Project Based Learning Report
on

Develop a 2D TIC-TAC-TOE Game

Submitted in the partial fulfillment of the requirements.
For the Project based learning in**:** ARTIFICIAL INTELLIGENCE AND DATA MINING
in
Electronics & Communication Engineering

By

SHIVAM SHASHWAT	PRN-2014111767
ADITYA TANDON	PRN-2014111124
ARYA TRIPATHI	PRN-2014111129

Under the guidance of Course In-charge
Dr. Tanuja. S. Dhope

Department of Electronics & Communication Engineering

Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 411043

Academic Year: 2023-24

Bharati Vidyapeeth
(Deemed to be University)
College of Engineering,
Pune – 411043

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CERTIFICATE

This is to be Certified that the Project Based Learning report entitled, "Develop a 2D TIC-TAC-TOE Game**"** Work is done by.

SHIVAM SHASHWAT	PRN-2014111767
ADITYA TANDON	PRN-2014111124
ARYA TRIPATHI	PRN-2014111129

In partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in ": ARTIFICIAL INTELLIGENCE AND DATA MINING" Bachelor of Technology Semester Vii, Electronics and Communication Engineering

Date:

Dr. Tanuja. S. Dhope — **Course In-charge**

Dr. Arundhati A. Shinde — **Professor & Head**

INDEX

Chapter 1

Problem Statement

Problem-

Develop a 2D TIC-TAC-TOE Game

Statement-

Make a graphical user interface for a 2D Tic-Tac-Toe game where two players can alternately position their symbols on a 3x3 grid. Accurate move validation, win condition checks for diagonal, vertical, and horizontal lines of three symbols, and an easy-to-use game status display should all be included in the game. The goal is to offer a fun and engaging digital version of this timeless board game.

Augmented Reality & Virtual Reality

Augmented Reality (AR) combines the digital world with real elements. It is a technology that is equally suitable for mobile devices and desktops. What makes it special is the fact that it offers the possibility of reflecting digital components in the real world.

How does Augmented Reality (AR) work?

One difference between VR and AR is that AR displays different content in the real world. Computer vision, depth tracking and mapping play a key role within this process. All data can be collected in real time via cameras, for example, and processed directly. This makes it possible to display digital content whenever the user needs it.

Special devices are required to fully use the functionality of AR. Smart Glasses, for example, are often used, which provide the data via Smart Glasses software.

Augmented Reality (AR): Advantages and disadvantages of the technology

If AR or VR is better is a question that cannot be answered in general terms. Both technologies have their advantages and disadvantages. These are some of the pros and cons of Augmented Reality:

Advantages:

- Enables individualized learning and enhances the learning process.
- AR offers a wide range of applications that are continuously being improved.
- The technology makes it possible to increase accuracy and efficiency.
- Experience or knowledge can be shared over long distances.

Disadvantages:

- The costs of implementing AR are comparatively high.
- Many devices have only a low level of performance.
- A key disadvantage is the lack of user privacy.
- If the focus on security is neglected, the introduction of augmented reality can lead to a security breach.

Application: Augmented Reality (AR) in practice

In practice, augmented reality offers a wide range of possibilities. This makes it interesting for both private and business users. Special apps can embed images, text, or videos.

- Fading in digital content over real magazines already works well in the printing and advertising industry.

- Users who want to translate texts into other languages can use modern translation apps thanks to AR technology.
- Augmented reality in construction and logistics is an attractive way to increase the efficiency of employees and the business processes.
- Augmented reality is an easy way to get in touch with customers, colleagues, or technicians**.**

Virtual world: What is Virtual Reality (VR)?

The main difference between AR vs VR is that VR is a computer-generated simulation. This means that reality or an alternative world is generated graphically.
By using appropriate hardware, it is possible for the user to be fully immersed in the digital world. Therefore, there are also important differences between AR headsets vs VR headsets**.** Hardware geared towards VR requires sensory devices that translate real-world movements into a modeled reality.
Here's how virtual reality (VR) works.

The focus of VR is to simulate a new reality. By using a VR screen, the user can perceive and interact in the digital world. This requires two lenses between the user and the screen. They interpret the movement of the eyes and adapt the individual movement to the VR. Therefore, in this case, extensive hardware is necessary to isolate the user from the real world.
Virtual Reality (VS): Pros and Cons

Every new technology has its very own pros and cons. This is also true for VR.

Advantages

- Immersive learning is possible in an interactive environment.
- Users can explore the virtual world in all its facets.
- The education sector benefits from these new possibilities.

Disadvantages

- A genuine interaction in the virtual environment is not possible.
- It is tempting to transfer one's life completely to the virtual world.
- Even though training or learning in the VR environment is very beneficial, it cannot completely replace the real training experience.
-

Practical application of virtual reality (VR)

Virtual Reality enjoys great popularity especially in the field of video games. Nevertheless, VR offers many other possible applications:

- In the military, this technology is used in flight simulators or battlefield simulations.
- In sports, digital training devices help athletes improve their own performance and analyse their techniques.

SOFTWARE

V S CODE

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality.

In the Stack Overflow 2023 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool among 86,544 respondents, with 73.71% reporting that they use it. It increased its use among those learning to code versus those developing as a profession (78% vs. 74%).

Visual Studio Code was first announced on April 29, 2015, by Microsoft at the 2015 Build conference. A preview build was released shortly thereafter.

