# HINDUSTAN COLLEGE OF SCIENCE & TECHNOLOGY

## FARAH , MATHURA

DEPARTMENT-> COMPUTER Sc. AND Engg.

PROJECT->TIC TAC TOE GAME USING PYTHON


**SUBMITTED TO :**

Mrs. SANJANA YADAV

(ASSISTANT PROFESSOR)

**SUBMITTED BY :**

RISHIKANT NAYAK(1806410084)

PRATYAKSH SHARMA(1806410073)

ESHAN TRIPATHI(1806410041)

## 1. Objectives:

Our project name is Tic-Tac-Toe game. This game is very popular and is fairly simple by itself. It is actually a two player game. In this game, there is a board with *n* x *n* squares. In our game, it is 3 x 3 squares.
The goal of Tic-Tac-Toe is to be one of the players to get three same symbols in a row - horizontally, vertically or diagonally - on a 3 x 3 grid.

## 2. Overview:

This game can be played in a 3x3 grid (shown in the figure 1) .The game can be played by two players.

(a) Player1 (b) Player2

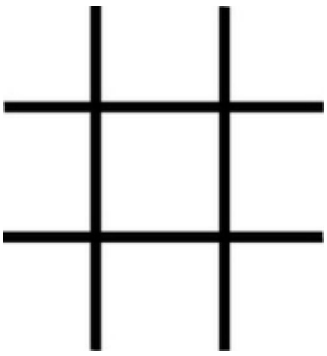


Figure: 1

## Players:

For playing this game, both the players are human , the first player is human and the second player is human.

## Theory of Game:

A player can choose between two symbols with his opponent, usual games use “X”and “O”. If first player choose “X” then the second player have to play with “O” and vice versa.

A player marks any of the 3x3 squares with his symbol (may be “X” or “O”) and his aim is to create a straight line horizontally or vertically or diagonally with two intensions:

a) Create a straight line before his opponent to win the game.

b) Restrict his opponent from creating a straight line first.

In case logically no one can create a straight line with his own symbol, the game results a draw.

Hence there are only three possible results – a player wins, his opponent wins or it’s a draw.

| | | |
|---|---|---|
| **1** | **2** | **3** |
| **4** | **5** | **6** |
| **7** | **8** | **9** |

Figure: 2

If any player is able to draw three Xs or three Os in the following combinations then that player wins. The combinations are:

a) 1, 2, 3 b) 4, 5, 6

c) 7, 8, 9 d) 1, 4, 7

e) 2, 5, 8 f) 3, 6, 9

h) 1, 5, 9 i) 3, 5, 7

## 3. Core Logic :

This is the core logic of this game – both players are human. Suppose the player1 use X and the player2 use O . The logic used for the AI is as follows:

### First move:

a) If the center is free, get the center. (Figure: 3.1)
b) Otherwise, get any of the corners. (Figure: 3.2)

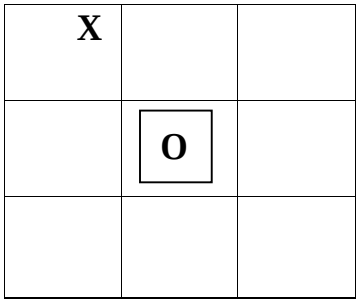



Figure: 3.1

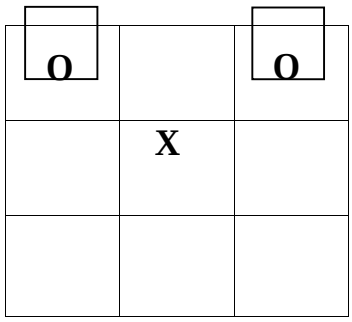



Figure: 3.2

### Second move:


a) Block user from winning. (Figure: 3.3)
b) Option for winning by applying the following logic:

If the center is occupied by user, get any of the corners. (Figure: 3.4)

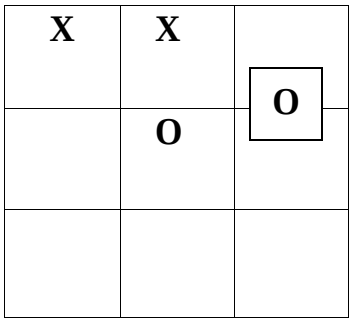



Figure: 3.3

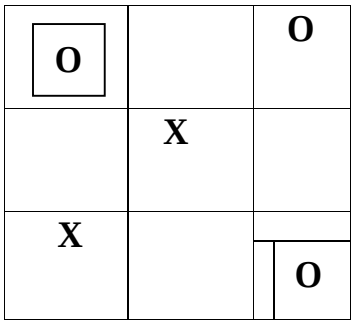



Figure: 3.4

Otherwise, the following cases happen:

## Case 1:

| | | |
|---|---|---|
| X | O | |
| O | O | O |
| | O | X |

Figure: 3.5

If any situation arises like the figure 3.5 then the opponent can set its symbol any one of the position among 2, 4, 6 and 8.

