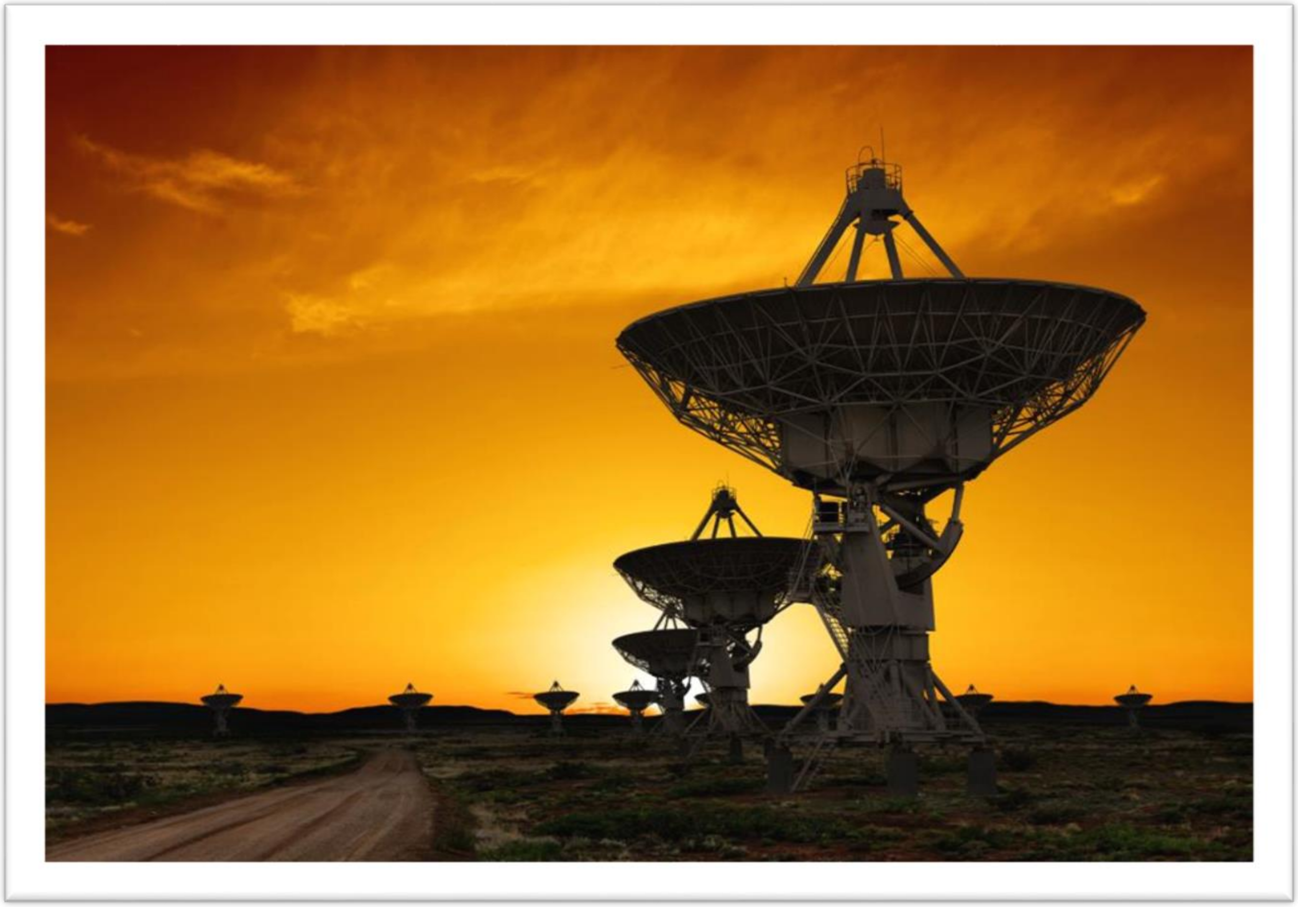


ASSIGNMENT - 1

SAI AKA(1910110333) & KONDURU SANDILYA(1910110205)

CSD 311 AI

Dr. SAROJ KAUSHIK

27-01-2021

PROBLEM -1

Magic squares predate recorded history. An ancient Chinese legend tells of a turtle emerging from the Lo river during a flood. The turtle's shell showed a very unusual pattern – a three-by-three grid containing various numbers of spots.

References to Lo Shu and the Lo Shu numerical pattern occur throughout Chinese history. Today, it is the mathematical basis for Feng Shui, the philosophy of balance and harmony in our surroundings and lives. An n-by-n magic square is an array containing the integers from 1 to n 2 , arranged so that each of the rows, each of the columns, and the two principal diagonals have the same sum. For each n > 2, there are many different magic squares of order n

ALGORITHM USED FOR GENERATING A MAGIC SQUARE USING FORMULA IS AS FOLLOWS:

Step 1: Start in the middle of the top row, and let n=1

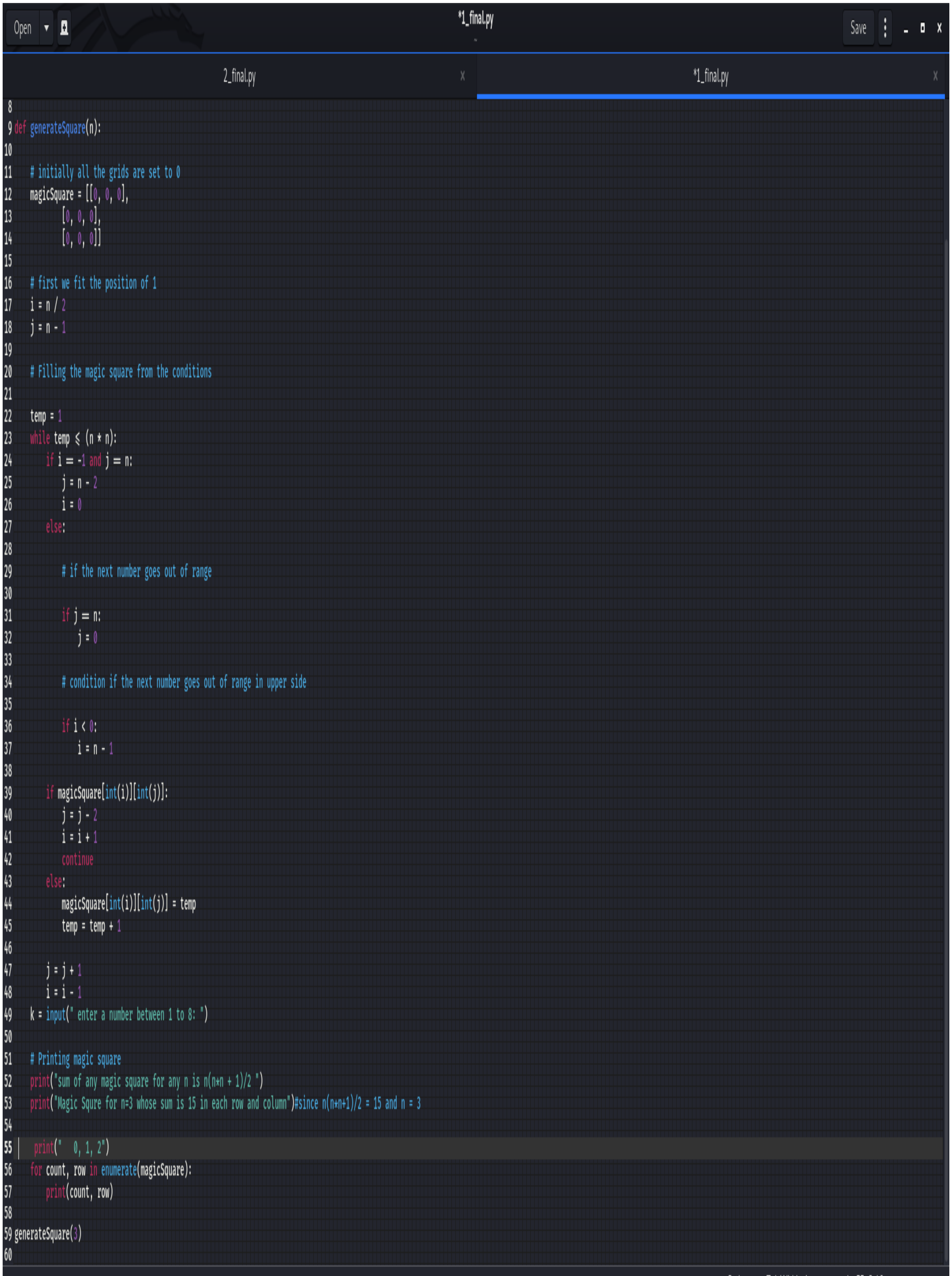
Step 2: Insert n into the current grid position;

Step 3: If n=N2 the grid is complete so stop. Otherwise increment n

Step 4: Move diagonally up and right, wrapping to the first column or last row if the move leads outside the grid. If this cell is already filled, move vertically down one space instead.