On November 18, 2015, the source of Visual Studio Code was released under the MIT License and made available on GitHub. Extension support was also announced. On April 14, 2016, Visual Studio Code graduated from the public preview stage and was released to the Web. Microsoft has released most of Visual Studio Code's source code on GitHub under the permissive MIT License, while the binary releases by Microsoft are freeware, and include proprietary code. A community distribution, called VSCodium, is maintained, which provides MIT licensed binaries.

PYTHON

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.

Python consistently ranks as one of the most popular programming languages.

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which users enter statements sequentially and receive results immediately.

Python also comes with an Integrated development environment (IDE) called IDLE, which is more beginner-oriented.

Other shells, including IDLE and IPython, add further abilities such as improved auto-completion, session state retention, and syntax highlighting.

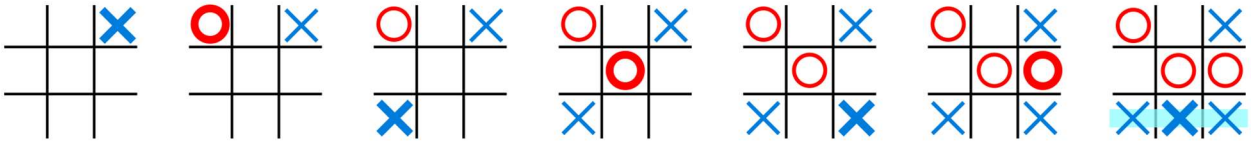
As well as standard desktop integrated development environments, there are web browser-based IDEs, including SageMath, for developing science- and math-related programs; PythonAnywhere, a browser-based IDE and hosting environment; and Canopy IDE, a commercial IDE emphasizing scientific computing

2D TIC TAC TOE

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

Tic-tac-toe is played on a three-by-three grid by two players, who alternately place the marks X and O in one of the nine spaces in the grid.

In the following example, the first player (*X*) wins the game in seven steps:



There is no universally agreed rule as to who plays first, but in this article the convention that X plays first is used.

Players soon discover that the best play from both parties leads to a draw. Hence, tic-tac-toe is often played by young children who may not have discovered the optimal strategy.

Because of the simplicity of tic-tac-toe, it is often used as a pedagogical tool for teaching the concepts of good sportsmanship and the branch of artificial intelligence that deals with the searching of game trees. It is straightforward to write a computer program to play tic-tac-toe perfectly or to enumerate the 765 essentially different positions (the state space complexity) or the 26,830 possible games up to rotations and reflections (the game tree complexity) on this space. If played optimally by both players, the game always ends in a draw, making tic-tac-toe a futile game.

The game can be generalized to an *m*,*n*,*k*-game, in which two players alternate placing stones of their own color on an *m*-by-*n* board with the goal of getting *k* of their own color in a row. Tic-tac-toe is the 3,3,3-game. Harary's generalized tic-tac-toe is an even broader generalization of tic-tac-toe. It can also be generalized as an n^d game, specifically one in which *n* equals 3 and *d* equals 2. It can be generalised even further by playing on an arbitrary incidence structure, where rows are lines and cells are points. Tic-tac-toe's incidence structure consists of nine points, three horizontal lines, three vertical lines, and two diagonal lines, with each line consisting of at least three points.

CODE

PYTHON

```
import math
X = 'X'
O = 'O'
EMPTY = ' '
def initialize_board():
    return [[EMPTY] * 3 for _ in range(3)]
def print_board(board):
    for row in board:
        print(" | ".join(row))
        print("---------")
def is_game_over(board):
    for row in board:
        if EMPTY in row:
            return False
    return True
def is_winner(board, player):
    for i in range(3):
        if all(board[i][j] == player for j in range(3)) or all(board[j][i] == player for j in range(3)):
            return True
    if all(board[i][i] == player for i in range(3)) or all(board[i][2 - i] == player for i in range(3)):
        return True
    return False
```

