

## Tic-tac-toe Algorithm

Step 1: Initialize a 2D array with 3 rows & 3 columns using '1' to differentiate columns & '-' to differentiate rows.

=>

|   |   |
|---|---|
| 1 | 1 |
| - | - |
| 1 | 1 |
| - | - |
| 1 | 1 |

~~print Board (board)~~

print ("1" \* 7)

print ("- " \* 7)

Step 2: In the main function, give the human the first move always.

Let X be human & 'O' be the system

put

current player as 'X' & computer player as 'O'.

Step 3: Check and if there is a winner after each move, if there is display winner @ and update the checker board and display it to take further inputs and also check if board is full & display it



- check winner  
 let 0-2 be the rows & columns  
 check rows, columns & diagonals  
 for the winner.  
 for i in range(3):  
   if board[i][0] == board[i][1] ==  
   board[i][2] != " ":  
     return board[i][0]  
   if board[0][i] == board[1][i] ==  
   board[2][i] != " ":  
     return board[0][i]  
   if board[0][0] == board[1][1] ==  
   board[2][2] != " ":  
     return board[0][0]  
   if board[0][2] == board[1][1] ==  
   board[2][0] != " ":  
     return board[0][2]

return None.

→ check full board:

return all (cell != " " for row in board  
for cell in row)

→ if no win ↓

row = player / current player - enter row (0-2): "

col = int(input(f"player {  
current player}, enter column (0-2): "))  
if the entered space is  
empty update & check for win



→ for computer move:-

find winning move:- (board, player)

for i in range(3):

for j in range(3):

if board[i][j] == " ":

board[i][j] = player

if check\_winner(board) == player:

board[i][j] = " "

return (i, j)

board[i][j]

return None.

python code:-

import random

def print\_board(board):

for row in board:

print(" | ".join(row))

print(" - " \* 9)

def check\_winner(board):

for i in range(3):

if board[i][0] == board[i][1] == board

[i][2] != " ":

return board[i][0]

if board[0][i] == board[1][i] == board

[2][i] != " ":

return board[0][i]



classmate  
Date \_\_\_\_\_  
Page \_\_\_\_\_

```
if board[0][0] == board[1][1] == board[2][2]:  
    return board[0][0]  
if board[0][2] == board[1][1] == board[2][0]:  
    return board[0][2]  
return None
```

```
def is_full(board):  
    return all(cell != " " for row in board  
               for cell in row)
```

```
def find-winning_move(board, player):  
    for i in range(3):  
        for j in range(3):  
            if board[i][j] == " ":  
                board[i][j] = player  
                if check_winner(board) == player:  
                    board[i][j] = " "  
                    return (i, j)  
                board[i][j] = " "  
    return None
```

```
def get_computer_move(board):  
    move = find-winning_move(board, "O")  
    if move:  
        return move  
    move = find-winning_move(board, "X")  
    if move:  
        return move
```



```

if board[1][1] == " ":
    return (1, 1)
corners = [(0, 0), (0, 2), (2, 0), (2, 2)]
for corner in corners:
    if board[corner[0]][corner[1]] == " ":
        return corner
for i in range(3):
    for j in range(3):
        if board[i][j] == " ":
            return (i, j)

```

```

def tic-tac-toe():
    board = [[" " for _ in range(3)] for _ in range(3)]
    current_player = "x"
    computer_player = "o"

```

```

    print("Player x goes first.")

```

```

    while True:
        print_board(board)

```

```

        if current_player == "x":
            while True:
                try:

```

```

                    row = int(input("Player x,
enter the row (0-2): "))

```

```

                    col = int(input("Player x,
enter the column (0-2): "))
                    if board[row][col] == " ":
                        break

```



```
else:  
    print("cell is taken, try again")  
except (ValueError, IndexError):  
    print("Invalid input")  
else:  
    print("computer's turn...")  
    row, col = get_computer_move(board)  
    print(f"computer chooses row {row},  
    column {col}")
```

```
board[row][col] = current_player
```

```
winner = check_winner(board)
```

```
if winner:  
    print_board(board)  
    print(f"player {winner} wins")  
    break
```

```
if is_full_board():  
    print_board(board)  
    print("It's a tie!")  
    break
```

```
current_player = computer_player if  
current_player == "x" else "o"
```

```
if __name__ == "__main__":  
    tic_tac_toe()
```



output:

player x goes first

|   |   |
|---|---|
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |

player x, row=0  
player x col=0

|   |   |
|---|---|
| X | X |
|   |   |
|   |   |

computer turn

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

player 2, enter row(0-2): 2  
player 2, enter col(0-2): 0

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

20/9/21

Player X goes first.

```
|  |
-----
|  |
-----
|  |
-----
```

Player X, enter the row (0-2): 0

Player X, enter the column (0-2): 1

```
| X |
-----
|  |
-----
|  |
-----
```

Computer's turn...

Computer chooses row 1, column 1

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

Player X, enter the row (0-2): 2

Player X, enter the column (0-2): 0

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

Computer's turn...

Computer chooses row 0, column 0

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

Player X, enter the row (0-2): 2

Player X, enter the column (0-2): 2

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



Computer's turn...

Computer chooses row 2, column 1

O | X |

-----

| O |

-----

X | O | X

-----

Player X, enter the row (0-2): 0

Player X, enter the column (0-2): 2

O | X | X

-----

| O |

-----

X | O | X

-----

Computer's turn...

Computer chooses row 1, column 2

O | X | X

-----

| O | O

-----

X | O | X

-----

Player X, enter the row (0-2): 1

Player X, enter the column (0-2): 1

Cell is already taken! Try again.

Player X, enter the row (0-2): 1

Player X, enter the column (0-2): 0

O | X | X

-----

X | O | O

-----

X | O | X

-----

It's a tie!