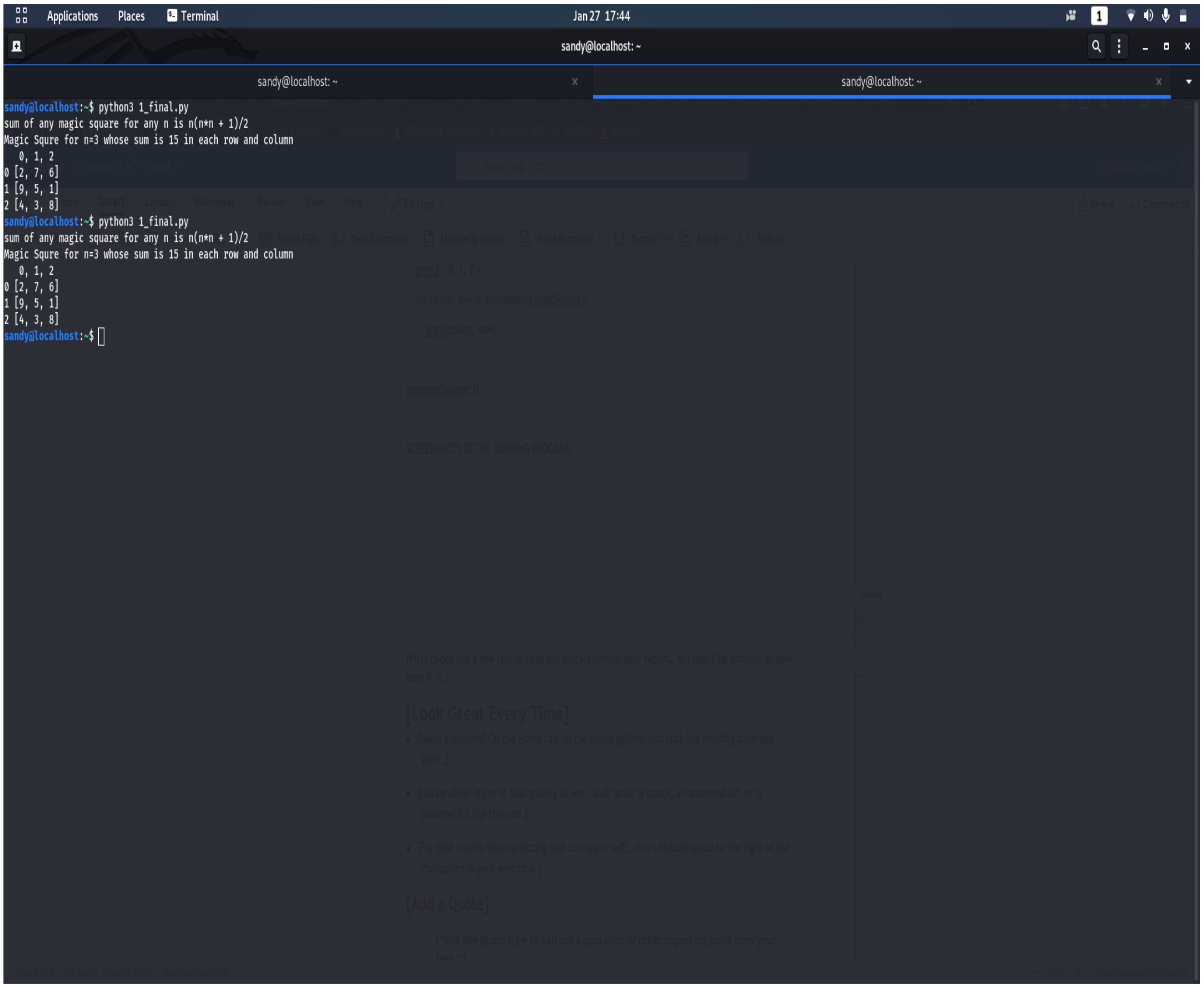
Step 5: Return to step 2.

CODE SNIPPET FOR THE PROBLEM:



```python

def generateSquare(n):

    # initially all the grids are set to 0
    magicSquare = [[0, 0, 0],
            [0, 0, 0],
            [0, 0, 0]]

    # first we fit the position of 1
    i = n / 2
    j = n - 1

    # Filling the magic square from the conditions

    temp = 1
    while temp <= (n * n):
        if i == -1 and j == n:
            j = n - 2
            i = 0
        else:

            # if the next number goes out of range

            if j == n:
                j = 0

            # condition if the next number goes out of range in upper side

            if i < 0:
                i = n - 1

        if magicSquare[int(i)][int(j)]:
            j = j - 2
            i = i + 1
            continue
        else:
            magicSquare[int(i)][int(j)] = temp
            temp = temp + 1

        j = j + 1
        i = i - 1
    k = input(" enter a number between 1 to 8: ")

    # Printing magic square
    print("sum of any magic square for any n is n(n*n + 1)/2 ")
    print("Magic Squre for n=3 whose sum is 15 in each row and column")#since n(n*n+1)/2 = 15 and n = 3

     print("   0, 1, 2")
    for count, row in enumerate(magicSquare):
        print(count, row)

generateSquare(3)

```

SCREENSHOTS OF THE RUNNING PROGRAM:



```
sandy@localhost:~$ python3 1_final.py
sum of any magic square for any n is n(n*n + 1)/2
Magic Squre for n=3 whose sum is 15 in each row and column
   0, 1, 2
0 [2, 7, 6]
1 [9, 5, 1]
2 [4, 3, 8]
sandy@localhost:~$ python3 1_final.py
sum of any magic square for any n is n(n*n + 1)/2
Magic Squre for n=3 whose sum is 15 in each row and column
   0, 1, 2
0 [2, 7, 6]
1 [9, 5, 1]
2 [4, 3, 8]
sandy@localhost:~$ 
```

PROBLEM2:

Tic-tac-toe (American English), noughts and crosses (Commonwealth English), or Xs and Os/“X’y O’sies” (Ireland), is a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a diagonal, horizontal, or vertical row is the winner. It is a solved game with a forced draw assuming best play from both players.

RULES FOR TIC-TAC-TOE

1. The game is played on a grid that's 3 squares by 3 squares.

2. You are X, your friend (or the computer in this case) is O. Players take turns putting their marks in empty squares.

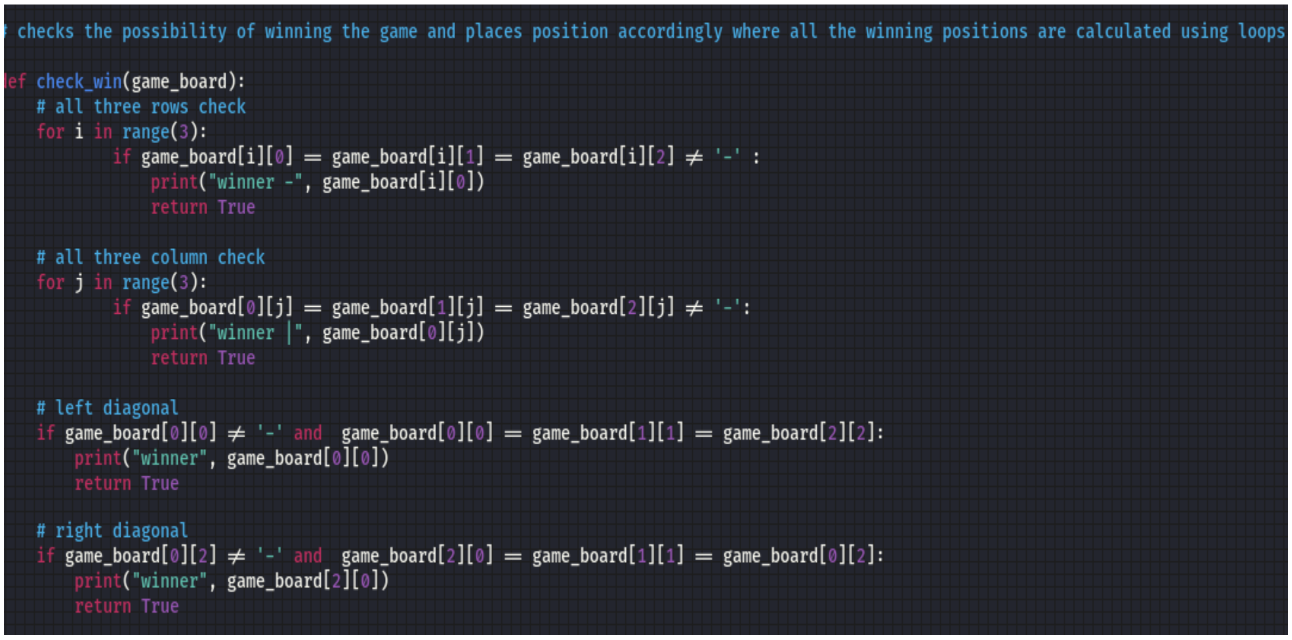
3. The first player to get 3 of her marks in a row (up, down, across, or diagonally) is the winner.