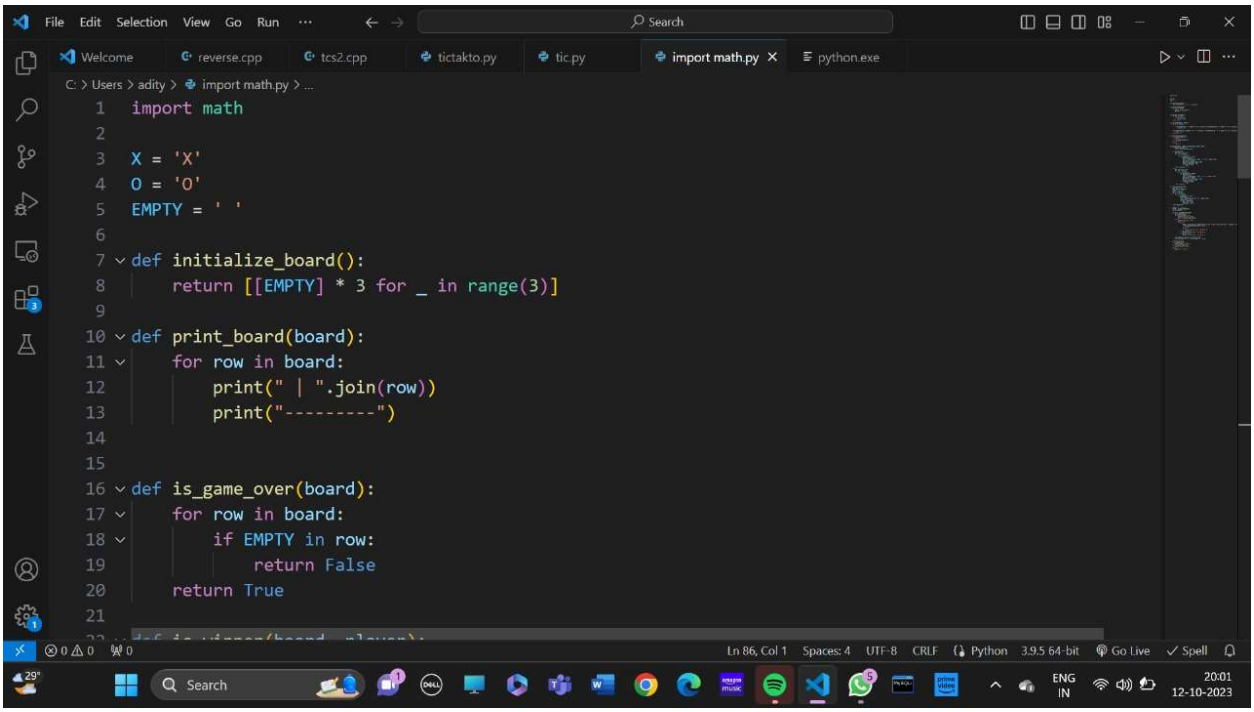
```
def evaluate_board(board):
    if is_winner(board, X):
        return 1
    elif is_winner(board, O):
        return -1
    else:
        return 0
def minimax(board, depth, is_maximizing, alpha, beta):
    if is_game_over(board):
        return evaluate_board(board)
    if is_maximizing:
        max_eval = -math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == EMPTY:
                    board[i][j] = X
                    eval = minimax(board, depth + 1, False, alpha, beta)
                    board[i][j] = EMPTY
                    max_eval = max(max_eval, eval)
                    alpha = max(alpha, eval)
                    if beta <= alpha:
                        break
        return max_eval
    else:
```

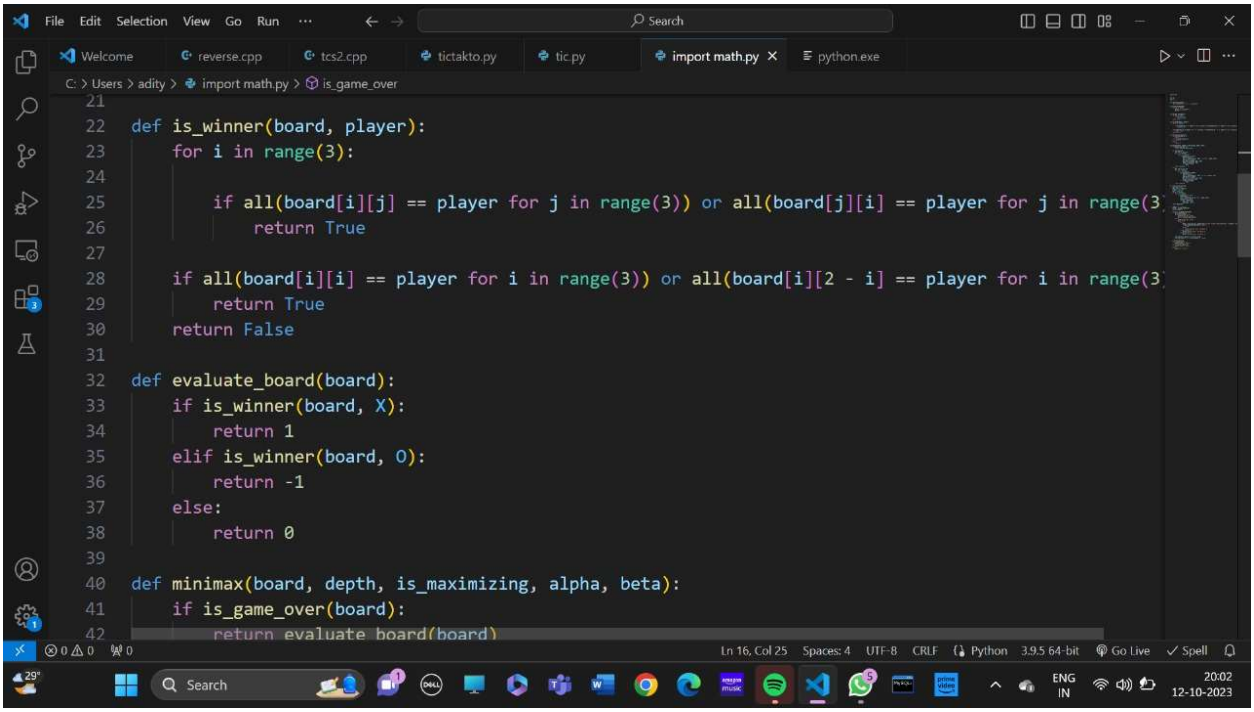
```
        min_eval = math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == EMPTY:
                    board[i][j] = O
                    eval = minimax(board, depth + 1, True, alpha, beta)
                    board[i][j] = EMPTY
                    min_eval = min(min_eval, eval)
                    beta = min(beta, eval)
                    if beta <= alpha:
                        break
        return min_eval
def find_best_move(board):
    best_move = None
    best_eval = -math.inf
    alpha = -math.inf
    beta = math.inf
    for i in range(3):
        for j in range(3):
            if board[i][j] == EMPTY:
                board[i][j] = X
                eval = minimax(board, 0, False, alpha, beta)
                board[i][j] = EMPTY
                if eval > best_eval:
```

