

EECS 1015: Midterm programming exam (Variables, expressions, strings, flow-control, functions)

Assigned: Oct 26, 2020 (Mon)

Due date: Oct 30, 2020 (Fri) [11.59pm 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 carefully – read the task descriptions carefully to understand everything you need to do. There are five (5) tasks.

2. INSTRUCTIONS

- 1) You have 5 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 requires Lists, Tuples, or Dictionaries.***
- 3) The midterm has five (5) tasks. Please complete each task.
- 4) This midterm is graded out of 100 points.
- 5) IMPORTANT: Your program must have the following code below (you can cut and paste it from the trinket.io link provided).
- 6) Please name your submission: **midterm.py**

```
# main function for EECS1015 midterm

def main():
    task0()
    print("\n-- TASK 1 --\n")
    task1()
    print("\n-- Task 2 --\n")
    task2()
    print("\n-- Task 3 --\n")
    task3()
    print("\n-- Task 4 --\n")
    task4()

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

You can cut and paste from here: <https://trinket.io/python/7a53c1ec41>

See next page for each task's description.

A video of describing the midterm is available here:

<http://www.eecs.yorku.ca/~mbrown/EECS1015midterm.mp4>

TASK 0 (0 points correct, -10 points deducted if you forget or have the wrong information)

The function task0 should print out the following information:

Midterm Exam - EECS1015
Name: Your Name
Student ID: Your York ID
email: youremail@aol.com

TASK 1 (15 points)

[Testing: Variables and expression, strings, input, and outputting formatted text]

Function task1 should do the following:

- (1) Prompt the user to input their first name.
- (2) Prompt the user to input their last name.
- (3) Prompt the user to ask them how many hours they work a week.
- (4) Prompt the user to ask them what is their hourly wage.
- (5) Based on the information obtained from (1)-(4), print out the following:

Employee: LASTNAME, Firstname
\$XXXX.XX Monthly salary (gross)
-\$YYYY.YY 25% Tax deduction
\$ZZZZ.ZZ Monthly salary (net)

You should trip off all white-spaces.
Last name should be printed with All capitals letters.
First name should be printed with first letter capital, the rest lower case.

Monthly salary (gross) is **hours per week * hourly wage * 4**
Tax deduction is monthly salary * 0.25
Monthly salary (net) is your (gross) salary minus the taxes. This is the amount you have after taxes.

All amounts should be printed with 2 decimal places.

Example:

Your first name: ABDEL
Your last name: Zhang
Hours you work per week: 30
Hourly wage: 18.88
Employee: ZHANG, Abdel
\$2265.60 Monthly salary (gross)
\$-566.40 25% Tax deduction
\$1699.20 Monthly salary (net)

Red text is entered by user. Notice the extra spaces for items 1 and 2.

NOTE: You may assume the that other than white-space, the user's input will be correct. For example, you do not need to check if the hours per week or hourly wage are numbers – you can assume they will be provided correctly.

See accompanying video: <http://www.eecs.yorku.ca/~mbrown/EECS1015midterm.mp4>

TASK 2 (30 points)

[Testing: Variables and expression, strings, conditional-statements, loops]

Function task2 allows users to order items from a menu as shown below.

The user starts out with a total \$10.00. Your function should begin by printing out the user's balance (amount of money they have left) and the 4 menu items below with the associated prices (you need to use the exact items and prices below). The user can then input the item they'd like to order. They can enter 0 to exit.

You have **\$10.00** - what item do you want?

1: Falafel \$3.00

2: Pizza \$6.00

3: Salad \$1.50

4: Coffee \$1.00

Enter 0 to exit.

Your order: **1**

Order for *falafel* confirmed.

Depending on what input the user provides you should do the following:

(1) Make sure they have enough money to purchase the item.

(2) If they do, print out "Order for *ITEM* confirmed." (replace ITEM with what they purchased).

(3) Deduct the item's price from the total.

You have **\$7.00** - what item do you want?

1: Falafel \$3.00

2: Pizza \$6.00

3: Salad \$1.50