4. When all 9 squares are full, the game is over. If no player has 3 marks in a row, the game ends in a tie.

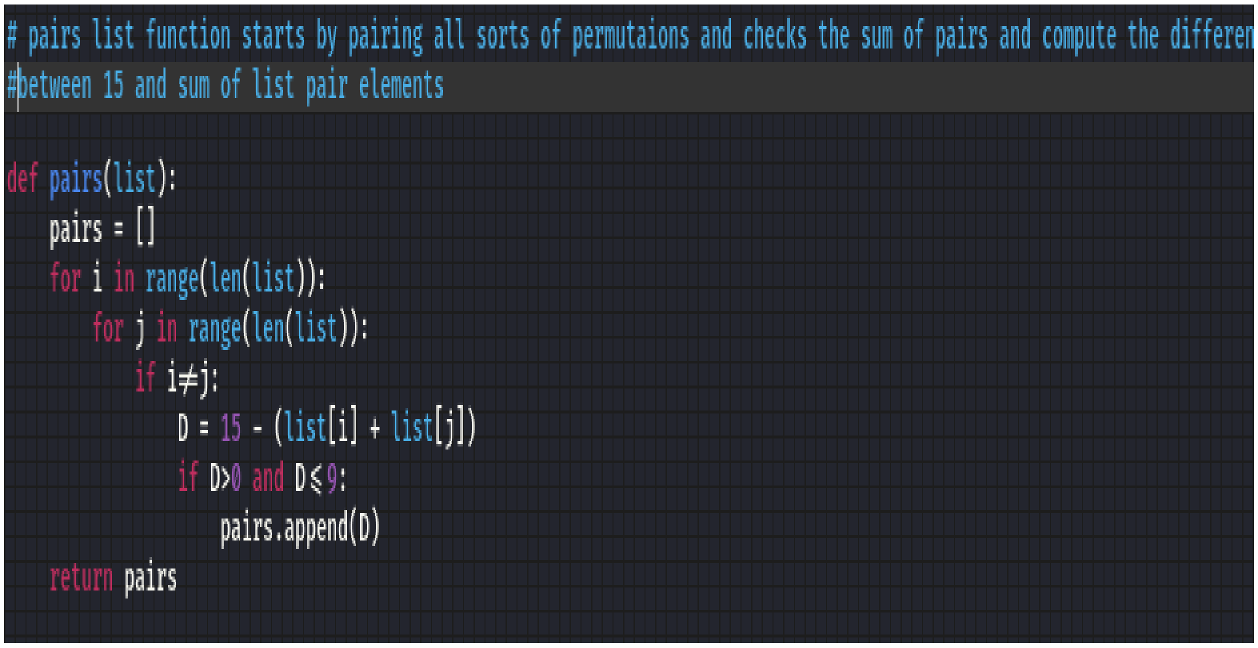
NOTE: One who plays first uses “X” as the marker.

CODE SNIPETS OF THE PROGRAM:

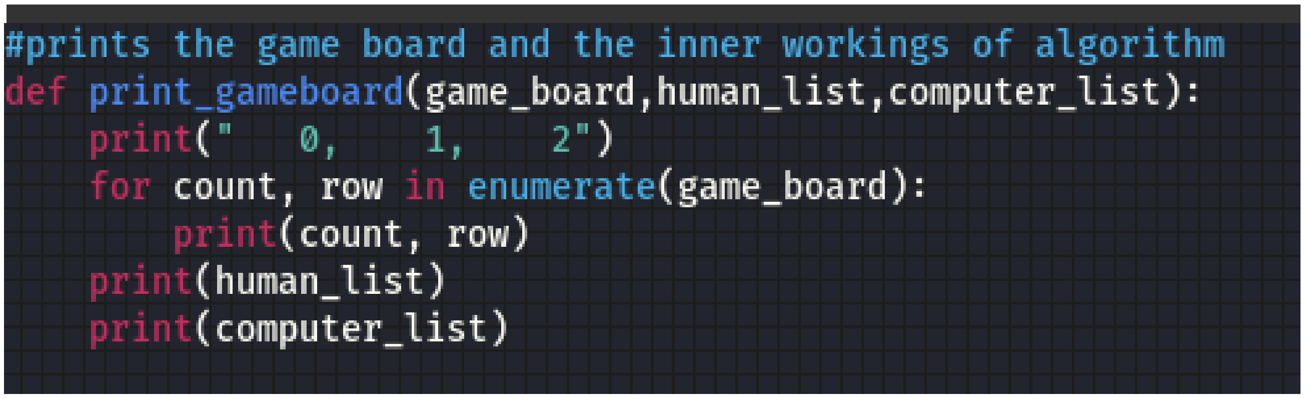
The following are the code snippets of the functions used:



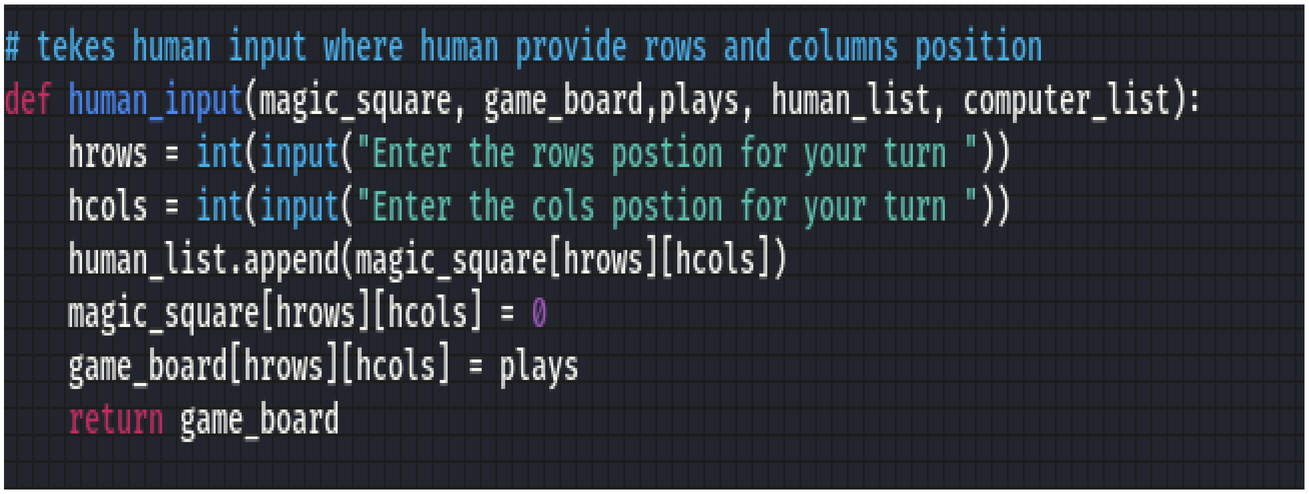
```
 checks the possibility of winning the game and places position accordingly where all the winning positions are calculated using loops

ef check_win(game_board):
    # all three rows check
    for i in range(3):
            if game_board[i][0] == game_board[i][1] == game_board[i][2] ≠ '-' :
                print("winner -", game_board[i][0])
                return True

    # all three column check
    for j in range(3):
            if game_board[0][j] == game_board[1][j] == game_board[2][j] ≠ '-':
                print("winner |", game_board[0][j])
                return True

    # left diagonal
    if game_board[0][0] ≠ '-' and  game_board[0][0] == game_board[1][1] == game_board[2][2]:
        print("winner", game_board[0][0])
        return True

    # right diagonal
    if game_board[0][2] ≠ '-' and  game_board[2][0] == game_board[1][1] == game_board[0][2]:
        print("winner", game_board[2][0])
        return True
```