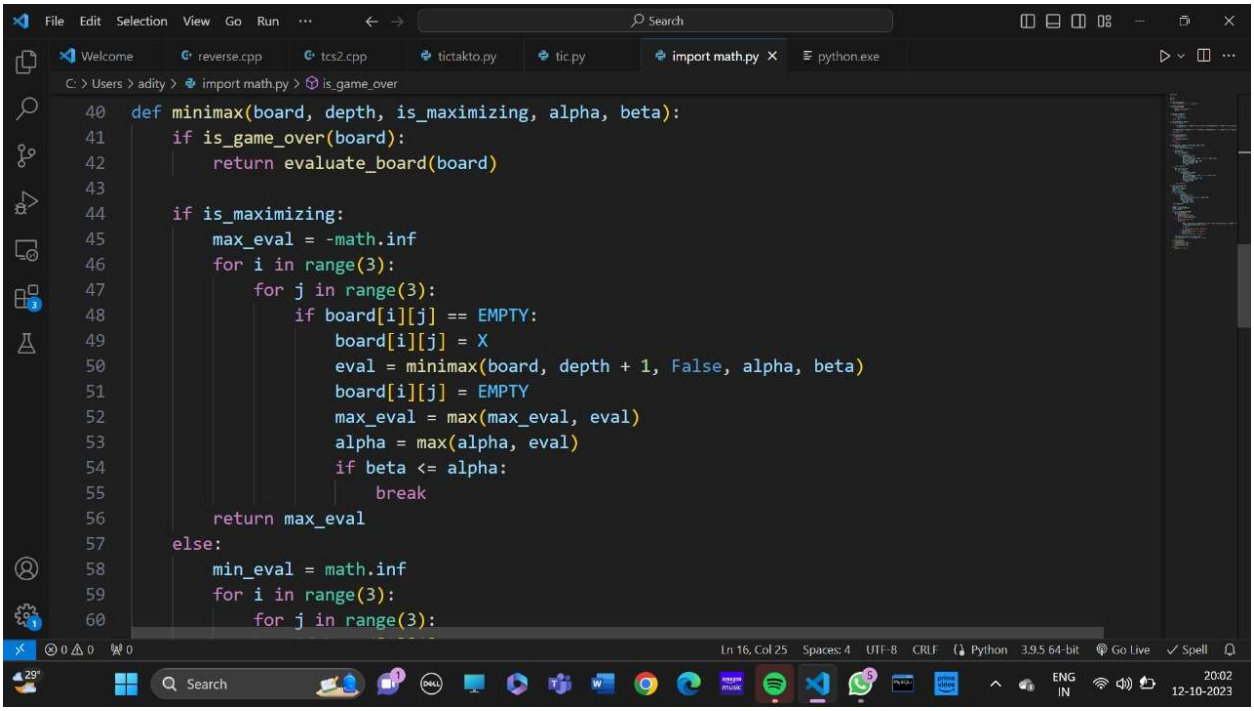
```
                best_eval = eval
                best_move = (i, j)
    return best_move
if _name_ == "_main_":
    board = initialize_board()
    current_player = X
    while not is_game_over(board):
        print_board(board)
        if current_player == X:
            print("Player X's turn")
            move = find_best_move(board)
        else:
            print("Player O's turn")
            while True:
                try:
                    move = tuple(map(int, input("Enter row and column (comma-separated): ").split(',')))
                    if board[move[0]][move[1]] == EMPTY:
                        break
                    else:
                        print("Invalid move. Try again.")
                except ValueError:
                    print("Invalid input. Try again.")
                except IndexError:
                    print("Invalid input. Try again.")
```

```
board[move[0]][move[1]] = current_player

        current_player = O if current_player == X else X

    print_board(board)

    if is_winner(board, X):

        print("Player X wins!")

    elif is_winner(board, O):

        print("Player O wins!")

    else:
        print("It's a tie!")
```

