

AI lab-1

Tik tac toe - program

```
import random as r
ai,player='O','X'
board=[['_','_','_'],['_','_','_'],['_','_','_']]
weights=[[3,2,3],[2,4,2],[3,2,3]]
def init():
    global ai,player,board,weights
    ai,player='O','X'
    board=[['_','_','_'],['_','_','_'],['_','_','_']]
    weights=[[3,2,3],[2,4,2],[3,2,3]]
def move(row,col,ch):
    if board[row][col]=='_':
        board[row][col],weights[row][col]=ch,0
        return True
    else : return False

def display(move_type='board'):
    if move_type=='cpu' : print('***5+CPU
MOVE'+***5)
    elif move_type=='board': print('***5+'
Board of Tic Tac Toe '+***5)
    else :print('***5+PLAYER MOVE'+***5)
    for i in range(3):
```

```
        print(board[i][j],end='\t')
    print('\n')
print('\n')
```

```
def compare_line(s1,ch):
    return '_' in s1 and s1.count(ch)==2
```

```
def get_position():
    max_value=max([max(x) for x in weights])
    positions=[(i,weights[i].index(max_value))
for i in range(3) if max_value in weights[i]]
    return positions
```

```
def has_tied():
    for row in board:
        if '_' in row: return False
    return True
```

```
def attacking_positiion(ch):
    default='_'
    for i in range(3):
        col=[board[0][i],board[1][i],board[2][i]]
        if compare_line(board[i],ch): return
(i,board[i].index(default))
        elif compare_line(col,ch): return
```

```

(col.index(default),i)
    diag1,diag2=[board[0][0],board[1]
[1],board[2][2]], [board[0][2],board[1]
[1],board[2][0]]
    if compare_line(diag1,ch):return
(diag1.index(default),diag1.index(default))
    elif compare_line(diag2,ch): return
(diag2.index(default),2-diag2.index(default))
    return False

```

```

def ai_move():
    global ai,player
    pos,f=attacking_positiion(ch=ai),False
    if pos!=False:(row,col),f=pos,True
    else :
        pos=attacking_positiion(ch=player)
        if pos!=False: row,col=pos
        else: row,col=r.choice(get_position())
    move(row,col,ai)
    return f

```

```

def run():
    global ai,player
    end,tied,move_type=False,False,None
    print('*'*10+ 'Tic Tac Toe'+ '*'*10+'\n')

```

```
display()
ch=input('Choose a Character X or O : ')
if ch=='O': ai,player=player,ai
while(True):
    if tied:
        print('***10+'The match is tied'+ '***10)
        return
    elif end:
        print('***10+move_type+' has own
'+ '***10)
        return
    move_type='player'
    r=int(input("\nEnter next move's row (1 to
3): "))
    c=int(input("Enter next move's column (1
to 3): "))
    if not move(r-1,c-1,player):
        print("\nEnter proper positions!!")
    else:
        display(move_type=move_type)
        tied=has_tied()
        if tied: continue
        move_type='cpu'
        end=ai_move()
        display(move_type=move_type)
```

```

        c=int(input("Enter next move's column (1
to 3): "))
        if not move(r-1,c-1,player):
            print('\nEnter proper positions!!')
        else:
            display(move_type=move_type)
            tied=has_tied()
            if tied: continue
            move_type='cpu'
            end=ai_move()
            display(move_type=move_type)
            tied=has_tied()
def main():
    run()
    f='Y'
    while(f=='Y'or f=='y'):
        f=input('Do you want to play again Y or N:
')
        init()
        if f=='Y' or f=='y':run()
        print('\n\n'+'*'*10+' Thank You '+'*'*10)
main()