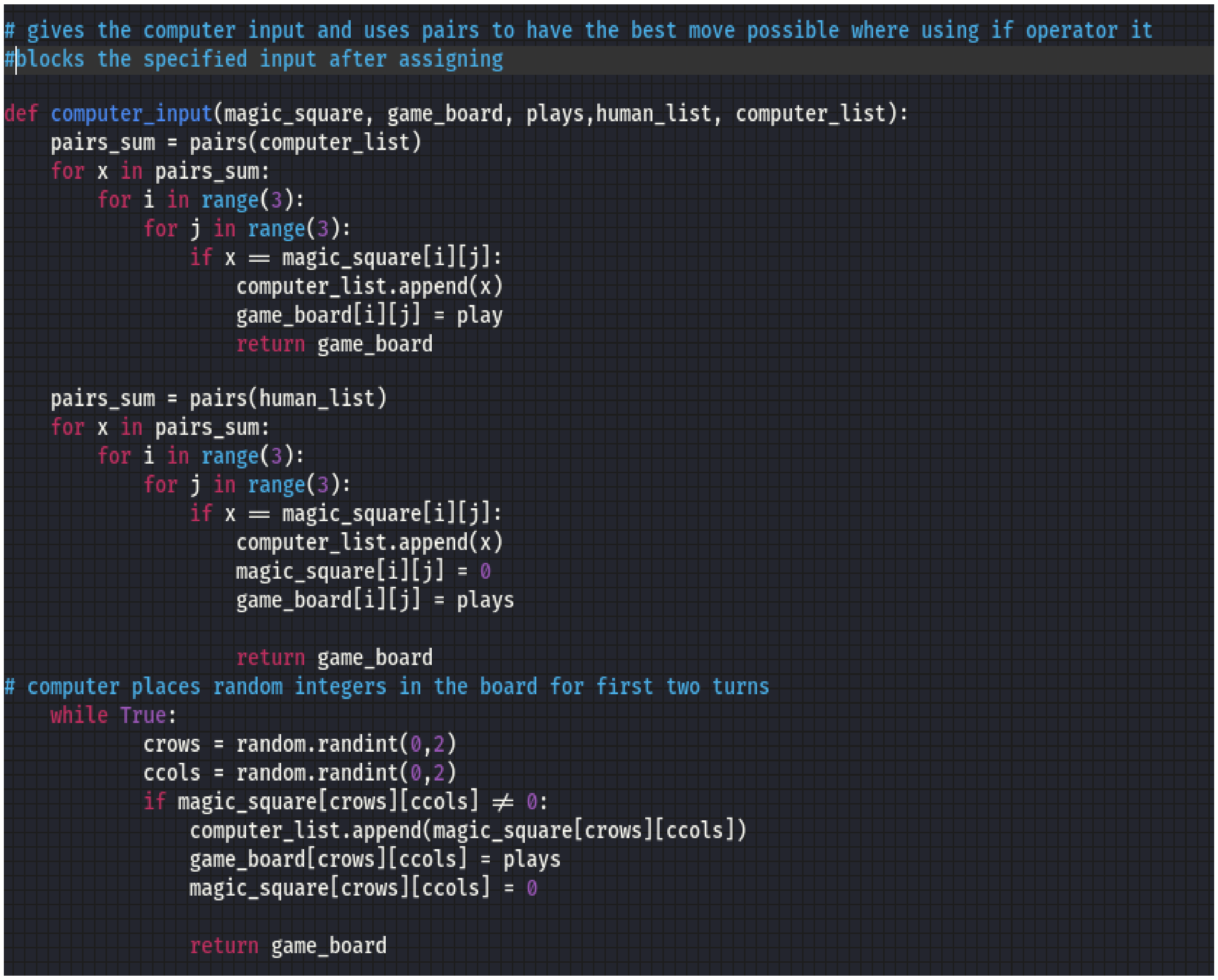
```
# pairs list function starts by pairing all sorts of permutaions and checks the sum of pairs and compute the differen
#between 15 and sum of list pair elements

def pairs(list):
    pairs = []
    for i in range(len(list)):
        for j in range(len(list)):
            if i≠j:
                D = 15 - (list[i] + list[j])
                if D>0 and D⩽9:
                    pairs.append(D)
    return pairs
```



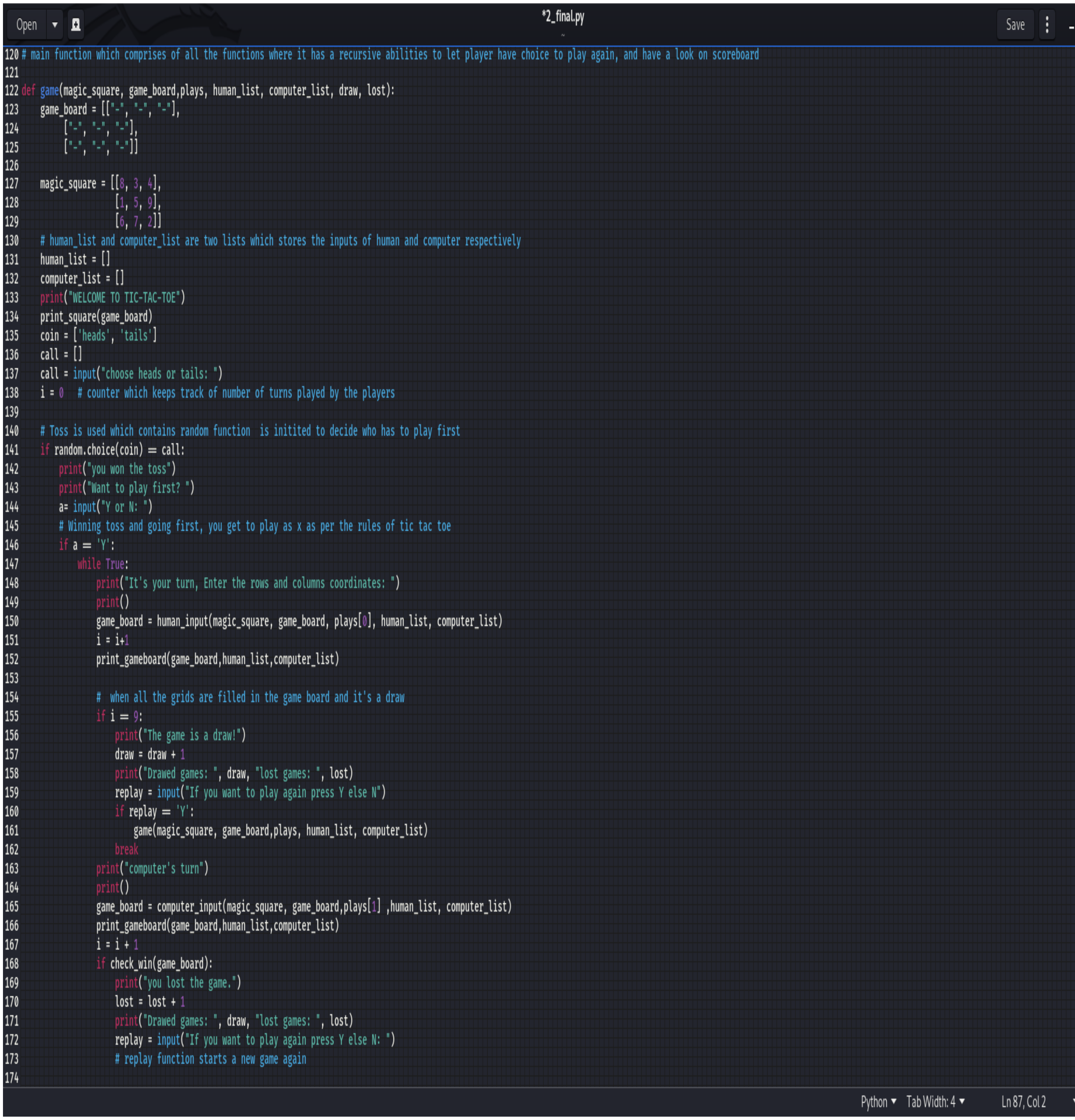
```
#prints the game board and the inner workings of algorithm
def print_gameboard(game_board,human_list,computer_list):
    print("   0,    1,    2")
    for count, row in enumerate(game_board):
        print(count, row)
    print(human_list)
    print(computer_list)
```