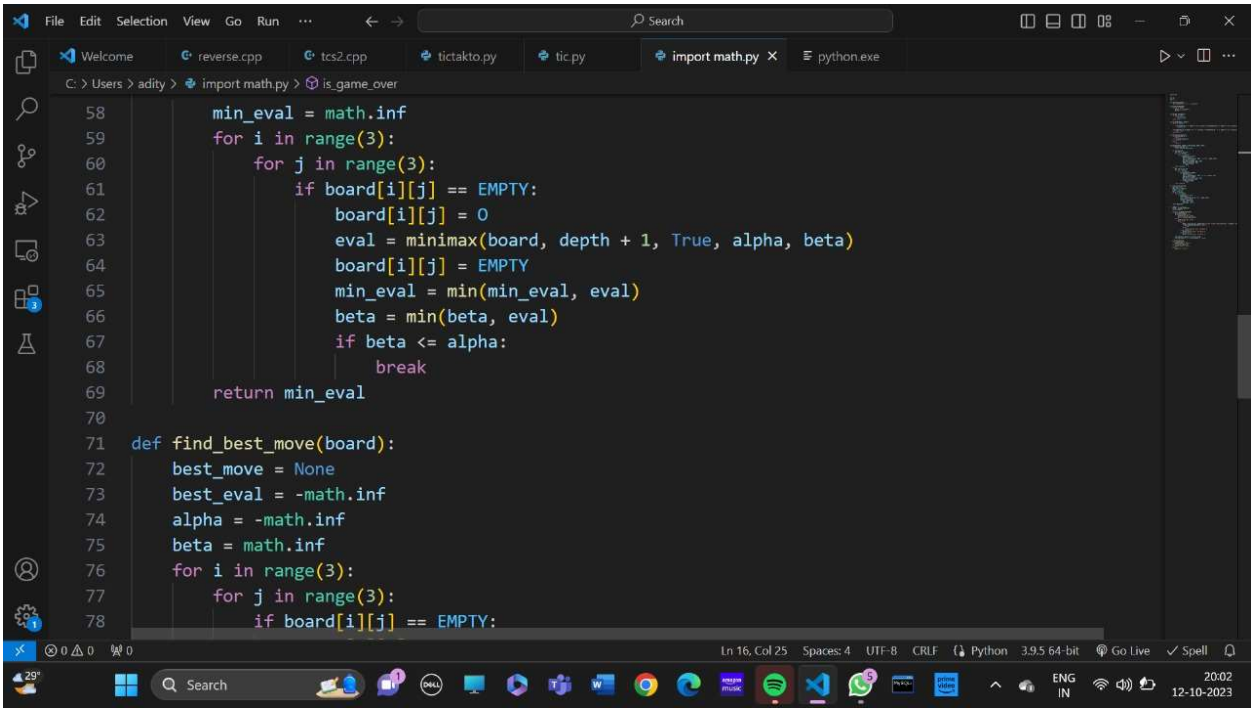
Code Screenshot



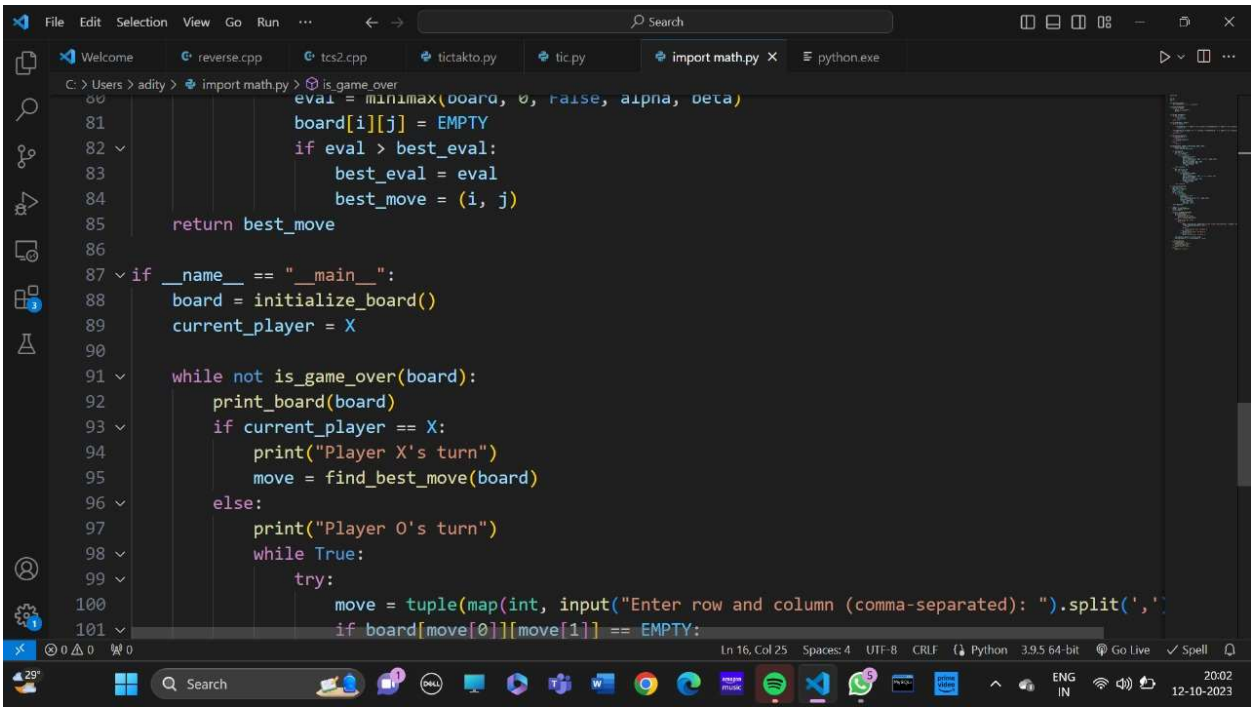

File Edit Selection View Go Run
Search
Welcome
reverse.cpp
tcs2.cpp
tictakto.py
tic.py
import math.py
python.exe
C: > Users > adity > import math.py > is_game_over
def is_winner(board, player):
for i in range(3):
if all(board[i][j] == player for j in range(3)) or all(board[j][i] == player for j in range(3
return True
if all(board[i][i] == player for i in range(3)) or all(board[i][2 - i] == player for i in range(3
return True
return False
def evaluate_board(board):
if is_winner(board, X):
return 1
elif is_winner(board, O):
return -1
else:
return 0
def minimax(board, depth, is_maximizing, alpha, beta):
if is_game_over(board):
return evaluate_board(board)
Ln 16, Col 25 Spaces: 4 UTF-8 CRLF Python 3.9.5 64-bit Go Live Spell
Search
ENG
IN
20:02
12-10-2023