## Case 2:

| | | |
|---|---|---|
| | X | |
| 4 | O | 6 |
| | X | |

Figure: 3.6

| | | |
|---|---|---|
| X | | |
| 4 | O | 6 |
| | X | |

Figure: 3.7

| | | |
|---|---|---|
| | X | |
| 4 | O | 6 |
| | | X |

Figure: 3.8

If any situation arises like the figure 3.6 or figure 3.7 or figure 3.8 then the opponent can set its symbol at any position among 4 and 6.

## Case 3:

| | | |
|---|---|---|
| | 2 | |
| | O | X |
| X | 8 | |

Figure: 3.9

| | | |
|---|---|---|
| | 2 | |
| X | O | X |
| | 8 | |

Figure: 3.10

| | | |
|---|---|---|
| | 2 | |
| X | O | |
| | 8 | X |

Figure: 3.11

If any situation arises like the figure 3.9 or figure 3.10 or figure 3.11 then the opponent can set its symbol at any position among 2 and 8.

## Case 4:

| | | |
|---|---|---|
| 1 | X | 3 |
| | O | X |
| 7 | | 9 |

Figure: 3.12

| | | |
|---|---|---|
| 1 | X | 3 |
| X | O | |
| 7 | | 9 |

Figure: 3.13

| | | |
|---|---|---|
| 1 | | 3 |
| X | O | |
| 7 | X | 9 |

Figure: 3.14

| | | |
|---|---|---|
| 1 | | 3 |
| | O | X |
| 7 | X | 9 |

Figure: 3.15

If any situation arises like the figure 3.12 or figure 3.13 or figure 3.14 or 3.15 then the opponent can set its symbol at any position among 1, 3, 7 and 9.

## Third and fourth move:

a) Option for winning. (Figure: 3.16)
b) Block user from winning. (Figure: 3.17)
c) Randomly play a move. (Figure: 3.18)

| | | |
|---|---|---|
| O | | X |
| X | O | |
| X | | O |

Figure: 3.16

| | | |
|---|---|---|
| O | | |
| X | O | |
| X | | X |

Figure: 3.17

| | | | |
|---|---|---|---|
| X | X | O | O |
| O | O | X | |
| X | O | | |

Figure: 3.18

## 4. Core Logic - Humans:

For each move, check whether any 3 combination is occupied by any player and display the winner accordingly.

## 5. Technology used:

Jupyter notebook , python

## 6. Methods:

The methods we used in our program are as follows:

```
From IPython.display import clear_output:
```

For display the board.

**def player_input():**

To take in a player input and assign marker (‘X’ or ‘O’)

**def win_check(board,mark):**

To check the combination whether any two symbols (X or O) are same for winning or blocking .

**Import random**

**random.randint**

uses the random module to randomly decide which player goes first. You may want to lookup random.randint() Return a string of which player went first

def **full_board_check(board):**

To check full board free or full

**def** player_choice(board):

To select the choice of player in board .

