GROUP OF INSTITUTIONS

PROJECT TOPIC : TIC -TAC TOE GAME USING PYTHON PROGRAMMING

SUBMITTED BY

~ SAMRIDDHA DAS (J-70)

~ SARAVANA SAIRAM (J-71)

~ SHILPA VINAYAK HEGDE (J-72)

~ SHRILAKSHMI P NAIK (J-73)

INTRODUCTION :

Tic-tac-toe also known as noughts and crosses is a paper and pencil game for two players, who take turns marking the spaces in a 3 x 3 grid traditionally. The player who succeeds in placing three of their marks in a horizontal, vertical or diagonal row wins the game. It is a zero-sum of perfect information game. This means that it is deterministic, with fully observable environments in which two agents act alternately and the utility values at the end of the game are always equal and opposite. Because of the simplicity of tic-tac-toe, it is often used as pedagogical tool inartificial intelligence to deal with searching of game trees. The optimal move for this game can be gained by using minimax algorithm, where the opposition between the utility functions makes the situation adversarial, hence requiring adversarial search supported by minimax algorithm with alpha beta pruning concept in artificial intelligence

. 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 nxn 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.

2.2 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 tie. Hence there are only three possible results – a player wins, his opponent(human or computer) wins or it’s a tie

A TIC-TAC-TOE BOARD

❖A tic-tac-toe board looks like a large hash symbol (#) with nine slots that can each contain an X, an O, or a blank. To represent the board with a dictionary, we can assign each slot a string value key, as shown in Figure 5-2.

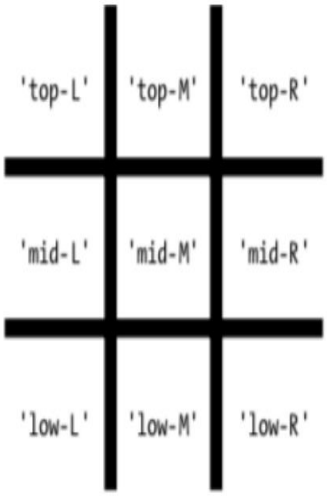


Figure 5-2: The slots of a tic-tac-toe board with their corresponding keys

❖ This dictionary is a data structure that represents a tic-tac-toe board. Store this board-as-a dictionary in a variable named Board.

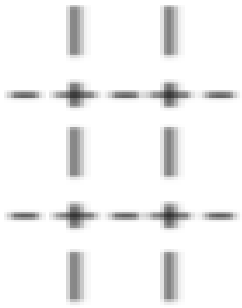
❖Open a new file editor window, and enter the following source code and save it :

```
Board = {'top-L': ' ', 'top-M': ' ', 'top-R': ' ',
 'mid-L': ' ', 'mid-M': ' ', 'mid-R': ' ',
 'low-L': ' ', 'low-M': ' ', 'low-R': ' '}
```

This will create an empty tic-tac-toe board.

- For an empty tic tac toe board we used the following code:

```
def printBoard(board):
 print(board['top-L'] + '|' + board['top-M'] + '|' + board['top-R'])
 print('-+-+-')
 print(board['mid-L'] + '|' + board['mid-M'] + '|' + board['mid-R'])
 print('-+-+-')
 print(board['low-L'] + '|' + board['low-M'] + '|' + board['low-R'])
printBoard(theBoard)
```



CODE SNIPPETS :