File Edit Selection View Go Run
Search
Welcome
reverse.cpp
tcs2.cpp
tictakto.py
tic.py
import math.py
python.exe
C: > Users > adity > import math.py > is_game_over
def minimax(board, depth, is_maximizing, alpha, beta):
if is_game_over(board):
return evaluate_board(board)
if is_maximizing:
max_eval = -math.inf
for i in range(3):
for j in range(3):
if board[i][j] == EMPTY:
board[i][j] = X
eval = minimax(board, depth + 1, False, alpha, beta)
board[i][j] = EMPTY
max_eval = max(max_eval, eval)
alpha = max(alpha, eval)
if beta <= alpha:
break
return max_eval
else:
min_eval = math.inf
for i in range(3):
for j in range(3):
Ln 16, Col 25 Spaces: 4 UTF-8 CRLF Python 3.9.5 64-bit Go Live Spell
Search
ENG
IN
20:02
12-10-2023

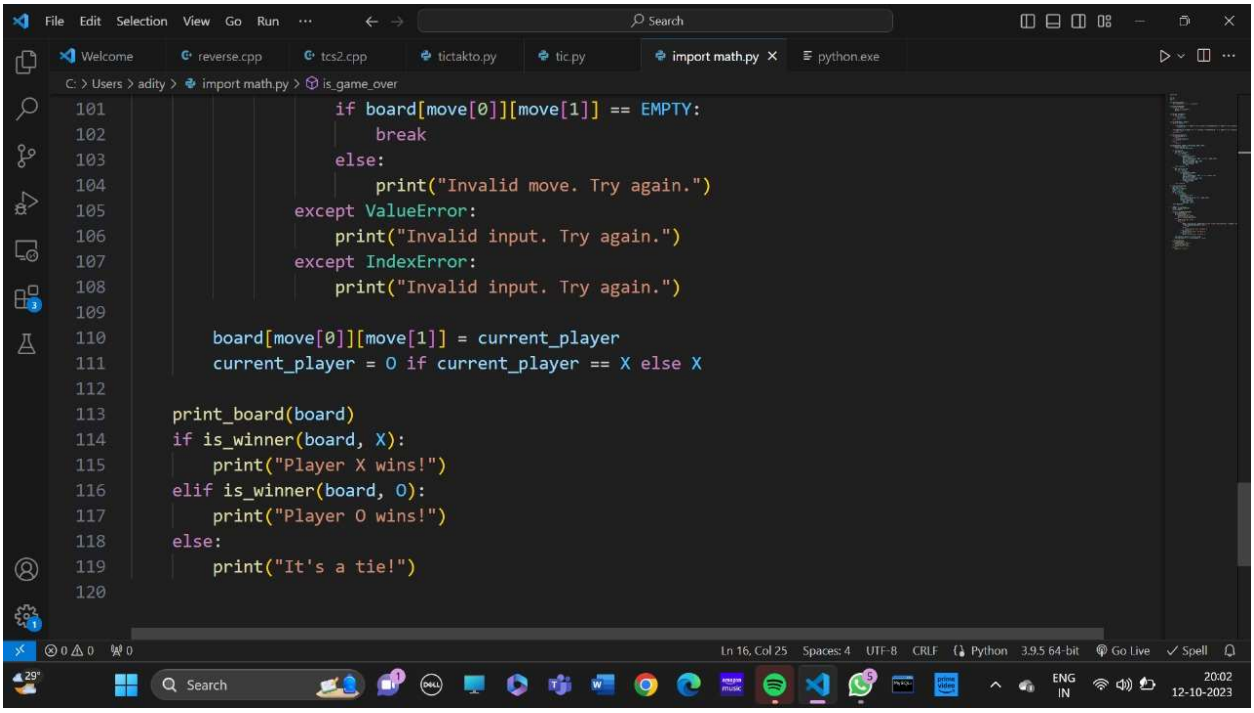
\



```
        min_eval = math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == EMPTY:
                    board[i][j] = O
                    eval = minimax(board, depth + 1, True, alpha, beta)
                    board[i][j] = EMPTY
                    min_eval = min(min_eval, eval)
                    beta = min(beta, eval)
                    if beta <= alpha:
                        break
        return min_eval

def find_best_move(board):
    best_move = None
    best_eval = -math.inf
    alpha = -math.inf
    beta = math.inf
    for i in range(3):
        for j in range(3):
            if board[i][j] == EMPTY:
```

