Tic-Tac-Toe

Hasan Can Aslan

Submission Date: 3rd of March 11:59 PM

1 Introduction

On this project, your challenge in this exercise is to implement the classical Tic-Tac-Toe game logic using Python, and a graphical visualization using the `turtle` module to make a two-player game that you can play with a friend.

1.1 Submission

We will use *OK*

1.2 Academic Honesty

Koç University's *Statement on Academic Honesty* holds for all the homework given in this course. Failing to comply with the statement will be penalized accordingly. If you are unsure whether your action violates the code of conduct, please consult with your instructor.

1.3 Aim of the Project

In your homework assignments, the **functionality** and **style** of your programs are **both important.** A program that "works" is not necessarily a good program. A good program is **comprehensible, readable and well structured.** Therefore you're expected to decompose the main problem into simpler subtasks and **implement helper functions** for these subtasks. You're also expected to write comments if needed, be careful about indentation and use descriptive names for helper functions. Even if your program functions well according to the project requirements, you can still improve your code style and how you decompose the task.

1.4 Given Code

This part is **optional** but advised as it will allow you to understand the given partitions of the code better. The code given to you has commented above all the functions. Below are the functions are given to you in the code with their explanation.

1.4.1 Given Functions

- `def draw_empty_board():`

  This function should draw the **empty Tic-Tac-Toe board** using turtle module.

- `def draw_x_in_square(row, column):`

  This function should draw a **X symbol** on the given square of tic-tac-toe board.

- `def draw_o_in_square(row, column):`

  This function should draw an **O symbol** on the given square of tic-tac-toe board.

- `def display_setup():`

  Sets the **display** i.e. screen size, animation speed.

- `if __name__ == '__main__':`

  '_main' is the name of the scope in which top-level code executes. You do not have to fully understand the benefits of writing your code under if __name__ == '__main__' block for now. We will discuss its meaning when we are talking about Python packages and modules. For now, just think of this as the main entry to your program.

1.4.2 Given Constants

Five constants: SCREEN_WIDTH, SCREEN_HEIGHT, PEN_SIZE, SQUARE_SIZE and ANIMATION_SPEED are given at the top of the project. You will use these constants as **arguments** for configuring the screen and turtle. Since you did not learn functions in detail yet, **we will provide necessary code in corresponding sections of the project.**

1.5 Further Questions

For further questions **about the project** you may contact **Hasan Can Aslan** at [haslan16@ku.edu.tr]. Note that it may take up to 24 hours before you receive a response so please ask your questions **before** it is too late. No questions will be answered when there is **less than two days** left for the submission.

2 Project Steps

2.1 Displaying Empty Board

Let's start to draw! Our first step is drawing an empty Tic-Tac-Toe board using the turtle module. To do that you should fill the given function:

```python
def draw_empty_board():
```

