# Homework Assignment 6: Using Debugger and Source Code Repository

|  |  |
| --- | --- |
| Assigned date | 2024-10-25 |
| Due date | 2024-11-08 |
| Estimate required time | 3 hours |

This is an individual assignment.

* You may consult with professor and TA about any aspect of the assignment.
* You may consult with other students only in a general way, e.g., about debugging or Python issues, or questions about wording on the assignment.
* You cannot actively work with someone unless the assignment specifically grants permission to work together with another student.

## Prerequisites:

Please install the following software:

* IDLE (this is the default Python integrated development environment) and you should have it on your computer already. If not, please download and install Python from <https://www.python.org/>
* Git client: https://git-scm.com/

## Purpose

This assignment gives you opportunities to practice using a Python debugger and a source code repository (GitHub). You are free to use the debugger of your choice provided it can achieve the tasks required by this assignment. If you would like to use IDLE’s debugger (the default IDLE/debugger that comes by Python), please consult the document “2024-3-CMPT\_140\_HW6 - IDLE.pptx” for a brief introduction. In this assignment, you will be asked to take screenshots of your debugger and describe information reported by your debugger.

## Specification

Please perform the tasks described below.

**Part 1: Debugger**

There is a bug in the file draw\_maze (buggy).py This code draws a maze. This code was writing by the CMPT140 instructor. It was working perfectly; however, he accidentally deleted a line of codes. Please help him fix this code by using a debugger.

1. Please download the following files from Moodle: draw\_maze (buggy).py and maze\_example\_very\_small.csv
2. Please update line 145 so that it points to the correct location of the input maze specification (CSV) file.
3. Execute the draw\_maze (buggy).py Python file. Open the CSV file in Excel and highlight the 0’s (walls). This will help you visualize what the maze *should* look like. The two screenshots shown below corresponds to the output of draw\_maze (buggy).py Python file (left) and the maze specification CSV file when opened in Excel (right).

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

As can be seen, draw\_maze (buggy).py is not drawing the maze correctly. Something is wrong.

1. To interrogate, please put a breakpoint at line 99. This is inside the main for-loop which iterates each row of the maze specification file and draw the corresponding wall blocks. This allows us to step through each line of codes and observe whether the values of the various variables are in fact what we expected them to be.

A screen shot of a computer code

Description automatically generated

1. After you’ve inserted the breakpoint, go ahead and execute the file with the Python debugger.
2. The program will stop at the breakpoint. **QUESTION (1)** what are the values of row, x and y shown by the debugger?

row = ['0', '0', '0', '0', '0', '0', '0', '0', '0\n']

X = 0

Y = 0

1. **Step into** (or **Step** in IDLE) the function draw\_maze\_row()
   * Note: you will have to click **Step into** a couple of times.
2. **QUESTION (2)**: once you are in the function draw\_maze\_row(), what are the values of x, y, block\_width and block\_height shown by the debugger? Please include a screen shot of your debugger window showing values of variables x, y, block\_width and block\_height.

A screen shot of a computer

Description automatically generated

1. Continue to **step over** (or **Over** in IDLE) the lines in the function draw\_maze\_row(). When you step through the iteration of the for loop (line 122-135), you will notice that the value of i (the loop index) changes. Please step through until you reach line 132 (i.e. when the function draw\_block() is called). **QUESTION (3)**: list out all the values of i that you observed.

i was equal to 1-8 before the draw\_block() function was called

1. **QUESTION (4)**: what is the value of num\_wall when turtle draws the bottom row (wall) i.e. when the function draw\_block() is called?

num\_wall = 9

1. After the turtle draws the bottom row (wall) and the function draw\_maze\_row() returns, the program execution will continue with the second iteration of the for-loop (line 97-101). The program will now read the second row and attempt to draw it. **Step into** (or **Step** in IDLE) the function draw\_maze\_row(). **QUESTION (5)**: what is the values of x and y shown by the debugger? Please include a screen shot of your debugger window showing values of variables x,y

A screen shot of a computer program

Description automatically generated

1. Continue to **step over** (or **Over** in IDLE) the lines in the function draw\_maze\_row() until the program draws the first block (wall) of the second row (from the bottom).
2. **QUESTION (6)**: what is the value of num\_wall when turtle draws the first block (wall) of the second row (from the bottom)?