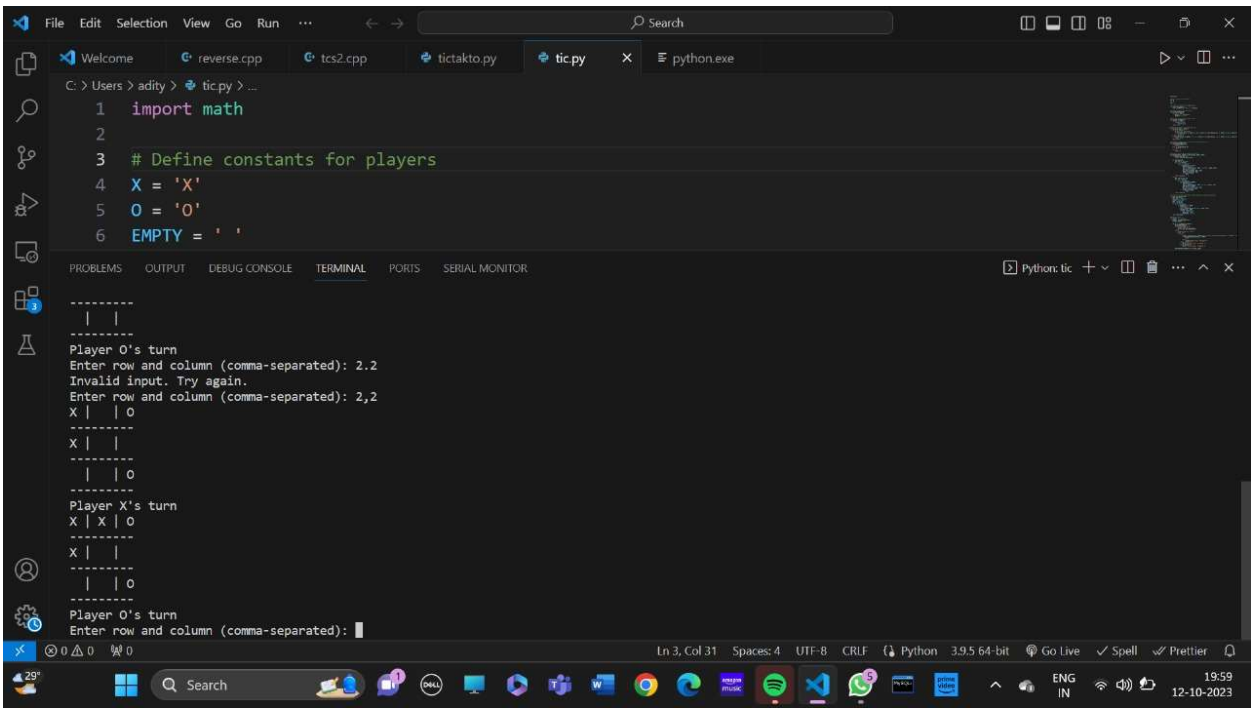


```
                    eval = minimax(board, 0, False, alpha, beta)
                    board[i][j] = EMPTY
                    if eval > best_eval:
                        best_eval = eval
                        best_move = (i, j)
    return best_move

if __name__ == "__main__":
    board = initialize_board()
    current_player = X

    while not is_game_over(board):
        print_board(board)
        if current_player == X:
            print("Player X's turn")
            move = find_best_move(board)
        else:
            print("Player O's turn")
            while True:
                try:
                    move = tuple(map(int, input("Enter row and column (comma-separated): ").split(',')
                    if board[move[0]][move[1]] == EMPTY:
```



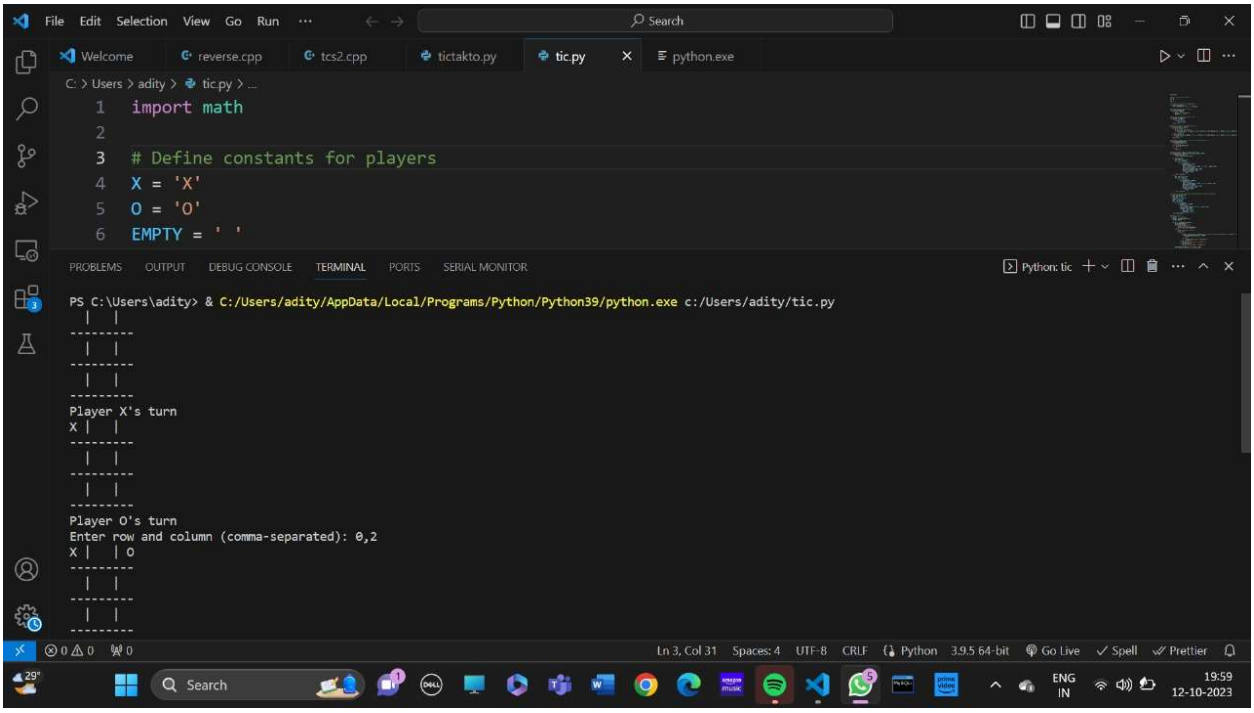
```
                    if board[move[0]][move[1]] == EMPTY:
                        break
                    else:
                        print("Invalid move. Try again.")
                except ValueError:
                    print("Invalid input. Try again.")
                except IndexError:
                    print("Invalid input. Try again.")

            board[move[0]][move[1]] = current_player
            current_player = O if current_player == X else X

    print_board(board)
    if is_winner(board, X):
        print("Player X wins!")
    elif is_winner(board, O):
        print("Player O wins!")
    else:
        print("It's a tie!")
```