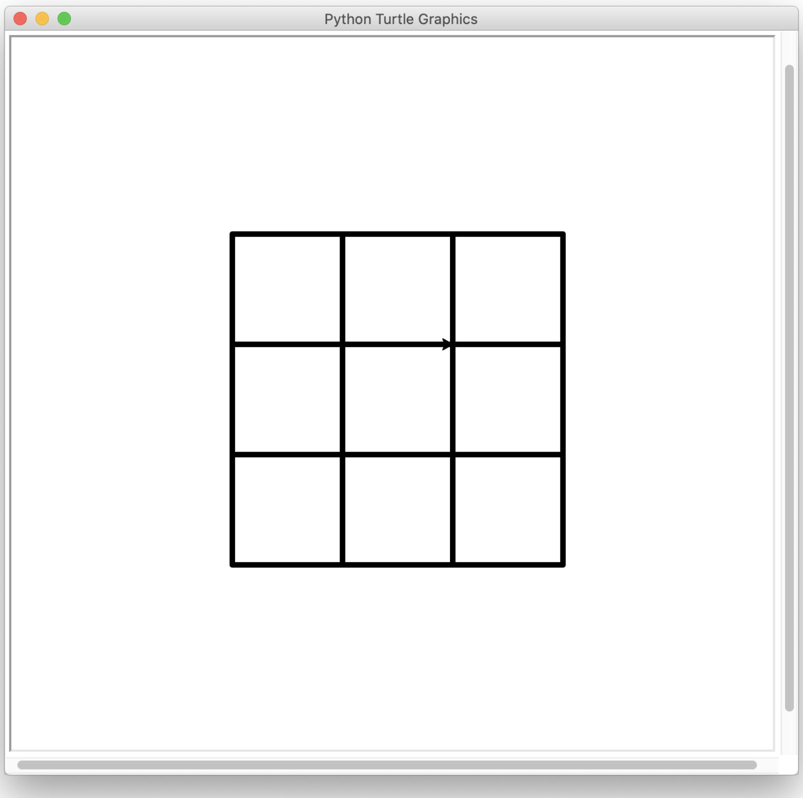


Figure 1: Empty board.

Hints:

1. You might want to implement a helper function for drawing only one square.
2. You can create a "nested" for loop to draw a 3x3 board using the helper function you created.

2.2 Drawing X

Now you are warmed up. Drawing two more lines should be a piece of cake for you. This function should draw an X symbol, here is the trickier part, you should draw X on the given square of the Tic-Tac-Toe board. Takes two arguments, row and column numbers for X to be drawn. You need to fill the given function:

```python
def draw_x_in_square(row, column):
```

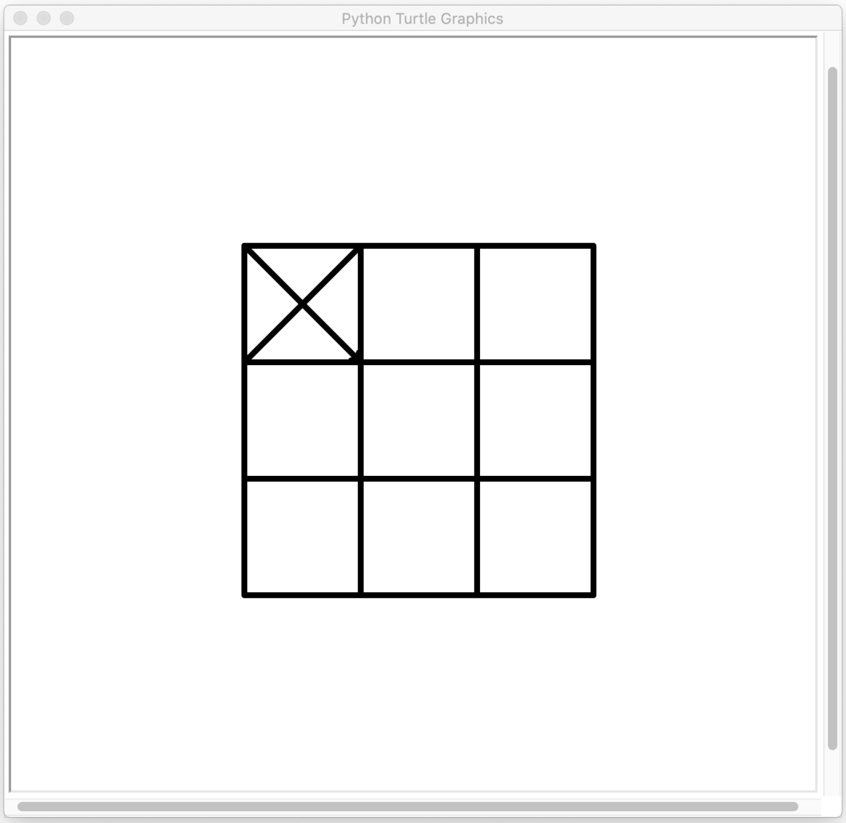


Figure 2: X drawing on board.

Hints:

1. You might want to use penup, pendown, setpos, setheading, forward functions from the turtle module.
2. We recommend you to spend some time with turtle module in the interactive shell to understand how it uses coordinates (which directions are positive, which angles correspond to which directions, etc.)

