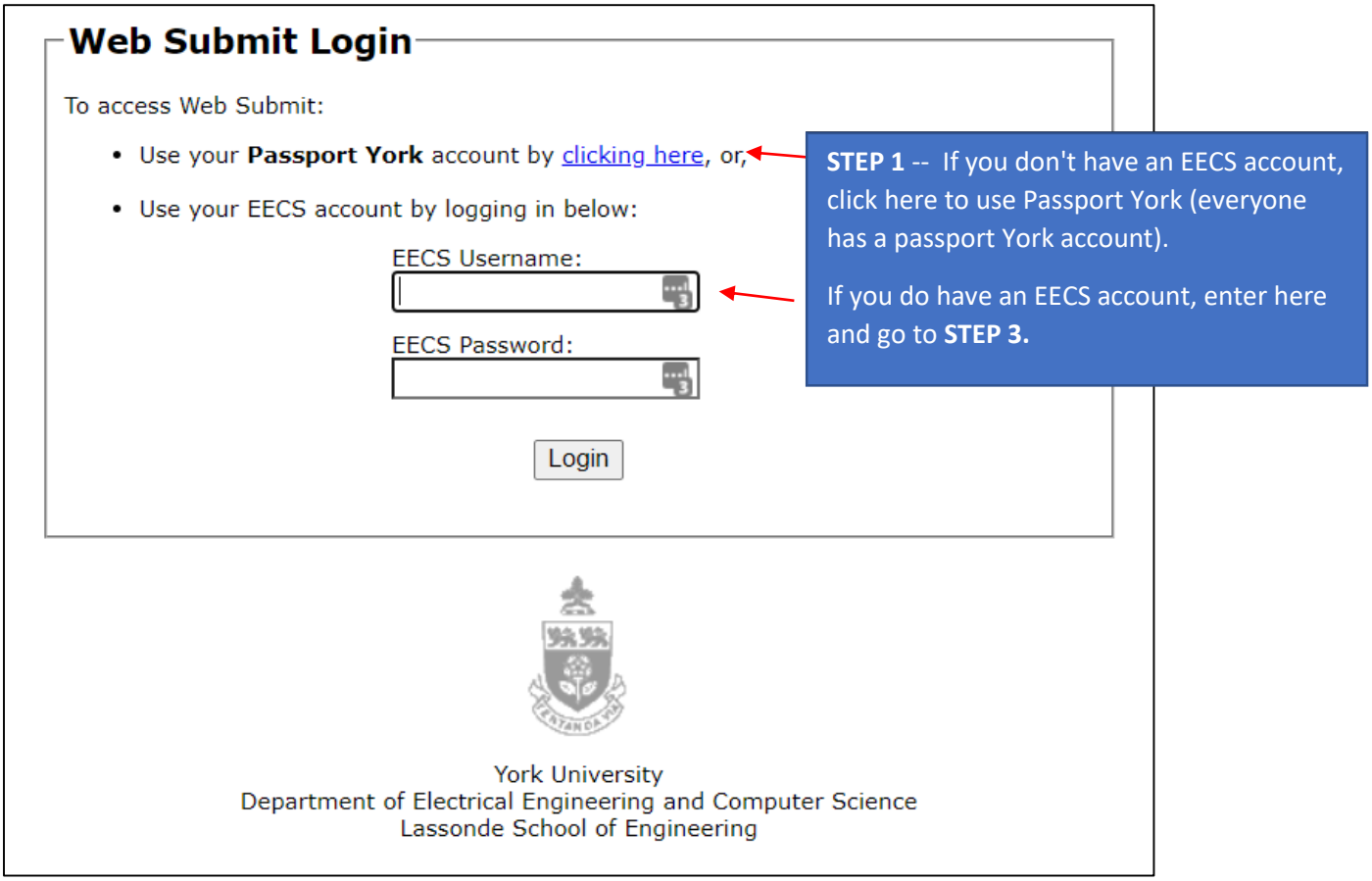
See pages below on how to submit your midterm code.

