ENGR 101 – Introduction to Programming
Mini Project 1

(Posted on October 30, 2019, Due on November 13, 2019)
Last Update: November 2, 18:50

In this mini project, you are going to develop a racing game called SEHIR Minesweeper by using the Swampy module. The primary goal of this project is to make you practice functions, loops, conditional execution, and getting user input through keyboard. The detailed explanations about the game that you are going to develop are provided below (Note: example console output from the game is written in BLUE):

A High Level Overview of the Game:

This game will be played by two players in a turn-based manner. Each player will have a turtle with a name and color. Turtles will be located initially on the first blocks of the board which is the leftmost side of the playground, and the turtle that arrives first to the rightmost side of the table will win the game. In each turn, the corresponding player will roll a dice to move his/her turtle N steps forward (N being the result of the dice, 1 - 6). In each roll, the game will randomly select a value from the dice (details are below). There are black turtles called bombs which will be placed randomly in a range of blocks on the playground. A player will be terminated if his/her turtle ends up at a block with a bomb on it. The first turtle player who reaches the last block on its track will win. The game will continue in a turn-based manner until one of the players wins.

Text-based User Interface:

The game will be played by two players in a turn-based manner. When the game first starts, the players will be asked to enter the names of their turtles. Names cannot be empty. If a player enters an empty name, then the game should warn the user and ask for a valid name again and again, until the player provides one. Below shows a sample running for this part of the game.

```
***** Welcome to Sehir Minesweeper *****

------- First Turtle -------
Please type the name of the first Turtle:  Harry
```

After the first user types the name of his/her turtle, the game will ask the user to select a color for the his/her turtle.

```
Please choose turtle color for Harry (red, blue, or green): black
```

If the user asks for a color that is not one of the three provided colors, the game should warn the user and ask for a valid color again and again, until the user provides a valid color.

ERROR! black is not a valid color, please select one of red, blue, or green colors: blue

After the first user types the name and the color of his turtle, the game will show a welcome label. Then, it will ask for the name of the second player's turtle.

------ Harry IS READY TO GO :) ------

------- Second Turtle -------
Please type the name of the second Turtle: Harry

Note that if the second player enters the same name as the first player, as in the above example, the game should warn the user, and then keep asking for a different name until the second user writes a different name. For instance, in the below example, if the second player typed Harry, the game would provide the following warning:

ERROR! Harry is already taken, please enter another name !

------- Second Turtle -------
Please type the name of the second Turtle: Ron

Again, the game must ask the user to select a color for his/her turtle.

Please choose turtle color for Ron (red, blue, or green): red

------ Ron IS READY TO GO :) ------

If the second user did not select a valid color, or chose the same color as player 1, the game should **keep** asking until a valid color is chosen as described above.

Once the user writes a valid name and color, the game will start. At the beginning of the game, the user will be asked to draw the game area.

Enter 1 to draw the game area and deploy bombs randomly: 1

After the user enters the appropriate number in the above, a TurtleWorld window should be opened, game area drawn, user and bomb turtles should be placed on the appropriate blocks as shown in below. There are a total of 30 blocks and 5 bombs.

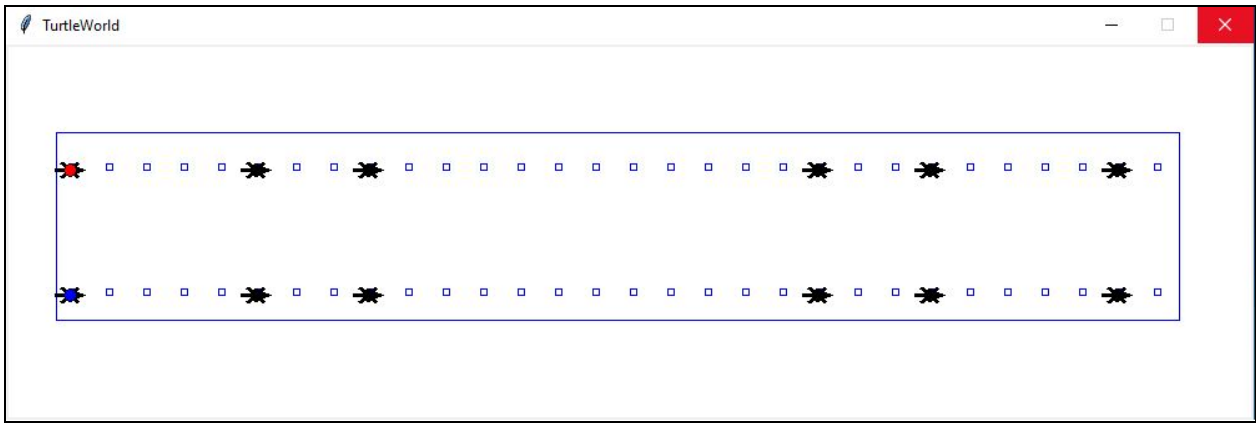

Figure 1: TurtleWorld view at the start of the game

As shown in above, blocks are represented by squares and a black turtle represents a bomb. The side length of each block is 5, the distance between the two blocks is 30, the size of the frame around the blocks is 900 x 150 and finally the size of the window is 1000 x 300. Bombs must be placed randomly between blocks 5 and 29. In each user's path, the bombs should be placed at the same positions. For this, you may use the random module (please see the implementation notes section below) to randomly choose the position of bombs. Two bombs cannot be located on the same block on a given track.