Num\_wall = 1

1. Continue to **step over** (or **Over** in IDLE) the lines in the for loop (line 122-135) – you should now be in the second iteration of this for-loop i.e. i should now have the value of 2. If we look at the maze specification CSV file, you will notice that this row (the second row from the bottom) consist of one wall block followed by 7 road blocks (i.e. 1’s) and ends with a wall block (0). We expect the program to draw one block at x=0 and then the *second* one at x = 160 (i.e. 8 x 20; note: width of block = 20). **QUESTION (7)**: what is the value of x when the program draws the *second* block (i.e. immediately before the program draws the *second* block)?

X = 280

1. Why did the program draw the second block earlier than expected? **QUESTION (8)**: what conditions need to be true when the program draws a block or a row of blocks?

* (int(row[i]) *not* equal to \_\_maze\_wall\_\_ OR i == \_\_\_len(row)\_\_\_ ) AND
* num\_wall > \_\_0\_\_

1. **QUESTION (9)**: what are the values of i, row[i] and num\_wall immediately before when the program draws the *second* block?

i = 1

row[i] = ['0', '1', '1', '1', '1', '1', '1', '1', '0\n']

num\_wall = 1

1. num\_wall is a variable used to track the number of wall blocks that have NOT been drawn but the program has “read” in memory. This variable is incremented at line 126 (when int(row[i])==maze\_wall and i<(len(row)-1) i.e. when the program is in the middle of a row of wall blocks) and line 130 (when the program is NOT in the middle of a row of blocks but the current block is a wall block). However, the program is currently NOT in the middle of a row of wall blocks and the current block is NOT a wall block. There should have been no updates to num\_wall and its values should be 0. So, it seems there is some error in the logic of this programs.

Turns out, we forgot to reset the value of num\_wall to zero after the program draws the block(s). Recall num\_wall is used to keep track of the number of blocks NOT yet drawn.

Line 132 executes the codes to draw the block(s). Line 133 keeps track of the value of x (i.e. x-coordinates) after the blocks were drawn. We should add a line of code to reset the value of num\_wall to zero after line 133. **QUESTION (10)**: in the space provided below, please enter the codes (should be one line of codes) that should be added:

Num\_wall = 0

1. Quit the debugger. Re-run the modified Python file. The maze drawn should look exactly like the one shown below.

A grey square with a black arrow

Description automatically generated

**Part 2: Source Code Repository**

Turns out, if the instructor has been tracking his source codes in a source code repository, he will be able to compare his buggy codes with his latest working copy and would have noticed the line that was accidentally deleted. He will not need to go through all the debugging!

Note: you are free to use any Git client of your choice provided it can achieve the tasks described below.

Please refer the document “2024-3-CMPT\_140\_HW6 - Git client.pptx” available on Moodle.

1. Please clone the following repository: <https://github.com/TWU-2024-3-cmpt140/hw6.git>
2. A folder “hw6” will be created. This will be your working copy of the source code.
3. Edit the file “hw6/ draw\_maze (buggy).py” by inserting the line of codes that was inserted in step 17 above.
4. **QUESTION (11)**: please include a screenshot of your Git client showing the difference between your working copy and the copy on the source code repository.

A screenshot of a computer

Description automatically generated

## Deliverables

Please put your answers to the above **11 questions** and the requested screenshots in a Microsoft Word document and please name it as hw6-LastnameFirstname.docx (e.g. hw6-LeungSamuel.docx)and please submit it at <http://learn.twu.ca>

Note: you can use this word document (2024-3-CMPT\_140\_HW6.docx) as a starting point and put your answers/screen shots here.

Grading scheme: 1 mark for each question correctly answered. There are a total of 11 questions.

## Hints

Please reach out to TA or instructor for help if needed and please do so early.