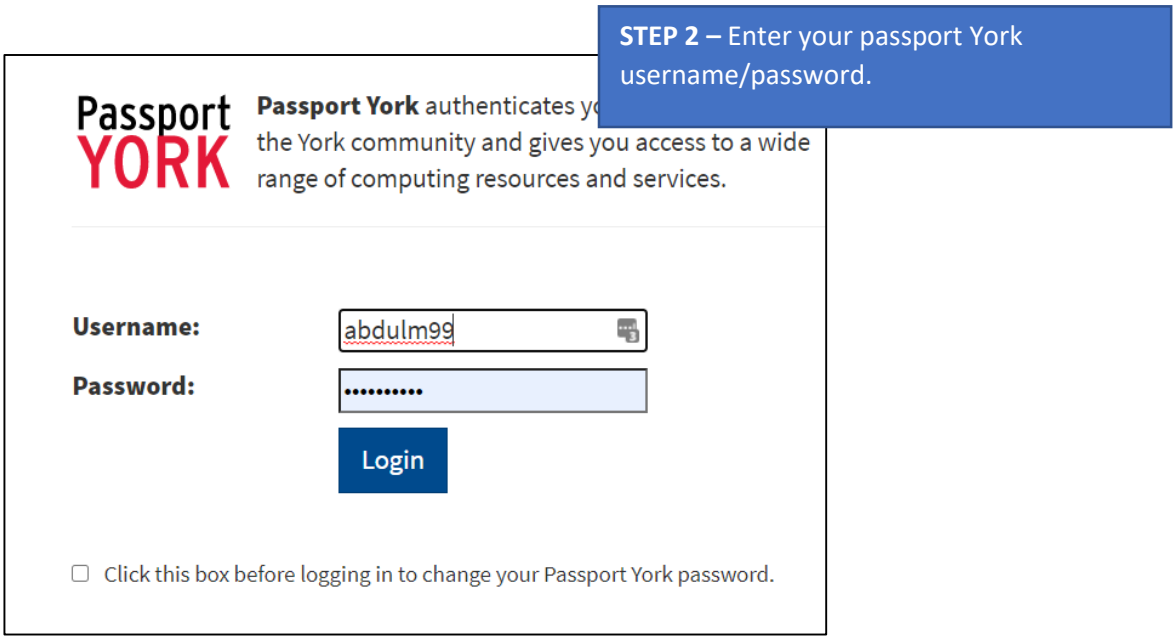
MAKE SURE TO SELECT **midterm** with websubmit.

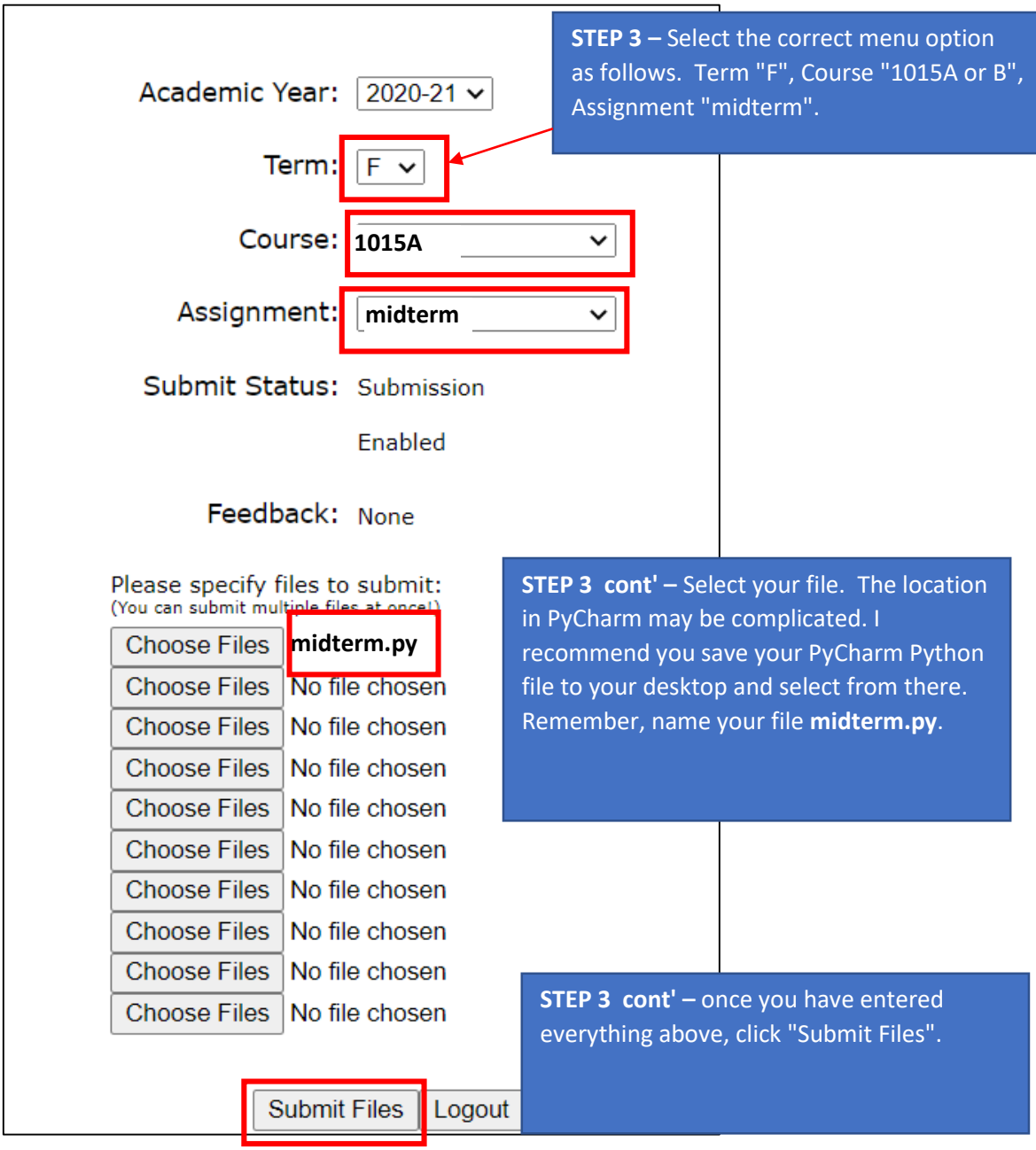