4: Coffee \$1.00

Enter 0 to exit.

Your order: **2**

Order for *pizza* confirmed.

(1) User enters a 0 or has no more money.

You have **\$1.00** - what item do you want?

1: Falafel \$3.00

2: Pizza \$6.00

3: Salad \$1.50

4: Coffee \$1.00

Enter 0 to exit.

Your order: **4**

Order for *coffee* confirmed.

Thank you!

If the user enters 0 or has no more money, finish the task by print "Thank you!"

(1) WHAT TO DO IF THE USER DOES NOT HAVE ENOUGH MONEY? or (2) puts in a order that isn't "1" to "4"?

You have **\$1.00** - what item do you want?

1: Falafel \$3.00

2: Pizza \$6.00

3: Salad \$1.50

4: Coffee \$1.00

Enter 0 to exit.

Your order: **1**

Sorry, you don't have enough money for that item.

(1) If the user does not have enough money for an item, print the following "Sorry, you don't have enough money for that item".

(2) If the user inputs an order that is not "1" to "4", just repeat and ask what they would like to purchase.

See accompanying video: <http://www.eecs.yorku.ca/~mbrown/EECS1015midterm.mp4>

TASK 3 (25 points)

[Testing: Variables and expression, strings, conditional-statements, loops]

Function task3 should perform as follows:

Simulate rolling a dice 10 times. Each roll generates a random number between 1-6 and print it as shown below. If a [6] is rolled, print it out with ** around it (see below). If two or more of the rolls are a "6"s, print "YOU WIN", otherwise print "YOU LOSE".

Keep repeating the game until the user types in a "N". See below:

Rolling dice 10 times . . .

[3]

[1]

[1]

[4]

[3]

[1]

[5]

[2]

[2]

[3]

YOU LOSE!

Do you want to play again? (Y/N):

Rolling dice 10 times . . .

[2]

[2]

[6]

[1]

[6]

[5]

[6]

[1]

[5]

[3]

YOU WIN!

Do you want to play again? (Y/N): N

See accompanying video: <http://www.eecs.yorku.ca/~mbrown/EECS1015midterm.mp4>

Task 4 (30 points)

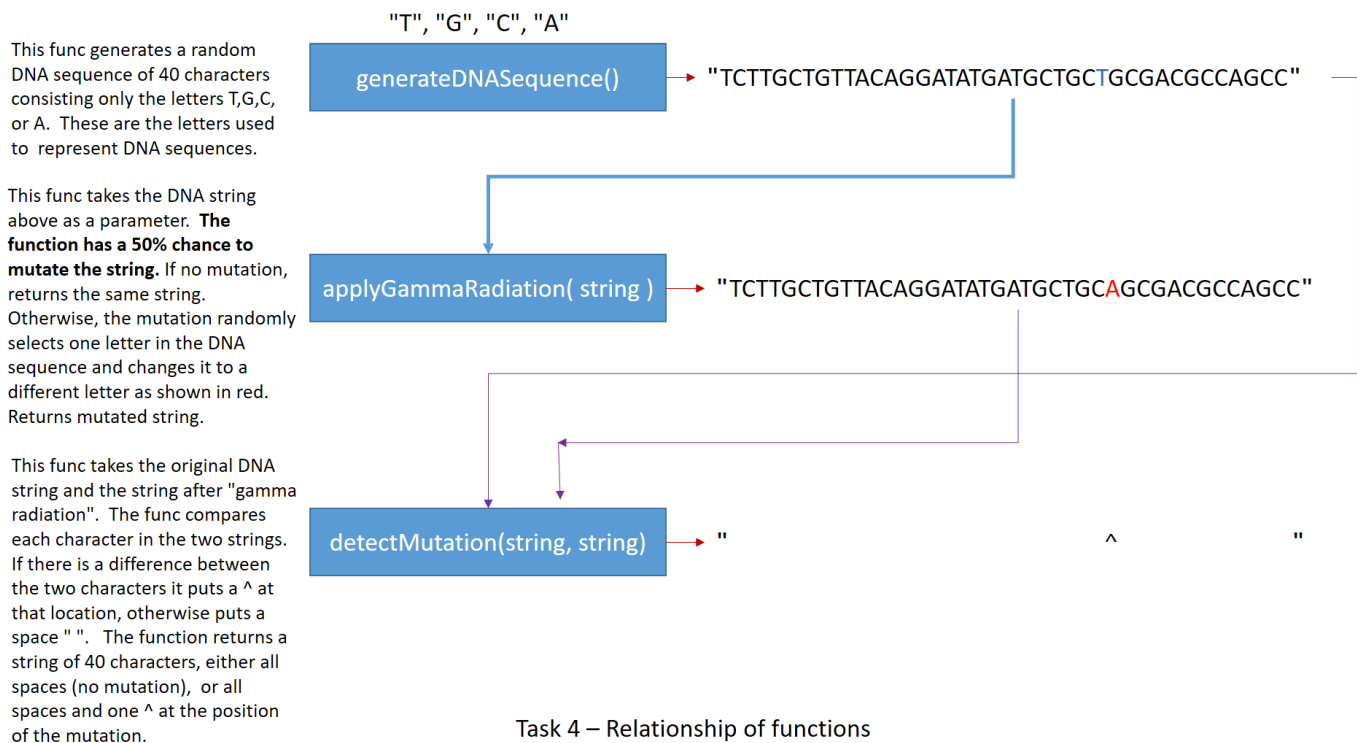
[Testing: Variables and expression, strings, conditional-statements, loops, functions]

Function task4 will require you to also define three additional functions as follows:

1. `generateDNASequence()` -> returns a string a string of 40 characters
2. `applyGammaRadiation(param: string)` -> returns a string of 40 characters
3. `detectMutation(param: string1, string2)` -> returns a string of 40 characters

See figure below that explains the relationship of the three functions.

Further details are provided after the figure.



Task 4 should be implemented as follows:

(1) Task4 should first call the function `generateDNASequence()`. The function `generateDNASequence()` will return a string of **40** characters. The string should be randomly generated character by character. Each character can be only one of the following characters: "T", "G", "C", or "A". These characters are the letters used to describe a DNA sequence. The function should return the randomly generated string.

See an example string: TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC

Task 4's description continues on the next page . . .

(2) Task4 should now call the function `applyGammaRadiation(param: string)`

The function `applyGammaRadiation()` should be passed the string created by the function `generatedDNASequence()` as an argument as shown in the figure. The function `applyGammaRadiation()` has a 50% chance to **mutate** one of the letters in the DNA sequence.

[Compute 50% chance?] To perform the 50% chance mutation, compute a random number between 1 and 100. If the number is greater than 50, mutate the DNA sequence. Otherwise, don't mutate it.

[How to mutate the sequence?] Randomly select one of the letters in the DNA string (recall the string is size 40 characters). For the letter at your selected position, you should then randomly change its value to one of the other allowed DNA values. For example, say you selected position 30 to mutate and it has the letter "A". Then you need to randomly select "T", "G", or "C". If at position 30 it instead had the letter "C", then you would need to randomly change it to "A", "G", or "T".

[What to return?] If you mutated the string, then you should return a new string where one of the DNA letters has been changed.

Here is an example:

```
TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC (parameter passed to function)
TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC (return string with a mutation)
```

If no mutation occurred, then return the same string passed. For example:

```
TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC (parameter passed to function)
TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC (return string with no mutation)
```

(3) Task4 should now call the function `detectMutation(string1, string2)`

The function `detectMutation(string1, string2)` is passed the original DNA string and the DNA string after gamma radiation has been applied. Detecting a mutation must be done by comparing each characters in the two DNA string sequences to see if they are the same or different. The function `detectMutation` should return a new string that is 40 characters long. If there is no mutation, this string will be all spaces, however, if there is a mutation then place a "^" in the string at that location the mutation occurred.

Here is an example (with a mutation):

```
Parameters:  "TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC"
              "TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC"
Returns:     "              ^              "
```

Here is an example (without mutation):

```
Parameters:  "TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC"
              "TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC"
Returns:     "              "
```

Task 4's description continues on the next page . . .

Now that you have seen the functions, task4 should print out as follows:

TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC (DNA)
TCTTGCTGTTACAGGATATGATGCTGCTGCGACGCCAGCC (DNA after radiation)
^
MUTATION DETECTED

Result from generateDNASequence
Result from applyGammaRadiation
Result from detectMutation
If there is a mutation detected, print
"MUTATION DETECTED"

Or, when no mutation is present, print out:

CTCTGAAAATCCTCCGTTTGTACGCGTGTCTATACATGTT (DNA)
CTCTGAAAATCCTCCGTTTGTACGCGTGTCTATACATGTT (DNA after radiation)
NO MUTATION DETECTED

Result from generateDNASequence
Result from applyGammaRadiation
Result from detectMutation
(all spaces - so it looks empty)
If there is no mutation detected,
print "NO MUTATION DETECTED"

See accompanying video: <http://www.eecs.yorku.ca/~mbrown/EECS1015midterm.mp4>

FINAL COMMENTS: For tasks 1-4 you may introduce additional functions, however, it is not required and will not affect your grade either way. For Task4, you must define the three functions as required in the task's description. All 5 tasks (Task0 – 4) 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>

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**.

Passport YORK

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

Username:

Password:

Login

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

STEP 2 – Enter your passport York username/password.

Academic Year: 2020-21 ▾

Term: F ▾

Course: 1015 ▾

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
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 "Lab4".

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

Failure to submit your assignment to the proper course

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.

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:

midterm.py submitted

You have submitted these files:

midterm.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>