Code + Text

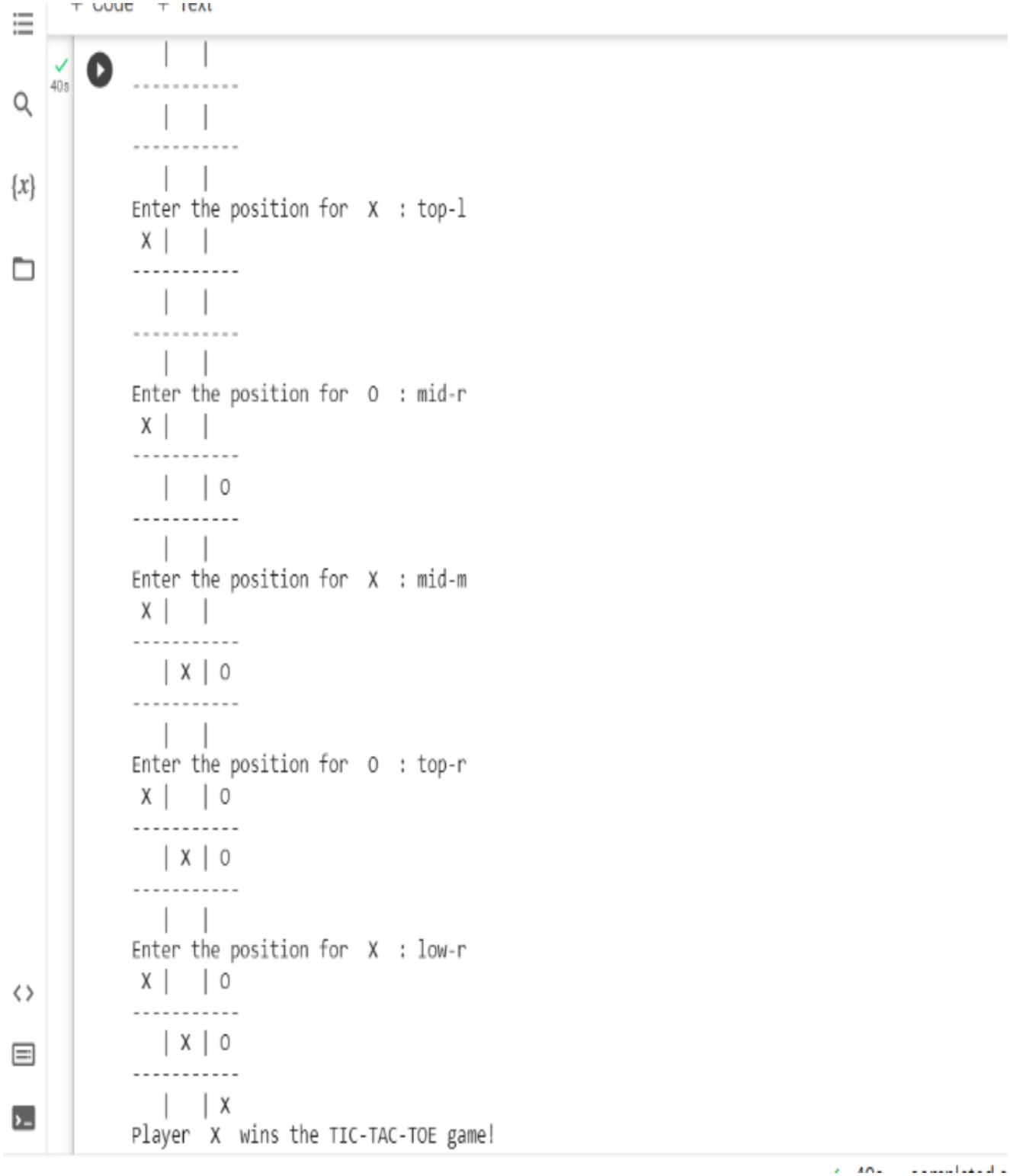
```
#tic-tac-toe
board = {'top-l':'   ', 'top-m':'   ', 'top-r':'   ',
        'mid-l':'   ', 'mid-m':'   ', 'mid-r':'   ',
        'low-l':'   ', 'low-m':'   ', 'low-r':'   '}

def printBoard(board):
    print(board['top-l'] + '|' + board['top-m'] + '|' + board['top-r'])
    print('-----------')
    print(board['mid-l'] + '|' + board['mid-m'] + '|' + board['mid-r'])
    print('-----------')
    print(board['low-l'] + '|' + board['low-m'] + '|' + board['low-r'])

def checkBoard(board):
    options = [" X ", " O "]
    if board['top-l'] == board['mid-m'] == board['low-r'] and board['top-l'] in options:
        return True
    elif board['top-r'] == board['mid-m'] == board['low-l'] and board['top-r'] in options:
        return True
    elif board['top-l'] == board['mid-l'] == board['low-l'] and board['top-l'] in options:
        return True
    elif board['top-m'] == board['mid-m'] == board['low-m'] and board['top-m'] in options:
        return True
    elif board['top-r'] == board['mid-r'] == board['low-r'] and board['top-r'] in options:
        return True
    elif board['top-l'] == board['top-m'] == board['top-r'] and board['top-l'] in options:
        return True
    elif board['mid-l'] == board['mid-m'] == board['mid-r'] and board['mid-l'] in options:
        return True
    elif board['low-l'] == board['low-m'] == board['low-r'] and board['low-l'] in options:
        return True
    else:
        return False
```

```
turn = " X "
for i in range(9):
    printBoard(board)
    ch = input("Enter the position for %s : " % turn)
    board[ch] = turn
    flag = checkBoard(board)
    if flag == True:
        printBoard(board)
        print("Player %s wins the TIC-TAC-TOE game!" % turn)
        break
    if turn == " X ":
        turn = " O "
    elif turn == " O ":
        turn = " X "
```

OUTPUT OF THE GAME :



CONCLUSION:

We have successfully developed the Tic-Tac-Toe game project in python. We explored the concept of dictionary, user defined function , if -else , and break statements, for loops , flags etc . In this way we successfully made a Tic-Tac-Toe game python project and enjoyed building tic-tac-toe game project.