3. SUBMISSIONS (EECS web-submit)

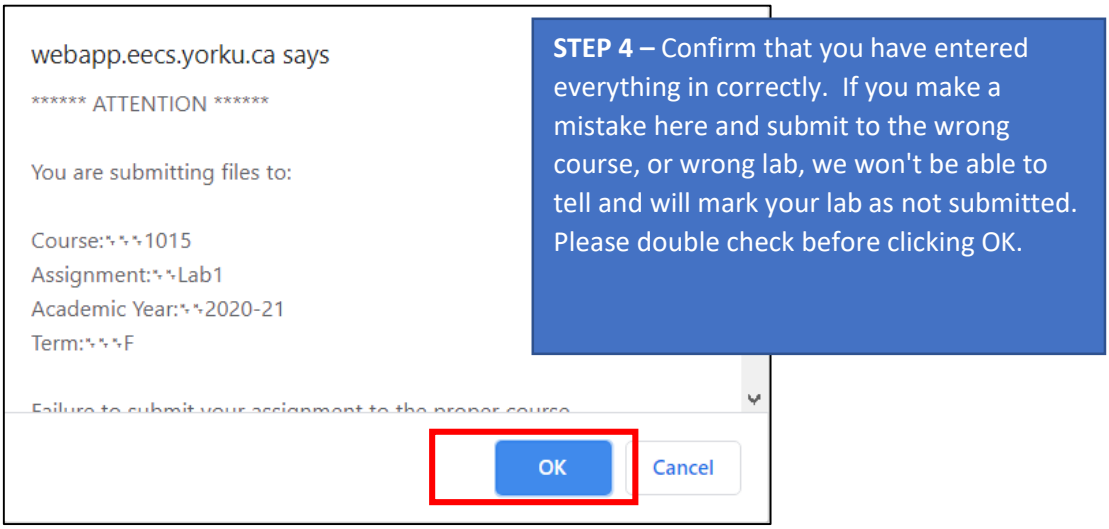
You will submit your lab using the EECS web submit.