def **replay():**

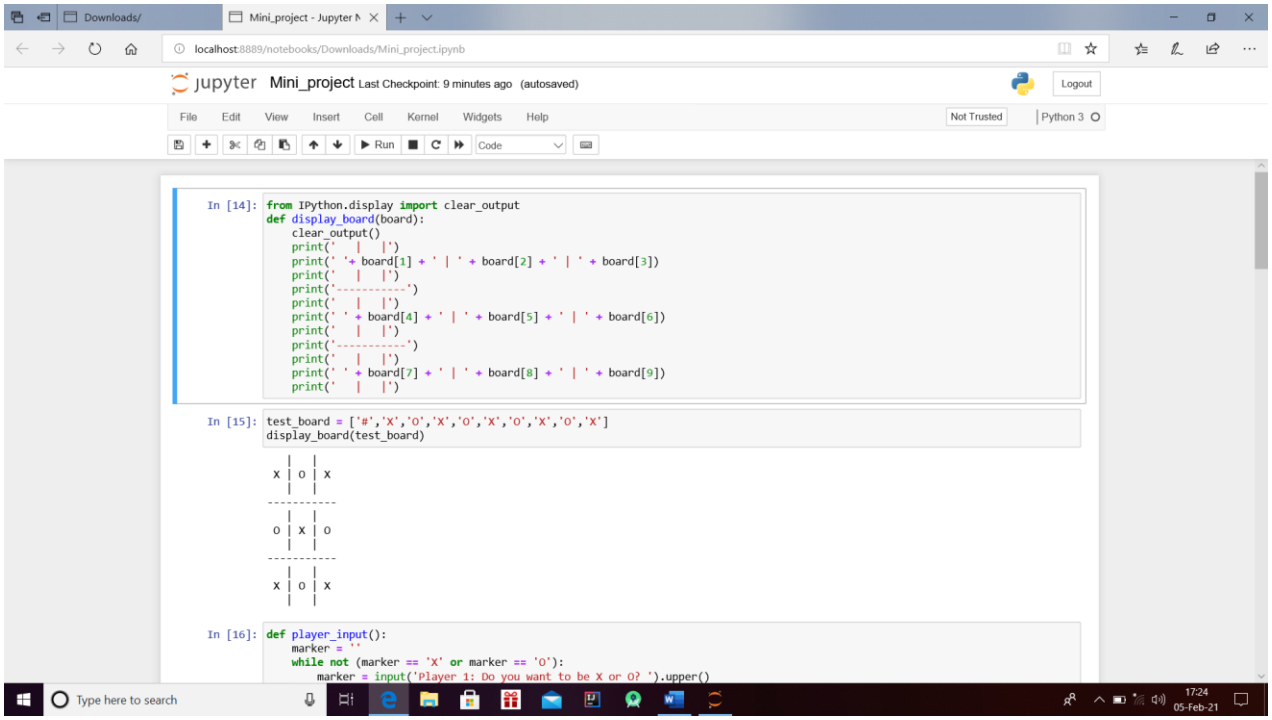
asks the player if they want to play again and returns a boolean True if they do want to play again.

**private void setCorner():**

To create scope for computer to win in the third move.

**Game logic:**

Here comes the hard part! Use while loops and the functions you've made to run the game!



Downloads/ | Mini_project - Jupyter N

localhost:8889/notebooks/Downloads/Mini_project.ipynb

jupyter **Mini_project** Last Checkpoint: 9 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

```
 |   |
```

In [16]:
```
def player_input():
    marker = ''
    while not (marker == 'X' or marker == 'O'):
        marker = input('Player 1: Do you want to be X or O? ').upper()
    if marker == 'X':
        return ('X', 'O')
    else:
        return ('O', 'X')
```

In [17]:
```
player_input()
```

```
Player 1: Do you want to be X or O? o
```

Out[17]:
```
('O', 'X')
```

In [18]:
```
def place_marker(board, marker, position):
    board[position] = marker
```

In [19]:
```
place_marker(test_board,'$',8)
display_board(test_board)
```

```
   |   |
 X | O | X
   |   |
-----------
   |   |
 O | X | O
   |   |
-----------
   |   |
 X | $ | X
   |   |
```

Type here to search

17:25
05-Feb-21

Downloads/ | Mini_project - Jupyter N

localhost:8889/notebooks/Downloads/Mini_project.ipynb

jupyter Mini_project Last Checkpoint: 10 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

```
   |   |
 x | $ | x
   |   |
```

In [20]:
```
def win_check(board,mark):
    return ((board[7] == mark and board[8] == mark and board[9] == mark) or
            (board[4] == mark and board[5] == mark and board[6] == mark) or
            (board[1] == mark and board[2] == mark and board[3] == mark) or
            (board[7] == mark and board[4] == mark and board[1] == mark) or
            (board[8] == mark and board[5] == mark and board[2] == mark) or
            (board[9] == mark and board[6] == mark and board[3] == mark) or
            (board[7] == mark and board[5] == mark and board[3] == mark) or
            (board[9] == mark and board[5] == mark and board[1] == mark))
```

In [21]:
```
win_check(test_board,'X')
```

Out[21]:
```
True
```

In [22]:
```
import random
def choose_first():
    if random.randint(0, 1) == 0:
        return 'Player 2'
    else:
        return 'Player 1'
```

In [23]:
```
def space_check(board, position):
    return board[position] == ' '
```

In [24]:
```
def full_board_check(board):
    for i in range(1,10):
        if space_check(board, i):
            return False
    return True
```

Type here to search | 17:26 05-Feb-21

Downloads/ | Mini_project - Jupyter N

localhost:8889/notebooks/Downloads/Mini_project.ipynb

jupyter Mini_project Last Checkpoint: 11 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