2.3 Drawing O

Very well, Let's bend these lines! This function is very alike to the previous one only difference is its shape. You should draw an O symbol on the given square of the Tic-Tac-Toe board. Takes two arguments, row and column numbers for O to be drawn. You need to fill the given function:

```python
def draw_o_in_square(row, column):
```

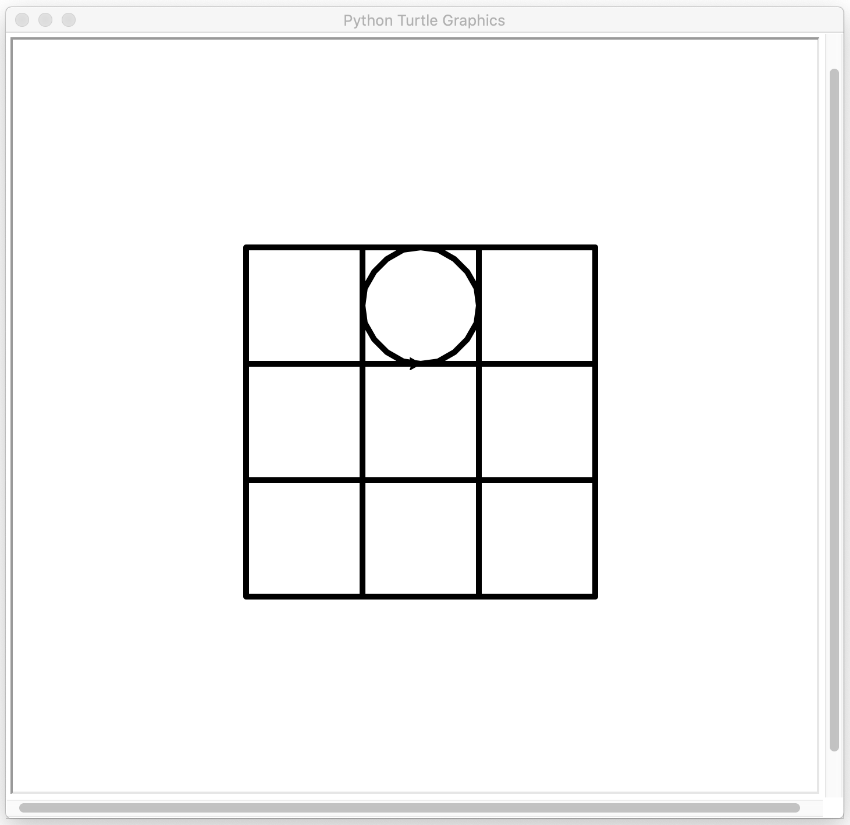


Figure 3: O drawing on board.

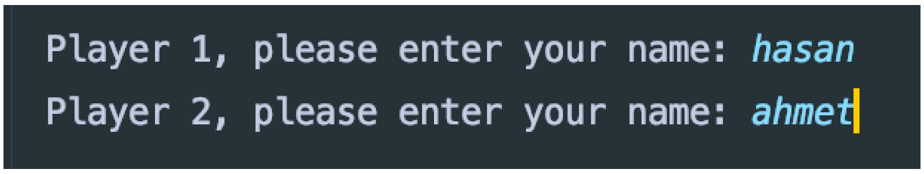
Hints:

1. You might want to use penup, pendown, setpos, setheading, circle functions from the turtle module.
2. We recommend you to spend some time with turtle module in the interactive shell to understand how it uses coordinates (which directions are positive, which angles correspond to which directions, etc.)

2.4 Taking Player's Names

You've done great so far! Now, you should take players' names, assign them to *player_names* list. Any string other than an empty string is considered a valid name.

```
player_names = []
```



```
Player 1, please enter your name: hasan
Player 2, please enter your name: ahmet
```

Figure 4: Enter name prompt.