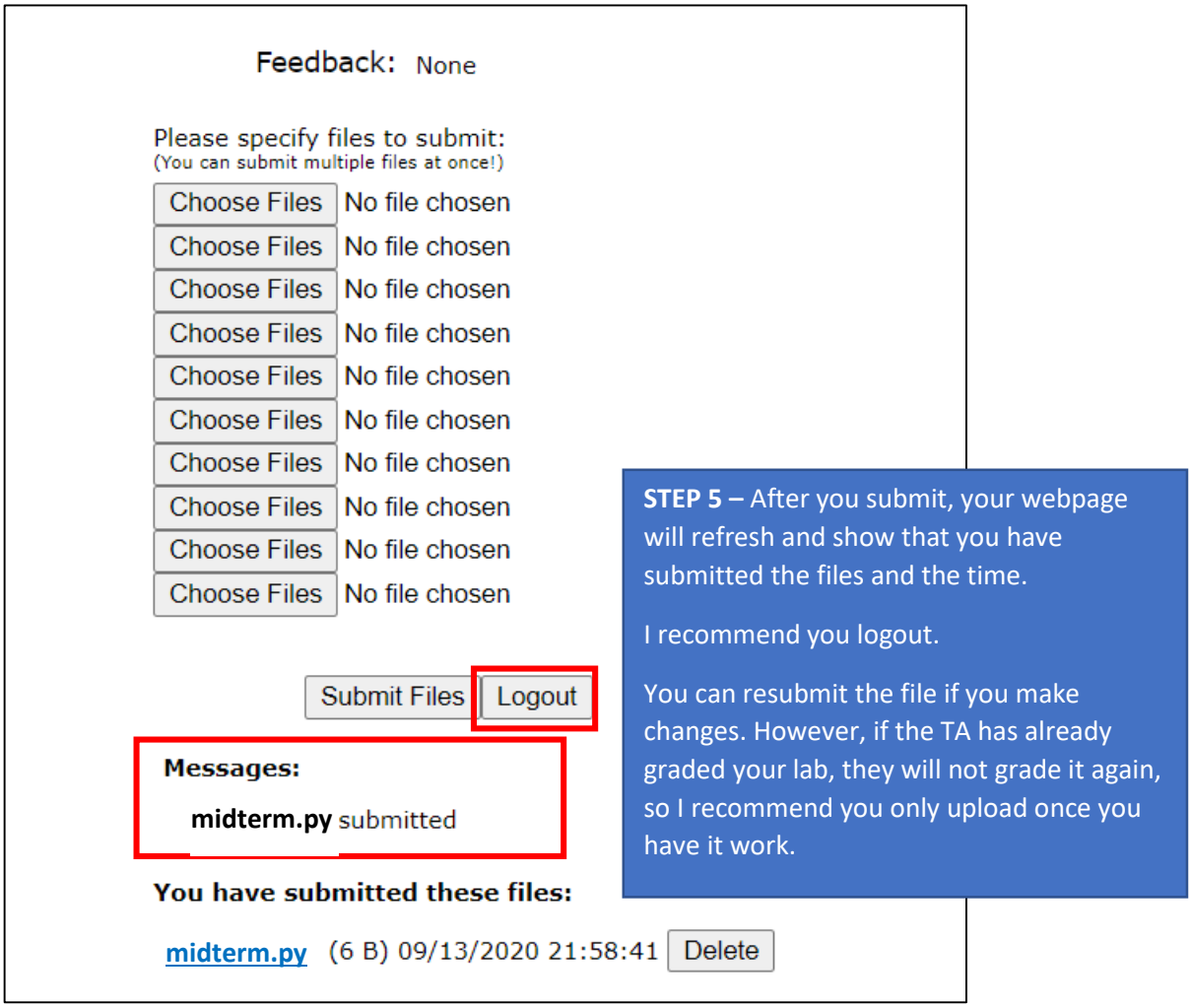
Click on the follow URL: https://webapp.eecs.yorku.ca/submit






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