**Outline | Program Structure:**

Please be advised of the following for the program BattleShips\_v1.py including:-

* Structure of the program
* How the location of the battleship is generated
* Program flow

1. Initializing the board

I visualized the board as a grid with x and y coordinates of a “picture” marked with text characters either “X” or “O”. Staring from the (0, 0) origin in the upper-left corner, with the x-coordinates increase going right, and the y-coordinates increase going down.

The next task was to setup the game-board by creating and declaring a variable called “board” and then setting it equal to an empty list i.e. board = []

* 1. Then using Python’s built in functions to generate the board, i.e. a 5 x 5 grid of all “O”s (<-not zero’s). The idea was to print [“O”] \* 5 per line to represent each row of the board; to be repeated 5 times to make five rows by using range () function to loop 5 times. Inside each of the loops I used .append () function to append a list containing 5 zero’s to the board variable:-

for x in range (5):

board.append ([“O”]\*5)

1. Start the game & print the board

At the start of the game and on each guess I opted to print the board, so the player can see which locations they have already guessed. I considered the fact that the information in the board is structured as a list within a list. First I set up a for loop to go through each of the elements in the outer list (each row of the board) and print them.

* 1. Define a function names “printBoard” with a single argument “board”.
  2. Inside the function, write a for loop to iterate through each “row” in “board” and print it to the screen.
  3. Also inside the function and inside the for loop I used “ “ as the separator to .join the elements of each row:-

def printBoard (board):

for row in board:

print(" ".join(row))

* 1. Then call the function “printBoard” after each guess:-

print ('Strike ' + str (strike+1) + ' out of 10.')

printBoard (board)

1. Computer “hides” the battleship using random co-ordinates – within the 5x5 grid
   1. Firstly I import the randint(low,high) function from the random module (before any code):-

from random import randint

* 1. Now I needed to generate the x (ship\_row) and y (ship\_ col) co-ordinates in the grid from 0 to 5, by defining two functions random\_row and random\_col that each take board as an argument. These functions should return a random row and random column index from the board. as follows:-

def random\_row(board):

return randint(0, len(board) - 1)

def random\_col(board):

return randint(0, len(board[0]) - 1)

Although, I could just called randint(0,4). I used len(board)-1 in case I wanted to change the board size later.

1. Get input co-ordinates from user

The code block here allows the user (or player) to guess the row and column co-ordinates where the battleship is.

* 1. I create new variables guess\_row and guess\_col respectively, and ask the user for input using input function. Then convert the input from a string to an integer values:

guess\_row = int(input('Guess Row(0-4):'))

guess\_col = int(input('Guess Col(0-4):'))

1. If player co-ordinates matches right the game ends

Okay. From Steps (1-4) above we now have the location of the ship, and the player’s guesses’ - so we can check to see if the player guessed right. I.e. the co-ordinates were correct.

* 1. For a guess to be right, guess\_col = ship\_col, and guess\_row = ship\_row.
  2. Added a break statement under the win condition to end the loop after a win :-

if guess\_row == ship\_row and guess\_col == ship\_col:

print ('BRAVO! HIT !! You sunk my battleship! You GOT ME !!')

print ('My Ship Row = '+str(ship\_row))

print ('My Ship Col = '+str(ship\_col))

break # game ends here!

1. If the guess is wrong, mark the point with an X and start again.

Of course, the player is not going to guess right all the time, so I needed to handle this where the guess is wrong, by adding an else under the if block in step 5.1, with an output to the screen “You missed my battleship!”.

* 1. Then set the list element at guess\_row, guess\_col = “X”. Also as last line in else statement, call printBoard(board) again so can see the “X”’s in the board:

else:

print ('You missed my battleship!')

board[guess\_row][guess\_col] = "X"

# Print strike and board again here

print ('Strike ' + str(strike+1) + ' out of 10.')

printBoard(board)

1. Check/validate the ship can be placed at given co-ordinates

What if the player is a “bad-shot” and guesses a spot off the board completely? Then I have added additional tests within my else condition for the miss condition as follows:-

* 1. I’ve added a new if statement that is nested under the else, that checks the players given inputs do not lie outside the range 0 and 4 for both the columns and rows, as follows:-

if (guess\_row < 0 or guess\_row > 4) or (guess\_col < 0 or guess\_col > 4):

print ("Ouch , that's not even in the ocean !")

1. Output warning if the guess matches previous guess given

Now here I handle the 2nd type of incorrect guess where the player guess’s a location that was already selected.

* 1. Here I add an elf to see if the guessed location already has an ‘X’ in it. If it has, output warning message to the screen, to confirm guess has already been made:-

elif(board[guess\_row][guess\_col] == "X"):

print ('You guessed that one already.')