2.5 Getting Current Player

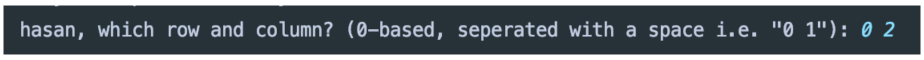
You should get current player's name to play, assign it to *current_player_name* variable.

```
current_player_name = ''
```

2.6 Taking Player's Move

Let's begin to play! The next step is to deal with handling user input. When a player (say player 1, who is X) wants to place an X on the screen, they can't just click on a screen. So we are going to approximate this clicking simply by asking the user for a coordinate of where they want to place their piece. Get where the user wants to place X or O. Feel free to choose which user to put which sign. Assign it to *move* variable for checking.

```
move = ''
```

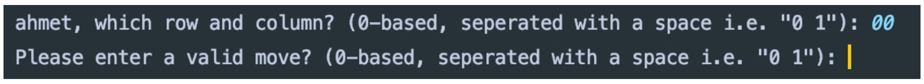


```
hasan, which row and column? (0-based, seperated with a space i.e. "0 1"): 0 2
```

Figure 5: Taking player's move.

2.7 Validating the Move

You must validate the user's move. Don't worry about checking whether someone won the game at this part, but if a player tries to put a piece in a game position where there already is another piece, do not allow the piece to go there. If the move is invalid, you need to ask again until the user enters a valid move.



```
ahmet, which row and column? (0-based, seperated with a space i.e. "0 1"): 00
Please enter a valid move? (0-based, seperated with a space i.e. "0 1"):
```

Figure 6: Invalid input.

Hints:

1. It should be a string consisting of two parts (Hint: use "string".split(' ') function)
2. Both its parts should be integers (Hint: "string".isnumeric() function)
3. Integers should be in range [0, 2] inclusive.
4. Selected square should be empty.

2.8 Playing the Move

It's time to play the move! There are two things to do: modifying the *game* list and display the move using the turtle. You should use the functions for drawing X and O that you created earlier in this project.

Hint:

- Game of Tic-Tac-Toe is represented as a list of lists, like so:
  game = [[X, O, X], [O, X, O], [O, X, X]]

2.9 Checking Win Conditions

Now, after the user plays his/her move you need to check there is a win condition holds. Your challenge in this part is checking conditions and assign result (True or False) to *there_is_winner* variable. If the user wins, terminate the loop and if there is a winner, and show a congratulatory message on the console. If there is no winner at the end of the game (*there_is_winner* variable is still False), given code prints "Tie".

```
there_is_winner = False
```

Hint:

- A Tic-Tac-Toe win is 3 in a row - either in a row, a column, or a diagonal.