In [25]:
```
def player_choice(board):
    position = 0
    while position not in [1,2,3,4,5,6,7,8,9] or not space_check(board, position):
        position = int(input('Choose your next position: (1-9) '))

    return position
```

In [26]:
```
def replay():
    return input('Do you want to play again? Enter Yes or No: ').lower().startswith('y')
```

In [28]:
```
print('Welcome to Tic Tac Toe!')
while True:
# Reset the board
    theBoard = [' '] * 10
    player1_marker, player2_marker = player_input()
    turn = choose_first()
    print(turn + ' will go first.')

    play_game = input('Are you ready to play? Enter Yes or No.')
    if play_game.lower()[0] == 'y':
        game_on = True
    else:
        game_on = False

    while game_on:
        if turn == 'Player 1':
      # Player1's turn.
            display_board(theBoard)
            position = player_choice(theBoard)
            place_marker(theBoard, player1_marker, position)

            if win_check(theBoard, player1_marker):
                display_board(theBoard)
                print('Congratulations! You have won the game!')
                game_on = False
            else:
```

Type here to search | 17:27 05-Feb-21

Downloads/ | Mini_project - Jupyter N

localhost:8889/notebooks/Downloads/Mini_project.ipynb

jupyter Mini_project Last Checkpoint: 12 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

```
            if win_check(theBoard, player1_marker):
                display_board(theBoard)
                print('Congratulations! You have won the game!')
                game_on = False
            else:
                if full_board_check(theBoard):
                    display_board(theBoard)
                    print('The game is a draw!')
                    break
                else:
                    turn = 'Player 2'


        else:
             # Player2's turn.
            display_board(theBoard)
            position = player_choice(theBoard)
            place_marker(theBoard, player2_marker, position)

            if win_check(theBoard, player2_marker):
                display_board(theBoard)
                print('Player 2 has won!')
                game_on = False
            else:
                if full_board_check(theBoard):
                    display_board(theBoard)
                    print('The game is a draw!')
                    break
                else:
                    turn = 'Player 1'


    if not replay():
        break
```

Type here to search | 17:27 05-Feb-21

Downloads/ | Mini_project - Jupyter N

localhost:8889/notebooks/Downloads/Mini_project.ipynb

jupyter Mini_project Last Checkpoint: 15 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

```
        else:
            # Player2's turn.
            display_board(theBoard)
            position = player_choice(theBoard)
            place_marker(theBoard, player2_marker, position)

            if win_check(theBoard, player2_marker):
                display_board(theBoard)
                print('Player 2 has won!')
                game_on = False
            else:
                if full_board_check(theBoard):
                    display_board(theBoard)
                    print('The game is a draw!')
                    break
                else:
                    turn = 'Player 1'


    if not replay():
        break
```

```
Welcome to Tic Tac Toe!
Player 1: Do you want to be X or O? x
Player 2 will go first.

Are you ready to play? Enter Yes or No.
```

In [ ]:

Type here to search

17:31
05-Feb-21

Downloads/ | Mini_project - Jupyter N

localhost:8889/notebooks/Downloads/Mini_project.ipynb

jupyter Mini_project Last Checkpoint: 18 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

```
                print('Player 2 has won!')
                game_on = False
            else:
                if full_board_check(theBoard):
                    display_board(theBoard)
                    print('The game is a draw!')
                    break
                else:
                    turn = 'Player 1'


    if not replay():
        break
```

```
   |   |
 O |   | X
   |   |
-----------
   |   |
   | O | X
   |   |
-----------
   |   |
   |   | O
   |   |
Player 2 has won!
Do you want to play again? Enter Yes or No: no
```

In [ ]:

Type here to search | 17:34 05-Feb-21

Downloads/ | Mini_project - Jupyter N

localhost:8889/notebooks/Downloads/Mini_project.ipynb

jupyter Mini_project Last Checkpoint: 21 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Code

```
                if full_board_check(theBoard):
                    display_board(theBoard)
                    print('The game is a draw!')
                    break
                else:
                    turn = 'Player 1'


    if not replay():
        break
```

```
   |   |
 X | X | X
   |   |
-----------
   |   |
   |   |
   |   |
-----------
   |   |
   | O | O
   |   |
Player 2 has won!
Do you want to play again? Enter Yes or No: yes

Player 1: Do you want to be X or O?
```

In [ ]:

Type here to search | 17:37 05-Feb-21

## 7. Limitations:

1. GUI is not available.

2. Only two human player can play, not play with computer.

## 8. Future plan:

1. GUI will be added.

2. We want to design more complex boards for the game in future.

3. We want to add computer and human option.