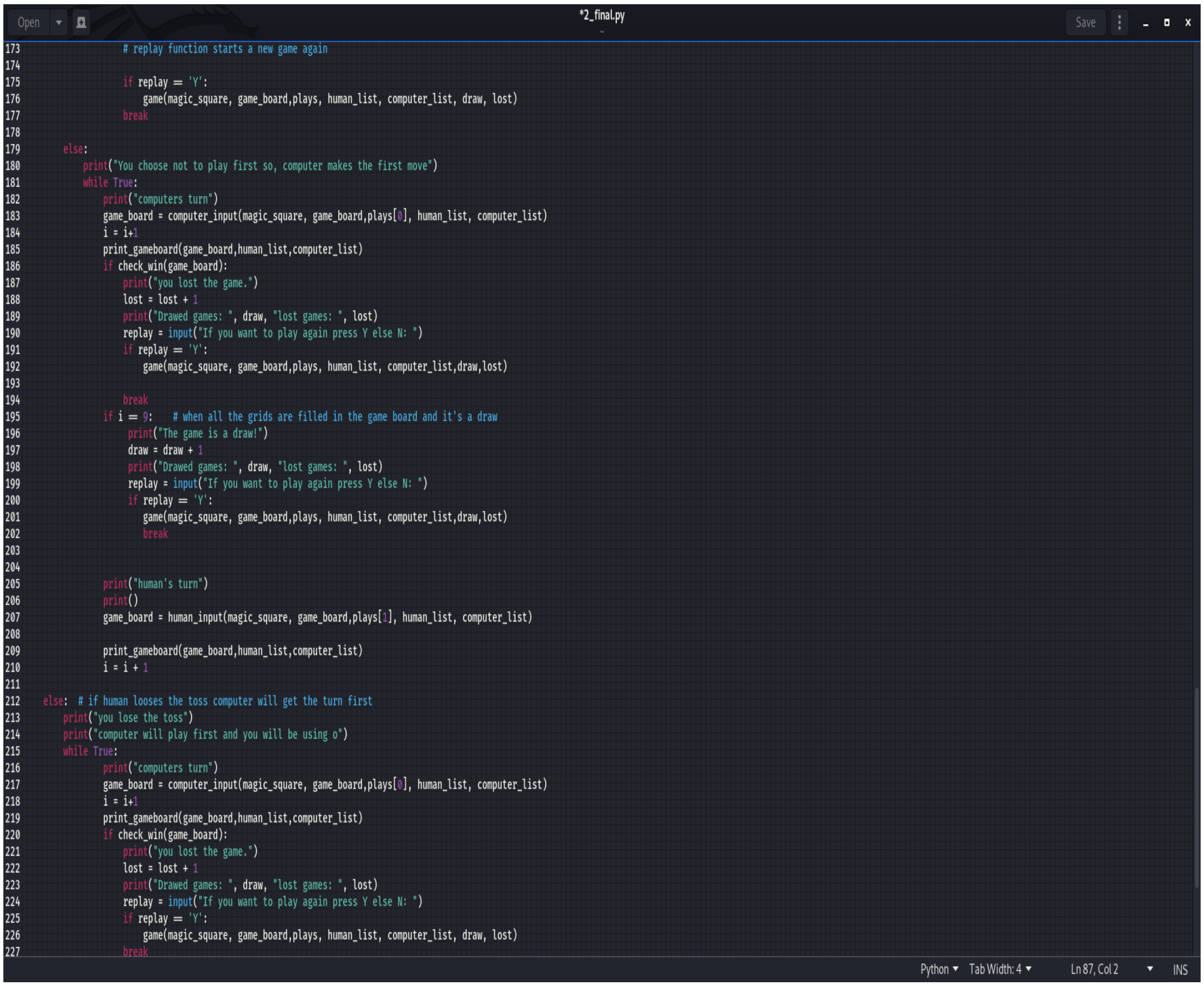
```
# tekes human input where human provide rows and columns position
def human_input(magic_square, game_board,plays, human_list, computer_list):
    hrows = int(input("Enter the rows postion for your turn "))
    hcols = int(input("Enter the cols postion for your turn "))
    human_list.append(magic_square[hrows][hcols])
    magic_square[hrows][hcols] = 0
    game_board[hrows][hcols] = plays
    return game_board
```



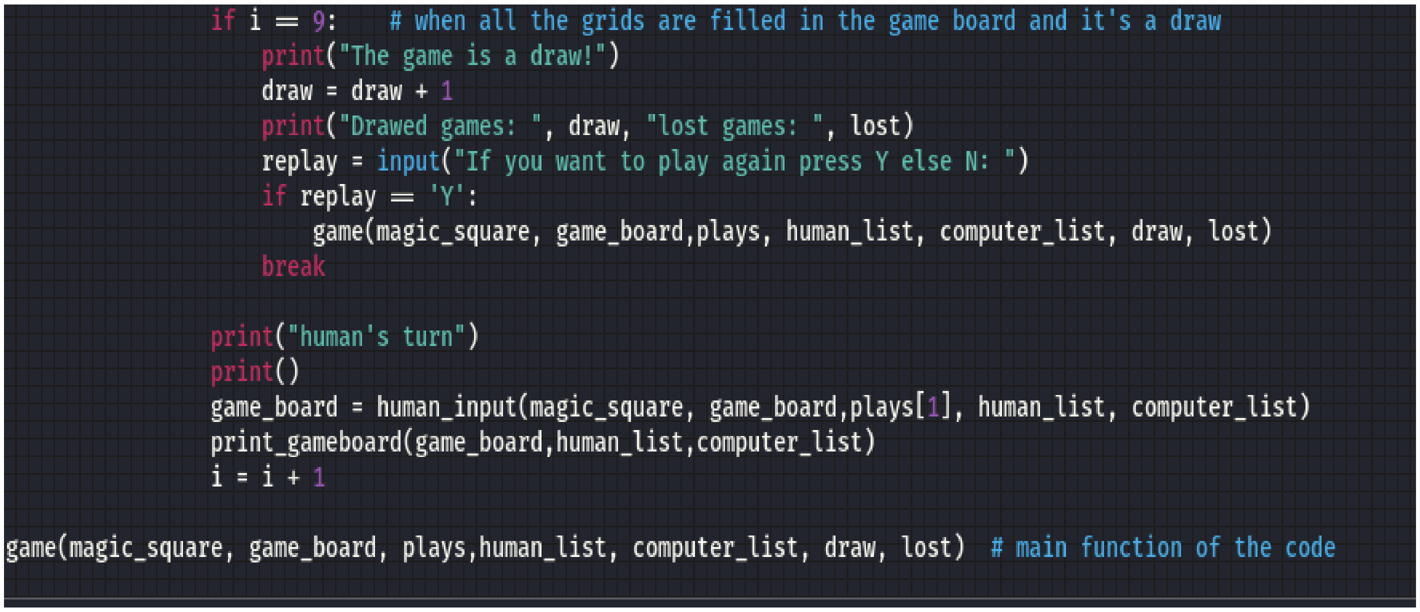
```python
# gives the computer input and uses pairs to have the best move possible where using if operator it
#blocks the specified input after assigning

def computer_input(magic_square, game_board, plays,human_list, computer_list):
    pairs_sum = pairs(computer_list)
    for x in pairs_sum:
        for i in range(3):
            for j in range(3):
                if x == magic_square[i][j]:
                    computer_list.append(x)
                    game_board[i][j] = play
                    return game_board

    pairs_sum = pairs(human_list)
    for x in pairs_sum:
        for i in range(3):
            for j in range(3):
                if x == magic_square[i][j]:
                    computer_list.append(x)
                    magic_square[i][j] = 0
                    game_board[i][j] = plays

                    return game_board
# computer places random integers in the board for first two turns
    while True:
            crows = random.randint(0,2)
            ccols = random.randint(0,2)
            if magic_square[crows][ccols] != 0:
                computer_list.append(magic_square[crows][ccols])
                game_board[crows][ccols] = plays
                magic_square[crows][ccols] = 0

                return game_board
```



```python
# main function which comprises of all the functions where it has a recursive abilities to let player have choice to play again, and have a look on scoreboard

def game(magic_square, game_board,plays, human_list, computer_list, draw, lost):
    game_board = [["-", "-", "-"],
         ["-", "-", "-"],
         ["-", "-", "-"]]

    magic_square = [[8, 3, 4],
                    [1, 5, 9],
                    [6, 7, 2]]
    # human_list and computer_list are two lists which stores the inputs of human and computer respectively
    human_list = []
    computer_list = []
    print("WELCOME TO TIC-TAC-TOE")
    print_square(game_board)
    coin = ['heads', 'tails']
    call = []
    call = input("choose heads or tails: ")
    i = 0   # counter which keeps track of number of turns played by the players

    # Toss is used which contains random function  is initited to decide who has to play first
    if random.choice(coin) == call:
        print("you won the toss")
        print("Want to play first? ")
        a= input("Y or N: ")
        # Winning toss and going first, you get to play as x as per the rules of tic tac toe
        if a == 'Y':
            while True:
                print("It's your turn, Enter the rows and columns coordinates: ")
                print()
                game_board = human_input(magic_square, game_board, plays[0], human_list, computer_list)
                i = i+1
                print_gameboard(game_board,human_list,computer_list)

                #  when all the grids are filled in the game board and it's a draw
                if i == 9:
                    print("The game is a draw!")
                    draw = draw + 1
                    print("Drawed games: ", draw, "lost games: ", lost)
                    replay = input("If you want to play again press Y else N")
                    if replay == 'Y':
                        game(magic_square, game_board,plays, human_list, computer_list)
                    break
                print("computer's turn")
                print()
                game_board = computer_input(magic_square, game_board,plays[1] ,human_list, computer_list)
                print_gameboard(game_board,human_list,computer_list)
                i = i + 1
                if check_win(game_board):
                    print("you lost the game.")
                    lost = lost + 1
                    print("Drawed games: ", draw, "lost games: ", lost)
                    replay = input("If you want to play again press Y else N: ")
                    # replay function starts a new game again