At the beginning, the game will decide who will start the game. Like placing the turtle bombs, you will use the random module to decide who will start first by making a coin toss. There will be 50% probability that any of the users start first. Player 1 was chosen to start the game first in below.

```
Let's start the game with a coin toss.
Harry won the toss, Harry starts first.
```

The toss winner will start the game by rolling a dice through pressing ENTER. A dice should be rolled and randomly return the result of the dice. In the meantime, the turtle of the user will move forward 30 * dice result (the distance between each block is 30), i.e. if the dice result is 3, the turtle will move 3 blocks forward. After each successful move, the result of dice and block number of each user should be displayed on the console as shown in the below.

```
Please press ENTER to roll the dice Harry !
Dice Result: 3
```

Harry's Score: 3
Ron's Score: 0

In the scenario that dice result causes the turtle to go off the playground, you should only move the turtle to final block, **turtles cannot go beyond the designated playground**. i.e. this happens when a player's turtle is 2 blocks away from the final block to win, but in that specific move, the dice result is 5, the turtle should only move 2 blocks. If the result of the dice is 6, the same user gets one more chance to roll the dice again as bonus. If a user's turtle is placed on the same block as a bomb after a move, that user will be eliminated from the game. After a user is eliminated, the turn should never go to the eliminated user and the score should not be updated. After a user is eliminated the following message should be displayed on the console.

Harry stepped on bomb. BOOOM !!!
Harry is eliminated

After this point on, the game will be played in a single turn manner until the surviving player either wins by reaching block 30 or steps on a bomb. If the surviving player steps on a bomb, the following message should be displayed on the console and the question of whether you want to play again should be asked.

Ron stepped on bomb. BOOOM !!!
Ron is eliminated

No one could win this game. Would you like to play again? (yes/no)

In the scenario that one of the turtles reaches the finish line (30th block on rightmost), the following message should be displayed.

Hooorrayy !! Ron has won.
Ron wins the game, would you like to play again? (yes/no)

If the user enters yes in the above question, the whole game should start from the very beginning and ask their names. The following screenshots visualizes different stages of the game on TurtleWorld.

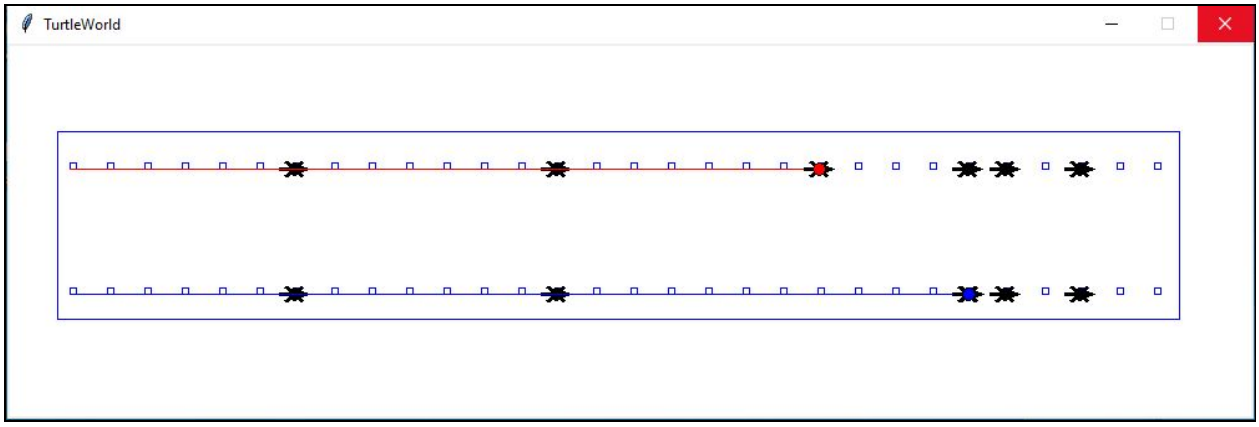

Figure 2: Blue turtle stepped on bomb and eliminated, red turtle continuing to play