1. Print strike and updated board:-

# Print strike and board again here

print ('Strike ' + str(strike+1) + ' out of 10.')

printBoard(board)

1. If player has had 10 turns or strikes, …then GAME OVER !

Here I used a for loop to iterate through each turn (or Strike).

* 1. Added a for loop that repeats the guessing and checking part of the game for 10 turns.
  2. At the beginning of each iteration, print “Strike”, strike+1 to let the player know what turn they are on:-

for strike in range(10):

>>> Steps 3 to 8 as per above <<<

print ('Strike ' + str(strike+1) + ' out of 10.')

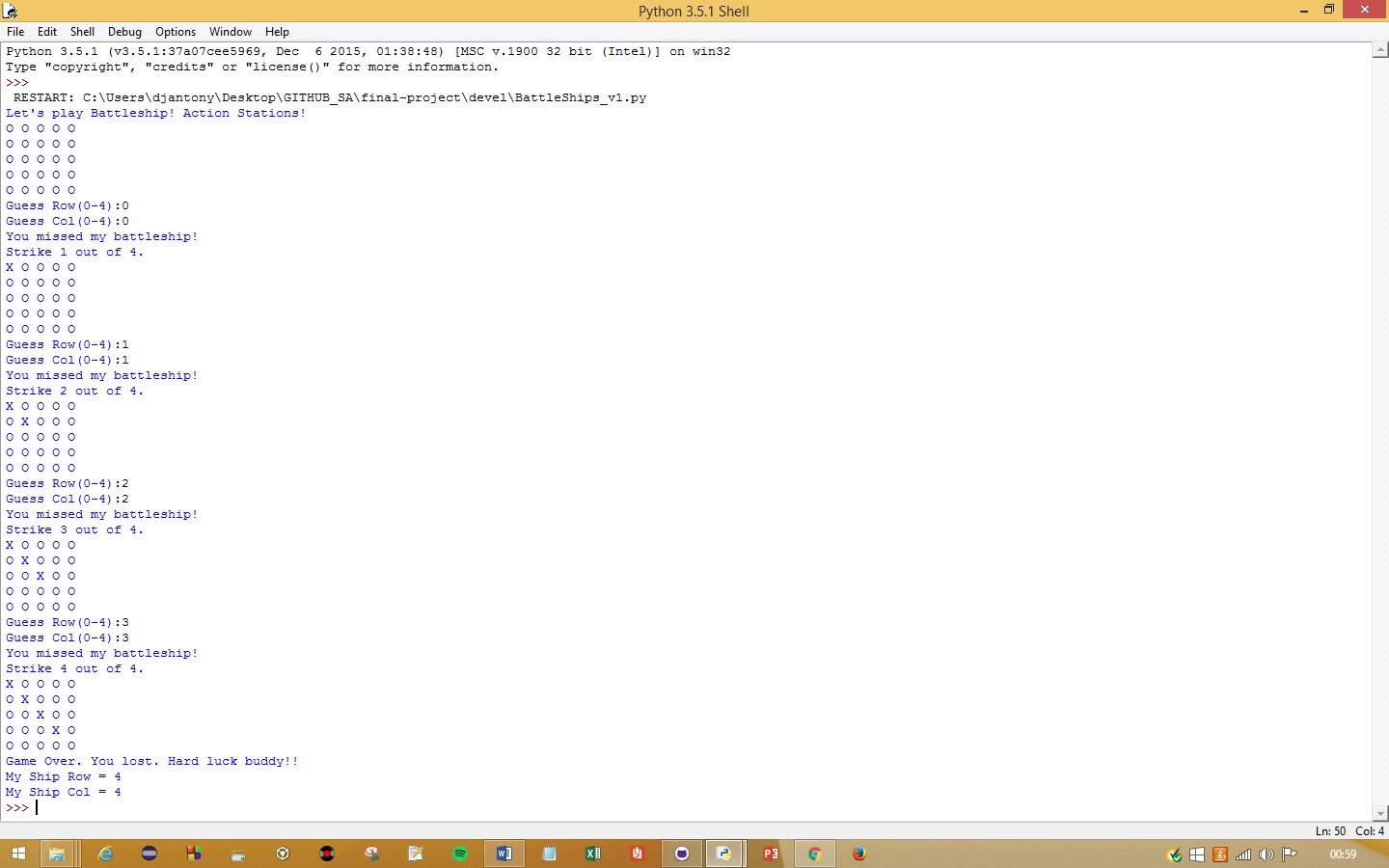
printBoard(board)

if strike >= 9:

print ('Game Over. You lost. Hard luck buddy!!')

print ('My Ship Row = '+str(ship\_row))

print ('My Ship Col = '+str(ship\_col))

Sample Game Play:

/End