```



```python
                    # replay function starts a new game again

                    if replay == 'Y':
                        game(magic_square, game_board,plays, human_list, computer_list, draw, lost)
                    break

        else:
            print("You choose not to play first so, computer makes the first move")
            while True:
                print("computers turn")
                game_board = computer_input(magic_square, game_board,plays[0], human_list, computer_list)
                i = i+1
                print_gameboard(game_board,human_list,computer_list)
                if check_win(game_board):
                    print("you lost the game.")
                    lost = lost + 1
                    print("Drawed games: ", draw, "lost games: ", lost)
                    replay = input("If you want to play again press Y else N: ")
                    if replay == 'Y':
                        game(magic_square, game_board,plays, human_list, computer_list,draw,lost)

                    break
                if i == 9:    # when all the grids are filled in the game board and it's a draw
                     print("The game is a draw!")
                     draw = draw + 1
                     print("Drawed games: ", draw, "lost games: ", lost)
                     replay = input("If you want to play again press Y else N: ")
                     if replay == 'Y':
                        game(magic_square, game_board,plays, human_list, computer_list,draw,lost)
                        break


                print("human's turn")
                print()
                game_board = human_input(magic_square, game_board,plays[1], human_list, computer_list)

                print_gameboard(game_board,human_list,computer_list)
                i = i + 1

    else:  # if human looses the toss computer will get the turn first
        print("you lose the toss")
        print("computer will play first and you will be using o")
        while True:
                print("computers turn")
                game_board = computer_input(magic_square, game_board,plays[0], human_list, computer_list)
                i = i+1
                print_gameboard(game_board,human_list,computer_list)
                if check_win(game_board):
                    print("you lost the game.")
                    lost = lost + 1
                    print("Drawed games: ", draw, "lost games: ", lost)
                    replay = input("If you want to play again press Y else N: ")
                    if replay == 'Y':
                        game(magic_square, game_board,plays, human_list, computer_list, draw, lost)
                    break
```

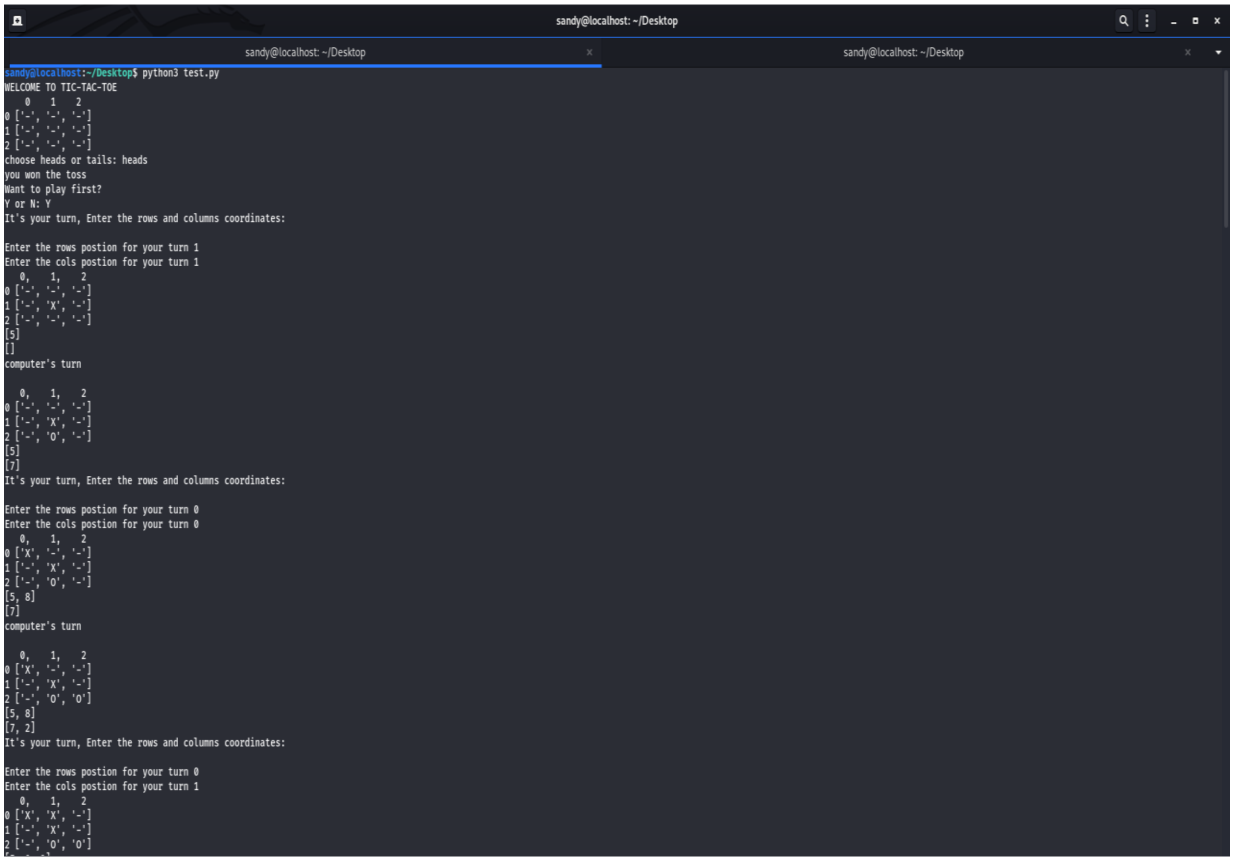


```
                if i == 9:    # when all the grids are filled in the game board and it's a draw
                    print("The game is a draw!")
                    draw = draw + 1
                    print("Drawed games: ", draw, "lost games: ", lost)
                    replay = input("If you want to play again press Y else N: ")
                    if replay == 'Y':
                        game(magic_square, game_board,plays, human_list, computer_list, draw, lost)
                    break

                print("human's turn")
                print()
                game_board = human_input(magic_square, game_board,plays[1], human_list, computer_list)
                print_gameboard(game_board,human_list,computer_list)
                i = i + 1

game(magic_square, game_board, plays,human_list, computer_list, draw, lost)  # main function of the code
```

SCREENSHOTS OF RUNNIG PROGRAM:

CASE 1 :

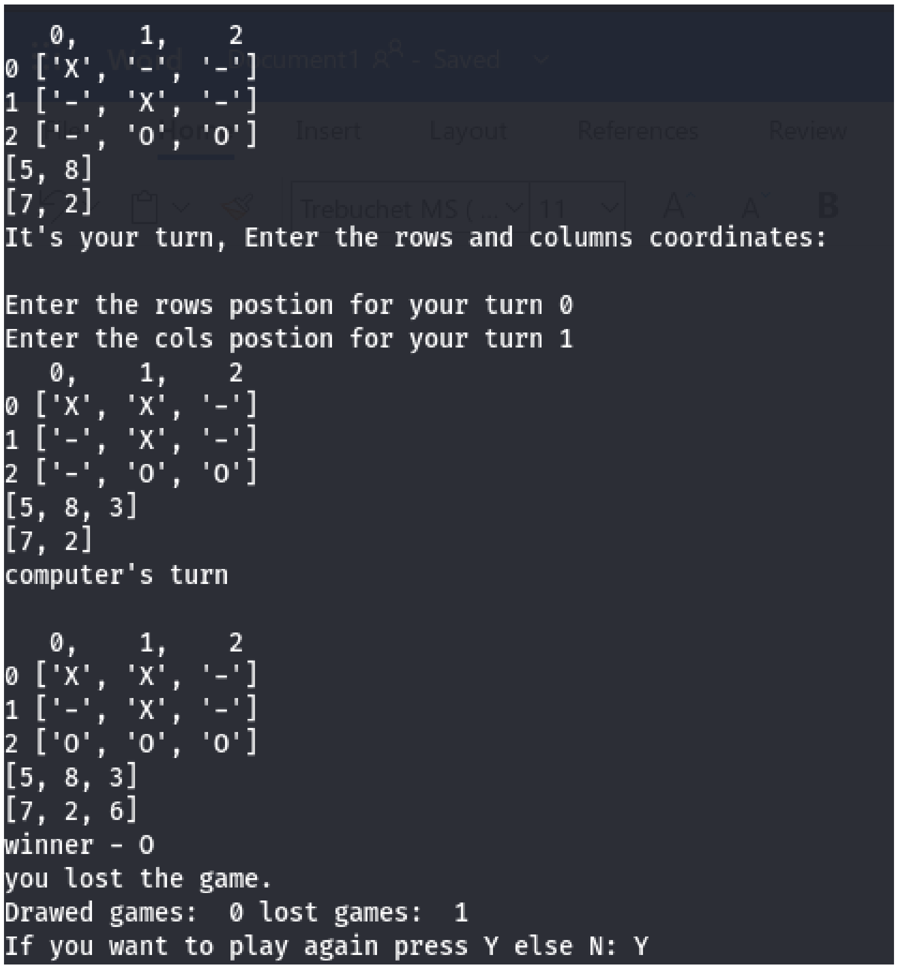
Human Won the Toss:



sandy@localhost: ~/Desktop

sandy@localhost: ~/Desktop   sandy@localhost: ~/Desktop

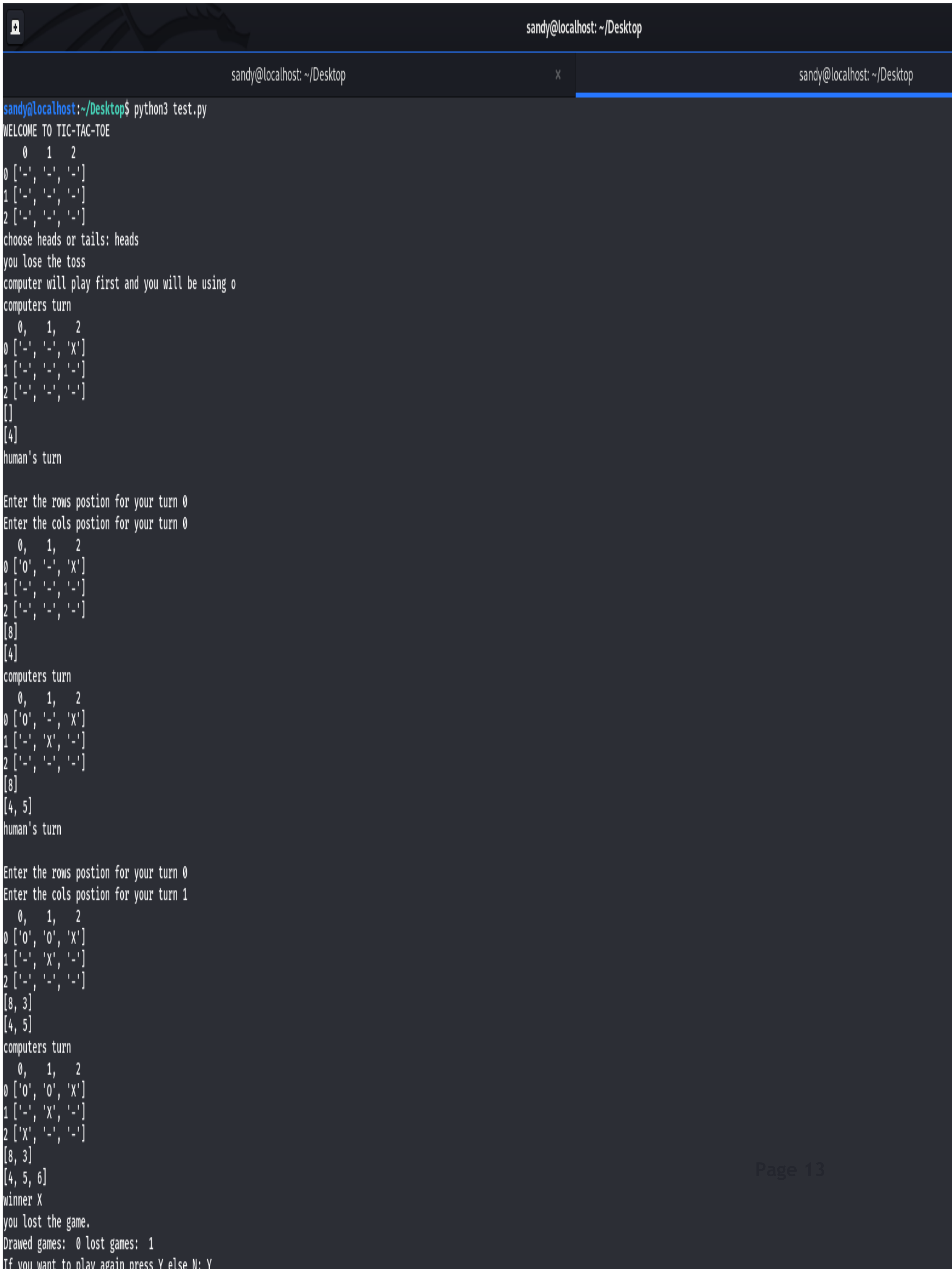
```
sandy@localhost:~/Desktop$ python3 test.py
WELCOME TO TIC-TAC-TOE
    0   1   2
0 ['-', '-', '-']
1 ['-', '-', '-']
2 ['-', '-', '-']
choose heads or tails: heads
you won the toss
Want to play first?
Y or N: Y
It's your turn, Enter the rows and columns coordinates:

Enter the rows postion for your turn 1
Enter the cols postion for your turn 1
   0,   1,   2
0 ['-', '-', '-']
1 ['-', 'X', '-']
2 ['-', '-', '-']
[5]
[]
computer's turn

   0,   1,   2
0 ['-', '-', '-']
1 ['-', 'X', '-']
2 ['-', 'O', '-']
[5]
[7]
It's your turn, Enter the rows and columns coordinates:

Enter the rows postion for your turn 0
Enter the cols postion for your turn 0
   0,   1,   2
0 ['X', '-', '-']
1 ['-', 'X', '-']
2 ['-', 'O', '-']
[5, 8]
[7]
computer's turn

   0,   1,   2
0 ['X', '-', '-']
1 ['-', 'X', '-']
2 ['-', 'O', 'O']
[5, 8]
[7, 2]
It's your turn, Enter the rows and columns coordinates:

Enter the rows postion for your turn 0
Enter the cols postion for your turn 1
   0,   1,   2
0 ['X', 'X', '-']
1 ['-', 'X', '-']
2 ['-', 'O', 'O']
```



```
   0,    1,    2
0 ['X', '-', '-']
1 ['-', 'X', '-']
2 ['-', 'O', 'O']
[5, 8]
[7, 2]
It's your turn, Enter the rows and columns coordinates:

Enter the rows postion for your turn 0
Enter the cols postion for your turn 1
   0,    1,    2
0 ['X', 'X', '-']
1 ['-', 'X', '-']
2 ['-', 'O', 'O']
[5, 8, 3]
[7, 2]
computer's turn

   0,    1,    2
0 ['X', 'X', '-']
1 ['-', 'X', '-']
2 ['O', 'O', 'O']
[5, 8, 3]
[7, 2, 6]
winner - O
you lost the game.
Drawed games:  0 lost games:  1
If you want to play again press Y else N: Y
```

CASE2:

COMPUTER WON THE TOSS:

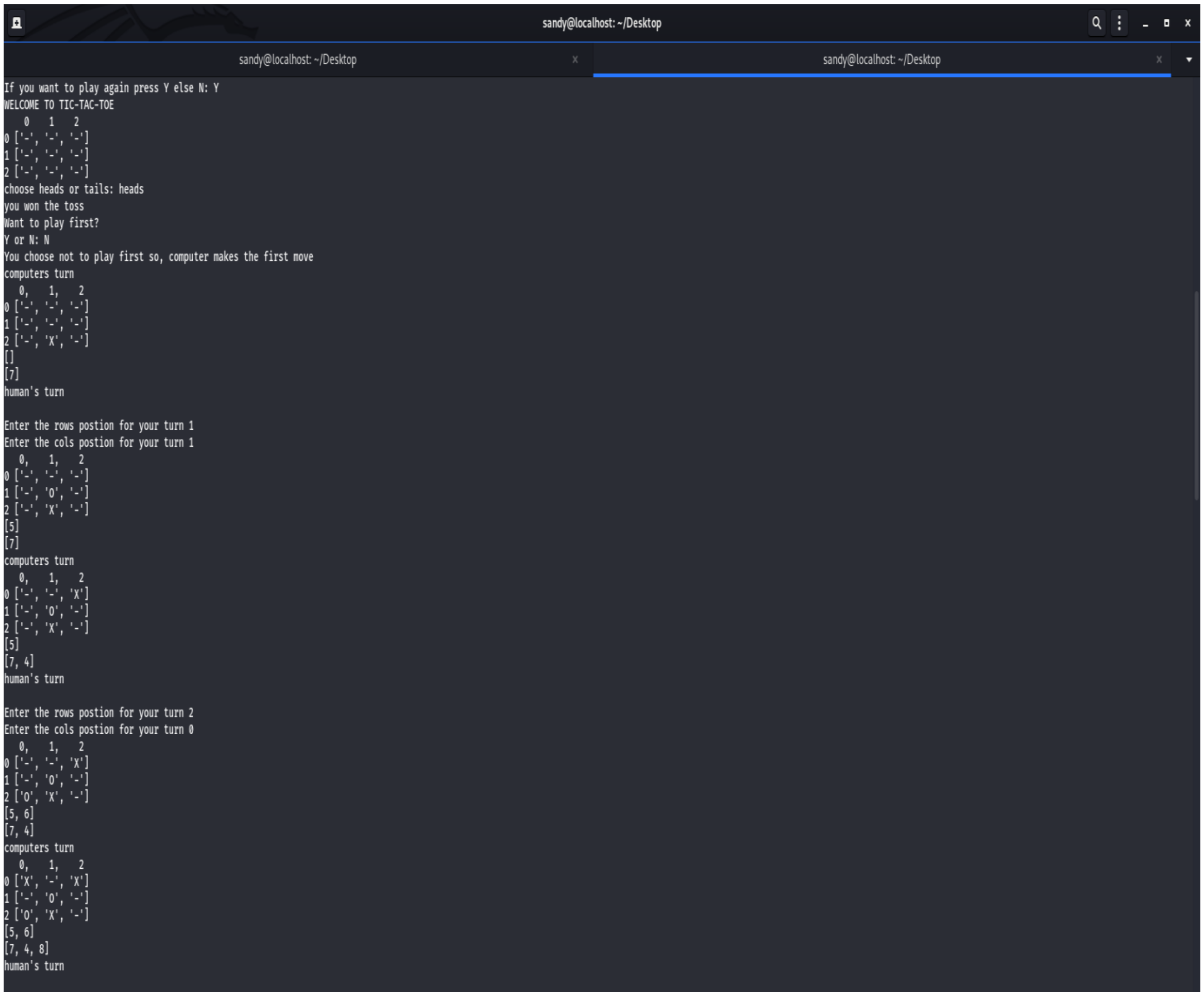


sandy@localhost: ~/Desktop

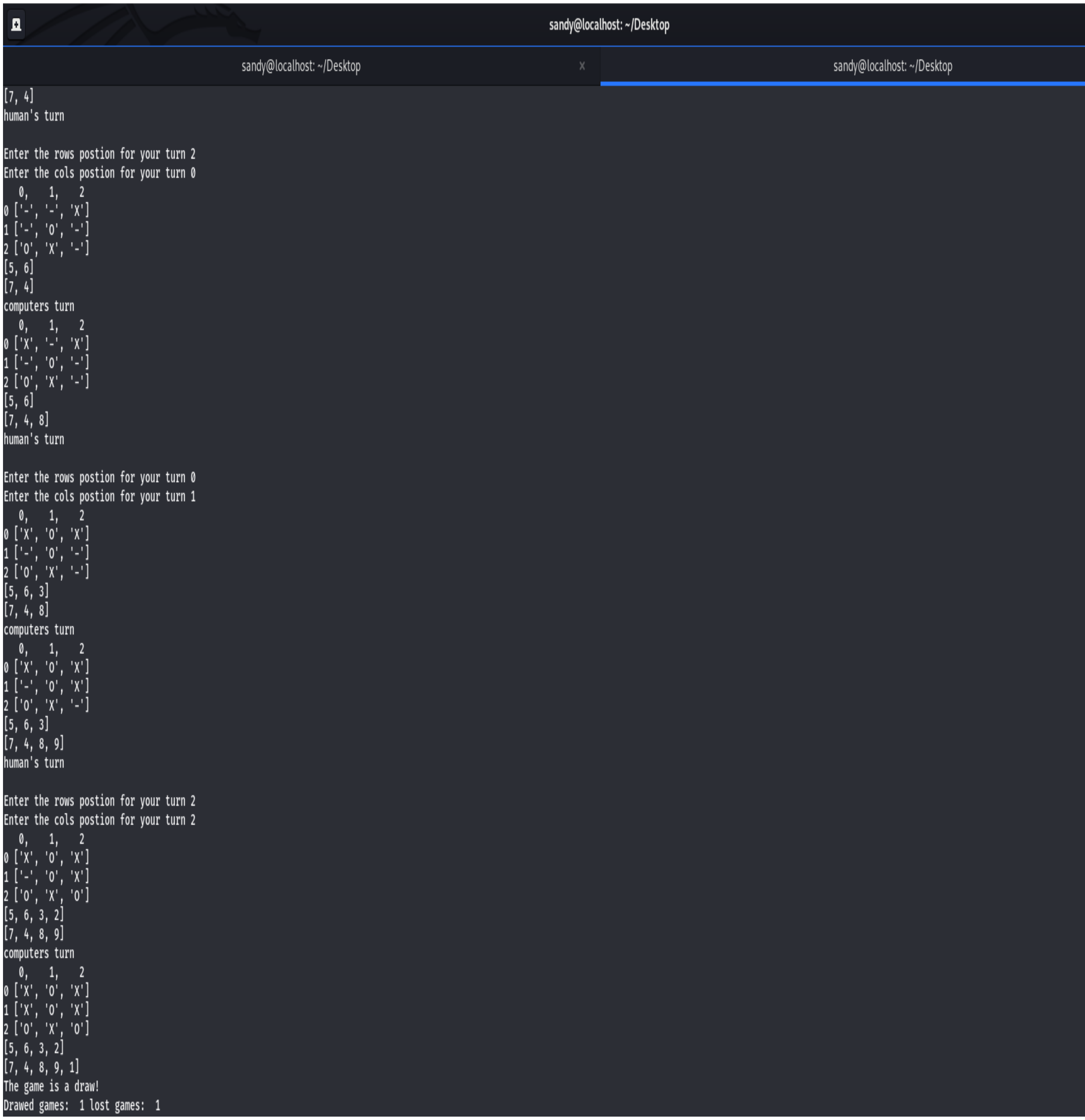
sandy@localhost: ~/Desktop x sandy@localhost: ~/Desktop

```
sandy@localhost:~/Desktop$ python3 test.py
WELCOME TO TIC-TAC-TOE
   0   1   2
0 ['-', '-', '-']
1 ['-', '-', '-']
2 ['-', '-', '-']
choose heads or tails: heads
you lose the toss
computer will play first and you will be using o
computers turn
  0,   1,   2
0 ['-', '-', 'X']
1 ['-', '-', '-']
2 ['-', '-', '-']
[]
[4]
human's turn

Enter the rows postion for your turn 0
Enter the cols postion for your turn 0
  0,   1,   2
0 ['O', '-', 'X']
1 ['-', '-', '-']
2 ['-', '-', '-']
[8]
[4]
computers turn
  0,   1,   2
0 ['O', '-', 'X']
1 ['-', 'X', '-']
2 ['-', '-', '-']
[8]
[4, 5]
human's turn

Enter the rows postion for your turn 0
Enter the cols postion for your turn 1
  0,   1,   2
0 ['O', 'O', 'X']
1 ['-', 'X', '-']
2 ['-', '-', '-']
[8, 3]
[4, 5]
computers turn
  0,   1,   2
0 ['O', 'O', 'X']
1 ['-', 'X', '-']
2 ['X', '-', '-']
[8, 3]
[4, 5, 6]
winner X
you lost the game.
Drawed games:  0 lost games:  1
If you want to play again press Y else N: Y
```

WE IMPLEMENTED A REPLAY FUNCTION IF HUMAN WANT TO PLAY AGAIN:


sandy@localhost: ~/Desktop
sandy@localhost: ~/Desktop
sandy@localhost: ~/Desktop

```
If you want to play again press Y else N: Y
WELCOME TO TIC-TAC-TOE
    0    1    2
0 ['-', '-', '-']
1 ['-', '-', '-']
2 ['-', '-', '-']
choose heads or tails: heads
you won the toss
Want to play first?
Y or N: N
You choose not to play first so, computer makes the first move
computers turn
   0,   1,   2
0 ['-', '-', '-']
1 ['-', '-', '-']
2 ['-', 'X', '-']
[]
[7]
human's turn

Enter the rows postion for your turn 1
Enter the cols postion for your turn 1
   0,   1,   2
0 ['-', '-', '-']
1 ['-', 'O', '-']
2 ['-', 'X', '-']
[5]
[7]
computers turn
   0,   1,   2
0 ['-', '-', 'X']
1 ['-', 'O', '-']
2 ['-', 'X', '-']
[5]
[7, 4]
human's turn

Enter the rows postion for your turn 2
Enter the cols postion for your turn 0
   0,   1,   2
0 ['-', '-', 'X']
1 ['-', 'O', '-']
2 ['O', 'X', '-']
[5, 6]
[7, 4]
computers turn
   0,   1,   2
0 ['X', '-', 'X']
1 ['-', 'O', '-']
2 ['O', 'X', '-']
[5, 6]
[7, 4, 8]
human's turn
```


sandy@localhost: ~/Desktop
sandy@localhost: ~/Desktop
sandy@localhost: ~/Desktop

```
[7, 4]
human's turn

Enter the rows postion for your turn 2
Enter the cols postion for your turn 0
   0,   1,   2
0 ['-', '-', 'X']
1 ['-', 'O', '-']
2 ['O', 'X', '-']
[5, 6]
[7, 4]
computers turn
   0,   1,   2
0 ['X', '-', 'X']
1 ['-', 'O', '-']
2 ['O', 'X', '-']
[5, 6]
[7, 4, 8]
human's turn

Enter the rows postion for your turn 0
Enter the cols postion for your turn 1
   0,   1,   2
0 ['X', 'O', 'X']
1 ['-', 'O', '-']
2 ['O', 'X', '-']
[5, 6, 3]
[7, 4, 8]
computers turn
   0,   1,   2
0 ['X', 'O', 'X']
1 ['-', 'O', 'X']
2 ['O', 'X', '-']
[5, 6, 3]
[7, 4, 8, 9]
human's turn

Enter the rows postion for your turn 2
Enter the cols postion for your turn 2
   0,   1,   2
0 ['X', 'O', 'X']
1 ['-', 'O', 'X']
2 ['O', 'X', 'O']
[5, 6, 3, 2]
[7, 4, 8, 9]
computers turn
   0,   1,   2
0 ['X', 'O', 'X']
1 ['X', 'O', 'X']
2 ['O', 'X', 'O']
[5, 6, 3, 2]
[7, 4, 8, 9, 1]
The game is a draw!
Drawed games:  1 lost games:  1
```