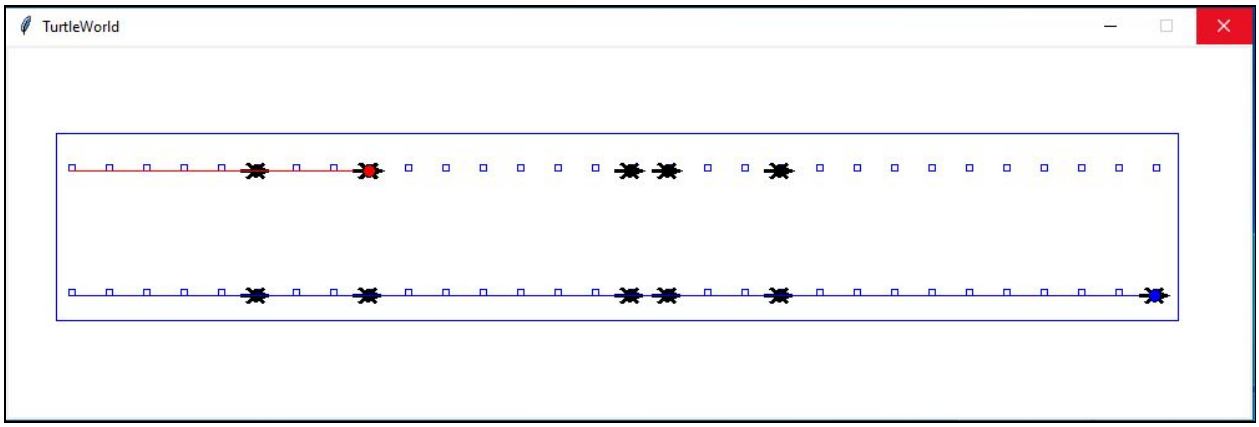

Figure 3: Blue turtle reached the finish line (30th block)

Implementation Notes:

- Use as many functions as you can. At least, you have to define and use 3 functions.
- You may check this link to get further information about Random module. (https://www.pythonforbeginners.com/random/how-to-use-the-random-module-in-pytho)
- You may use turtle_name.x and turtle_name.y to set the position of a turtle on the TurtleWorld. Negative coordinate values are allowed.
- 2 bombs cannot be placed on the same block. After randomly placing the bombs, there should be 5 bombs placed in different blocks in a single path. Each track is used by a single user only. The position of bombs on both paths should be the same.
- You may use TurtleWorld().geometry('width x height') to set the size of TurtleWorld window. '1000x300' is recommended. The data type for geometry method is string.
- You may use Turtle().die() after you are done using a turtle in order to remove it from playground.
- To change the color of a turtle, you may use turtle_name.set_color('blue') function.

- To move the turtle without drawing anything, you may use pu() function call (to hold the pen up), and then to start drawing again, you may put the pen down with pd() function call.
- You may set the delay parameter (i.e. turtle_name.delay = 0.01) to a small number to move your turtles faster.

Warnings:

- **Do not** talk to your classmates on project topics when you are implementing your projects (This is serious).
- **Do not** show or email your code to others (This is even more serious).
- If you need help, talk to your TAs or the instructor, not to your classmates. If somebody asks you for help, explain them the lecture slides, but do not explain any project related topic or solution. Any similarity in your source codes will have **serious** consequences for both parties.
- Carefully read the project document, and pay special attention to sentences that involve "**should**", "**should not**", "**do not**", and other underlined/bold font statements.
- If you use code from a resource (web site, book, etc.), make sure that you reference those resources at the top of your source code file in the form of comments. You should give details of which part of your code is from what resource. Failing to do so **may result in** plagiarism investigation. Last but not least, you need to understand code pieces that you may get from other resources. This is one of the goals of the mini projects.
- Even if you work as a group of two students, each member of the team should know every line of the code well. Hence, it is **important** to understand all the details in your submitted code.

How and when do I submit my project?

- Projects may be done individually or as a small group of two students (doing it individually is **strongly** recommended for the best learning experience). If you are doing it as a group, only **one** of the members should submit the project. File name will tell us group members (Please see the next item for file naming details).
- Submit your own code in a **single** Python file. Name it with your and your partner's first and last names (see below for naming):
    - If your team members are Deniz Barış and Ahmet Erdem Çalışkan, then name your code file as deniz_baris_ahmet_erdem_caliskan.py (Do **not** use any Turkish characters in the file name).
    - If you are doing the project alone, then name it with your name and last name similar to the above naming scheme.

- Those who **do not** follow the above naming conventions **will get 5 pts off** of their grade

- Submit it online on LMS by **17:00 on November 13, 2019.**

Late Submission Policy:

- -10%: Submissions between 17:01 – 18:00 on the due date.
- -20%: Submissions between 18:01 – midnight (00:00) on the due date.
- -30%: Submissions which are up-to 24 hours late.
- -50%: Submissions which are up-to 48 hours late.
- Submission more than 48 hours late will not be accepted.

Grading Criteria?

Code Organization			**Functionality**				
Not using meaningful variable names (-10% cut off from your overall grade)	Not properly using functions, code repetitions (-10% cut off from your overall grade)	Insufficient commenting (-10% cut off from your overall grade)	Coin Toss (10 pts)	Properly draw the game table and deploy 5 turtle bombs (40 pts)	Properly handling the dice rolling and making moves accordingly (20 pts)	Making it continue to another round (20 pts)	Others (10 pts)

Have further questions?

If you need help with anything, please use the office hours of your TAs and the instructor to get help. **Do not walk in randomly (especially on the last day) into your TAs' or the instructor's offices. Observe office hour times or make an appointment first outside of office hours. This is important. Your TAs have other responsibilities. Please respect their personal schedules.**