```

## Output:

The screenshot shows a web browser window with the URL `onlinegdb.com/online_python_compiler`. The browser's address bar and tabs are visible at the top. The main content area displays a Python script for a Tic Tac Toe game. The script includes a `main` function that handles user input and game logic. The game board is represented by a 3x3 grid. The output of the program shows the game progress, including user moves and computer moves.

```

def main():
    board = [' ' for _ in range(9)]
    while True:
        print_board(board)
        user_move = input("Do you want to play again? (Y/N)y\n")
        if user_move.lower() != 'y':
            break
        welcome = "Welcome to Tic Tac Toe!"
        print_board(board)
        user_move = input("Please select a position to place an 'X' (1-9): 1\n")
        board[user_move-1] = 'X'
        print_board(board)
        computer_move = input("Computer placed an 'O' in position 9 : \n")
        board[computer_move-1] = 'O'
        print_board(board)
        user_move = input("Please select a position to place an 'X' (1-9): 4\n")
        board[user_move-1] = 'X'
        print_board(board)
        computer_move = input("Computer placed an 'O' in position 7 : \n")
        board[computer_move-1] = 'O'
        print_board(board)

```

The output of the program shows the game progress, including user moves and computer moves. The game board is represented by a 3x3 grid. The output shows the user placing 'X' at position 1, the computer placing 'O' at position 9, the user placing 'X' at position 4, and the computer placing 'O' at position 7.

The screenshot shows a web browser window with the URL `onlinegdb.com/online_python_compiler`. The browser's address bar and tabs are visible at the top. The online compiler interface includes a file explorer on the left with a file named `main.py`. The main editor area contains the following Python code:

```
def main():
    board = [' ' for _ in range(9)]
    for i in range(3):
        for j in range(3):
            board[i*3+j] = 'X' if i+j%2 == 0 else 'O'
    print_board(board)
    if check_win(board):
        print("Game Over!")
    else:
        print("Game Over!")
```

The terminal window on the right shows the output of the program, which is a Tic Tac Toe game. The game board is displayed as a 3x3 grid. The game is over with 'O' winning. The terminal output is as follows:

```

Please select a position to place an 'X' (1-9): 4
X | |
---
X | |
---
| | O
Computer placed an 'O' in position 7 :
X | |
---
X | |
---
C | | C
Please select a position to place an 'X' (1-9): 3
X | | X
---
X | |
---
C | | C
Computer placed an 'O' in position 8 :
X | | X
---
X | |
---
C | C | C
Sorry, O's won this time!
Do you want to play again? (Y/N)
```

The interface also includes a toolbar with buttons for Run, Debug, Stop, Share, and Save, and a language dropdown menu set to Python 3. The bottom of the screen shows the Windows taskbar with the search bar and several application icons.

Online Python Compiler - online x Tech With Tim - Python Tic Tac Toe x append means in python - Google x +

onlinegdb.com/online\_python\_compiler#

Gmail YouTube Maps

input

```
Welcome to Tic Tac Toe!
| |
| |
| |
Please select a position to place an 'X' (1-9): 1
X | |
| |
| |
Computer placed an 'O' in position 9 :
X | |
| |
| |
Please select a position to place an 'X' (1-9): 5
X | |
| X |
| |
Computer placed an 'O' in position 3 :
X | | O
| X |
| |
Please select a position to place an 'X' (1-9): 6
X | | O
| X |
| X |
```

Activate Windows  
Go to Settings to activate Windows.

Type here to search

26°C Haze 13:01 28-10-2021

Online Python Compiler - online x Tech With Tim - Python Tic Tac Toe x append means in python - Google x +

onlinegdb.com/online\_python\_compiler#

Gmail YouTube Maps

input

```
| | O
Please select a position to place an 'X' (1-9): 6
X | | O
| X | X
| | O
Computer placed an 'O' in position 4 :
X | | O
O | X | X
| | O
Please select a position to place an 'X' (1-9): 2
X | X | O
O | X | X
| | O
Computer placed an 'O' in position 8 :
X | X | O
O | X | X
| O | O
Please select a position to place an 'X' (1-9): 7
X | X | O
O | X | X
X | O | O
Tie Game!
Tie Game!
Do you want to play again? (Y/N)
```

Activate Windows  
Go to Settings to activate Windows.

Type here to search

26°C Haze 13:02 28-10-2021