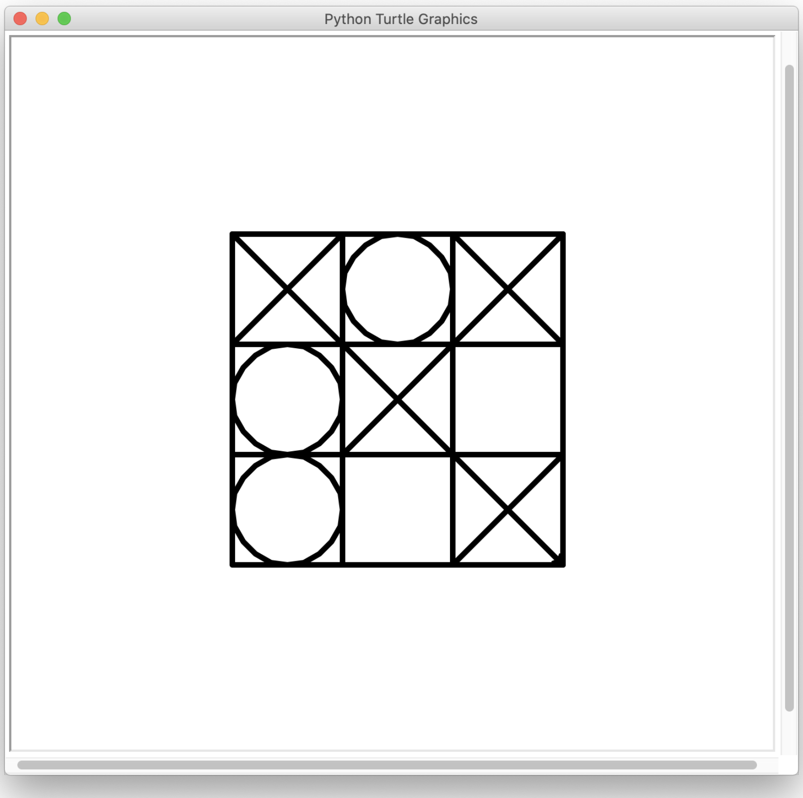
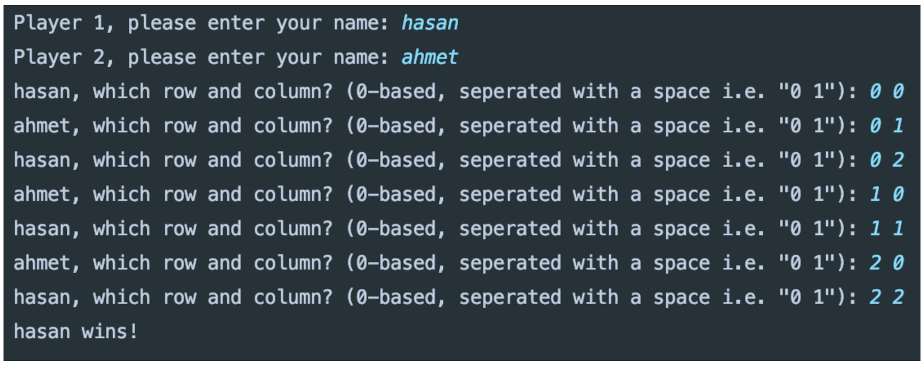


Figure 7: Example win game image.



```
Player 1, please enter your name: hasan
Player 2, please enter your name: ahmet
hasan, which row and column? (0-based, seperated with a space i.e. "0 1"): 0 0
ahmet, which row and column? (0-based, seperated with a space i.e. "0 1"): 0 1
hasan, which row and column? (0-based, seperated with a space i.e. "0 1"): 0 2
ahmet, which row and column? (0-based, seperated with a space i.e. "0 1"): 1 0
hasan, which row and column? (0-based, seperated with a space i.e. "0 1"): 1 1
ahmet, which row and column? (0-based, seperated with a space i.e. "0 1"): 2 0
hasan, which row and column? (0-based, seperated with a space i.e. "0 1"): 2 2
hasan wins!
```

Figure 8: Example win game console.

2.10 Congratulations!

You reach the end of the project! If you are new to Python, this might be one of the longest programs you have written so far, so you should proud of yourself!

2.11 End of Project

Your project ends here. You may continue to tinker with the code to implement any desired features and discuss them with your section leader. However, **do not** include any additional features that you implement after this point into your submission.

3 Resources

You can find all course and section related material in our website. Also, the turtle reference below could be useful when implementing the assignment. You can find the detailed reference for turtle in here.

3.1 Turtle Reference

- `turtle.penup()`

  Pull the pen up – no drawing when moving.

- `turtle.pendown()`

  Pull the pen down – drawing when moving.

- `turtle.forward(distance)`

  distance – a number (integer or float)
  Move the turtle forward by the specified *distance*, in the direction the turtle is headed.

- `turtle.left(angle)`

  angle – a number (integer or float)
  Turn turtle left by *angle* units. Angle orientation depends on the turtle mode.

- `turtle.right(angle)`

  angle – a number (integer or float)
  Turn turtle right by *angle* units. Angle orientation depends on the turtle mode.

- `turtle.circle(radius)`

  radius – a number (integer or float)
  Draw a circle with given *radius*.

- `turtle.setpos(x, y)`

  x – a number (integer or float)
  y – a number (integer or float)
  Move turtle to an absolute position. If the pen is down, draw line. Do not change the turtle's orientation.

- `turtle.setheading(to_angle)`

  to_angle – a number (integer or float)
  Set the orientation of the turtle to *to_angle*. Here are some common directions in degrees