OUTPUT-



```
import math

# Define constants for players
X = 'X'
O = 'O'
EMPTY = ' '
```

```
---------
  |   |
---------
Player O's turn
Enter row and column (comma-separated): 2.2
Invalid input. Try again.
Enter row and column (comma-separated): 2,2
X |   | O
---------
X |   |
---------
  |   | O
---------
Player X's turn
X | X | O
---------
X |   |
---------
  |   | O
---------
Player O's turn
Enter row and column (comma-separated):
```


File Edit Selection View Go Run
Search
Welcome
reverse.cpp
tcs2.cpp
tictakto.py
tic.py
python.exe
C: > Users > adity > tic.py > ...
1 import math
2
3 # Define constants for players
4 X = 'X'
5 O = 'O'
6 EMPTY = ' '
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SERIAL MONITOR
Python: tic
PS C:\Users\adity> & C:/Users/adity/AppData/Local/Programs/Python/Python39/python.exe c:/Users/adity/tic.py
| |

| |

| |

Player X's turn
X | |

| |

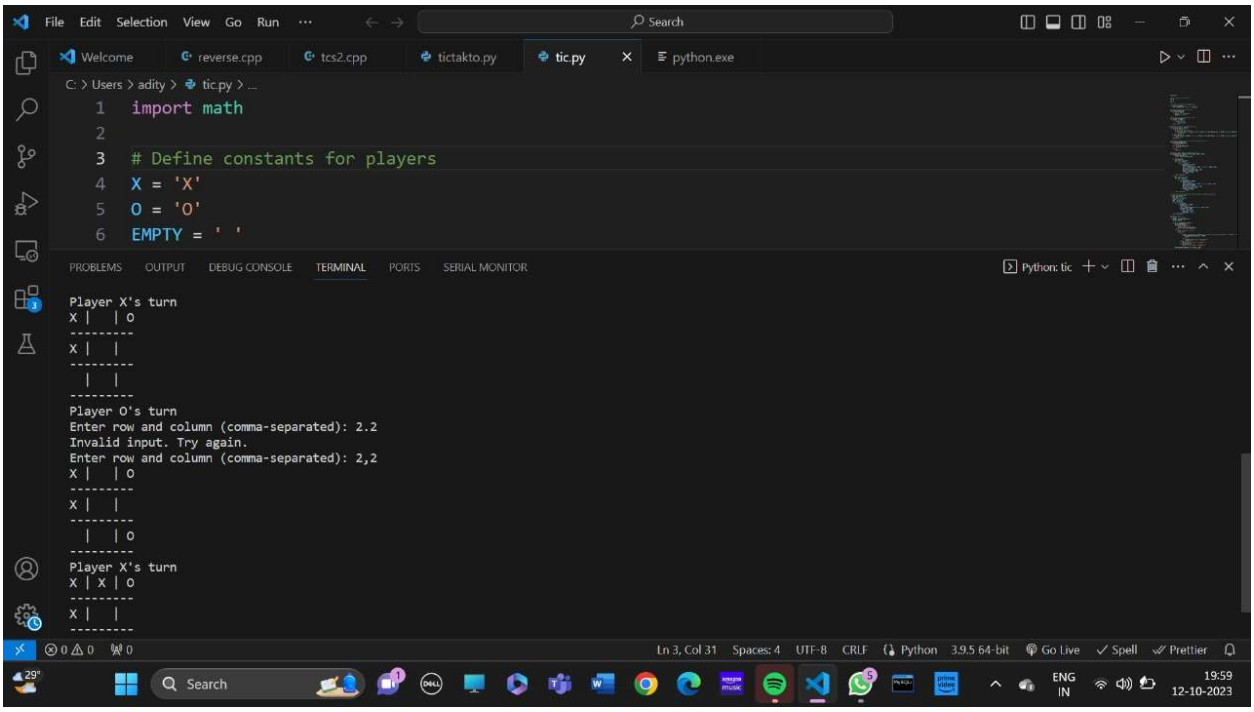
| |

Player O's turn
Enter row and column (comma-separated): 0,2
X | | O

| |

| |

Ln 3, Col 31 Spaces: 4 UTF-8 CRLF Python 3.9.5 64-bit Go Live Spell Prettier
Search
ENG IN
19:59
12-10-2023


File Edit Selection View Go Run
Search
Welcome
reverse.cpp
tcs2.cpp
tictakto.py
tic.py
python.exe
C: > Users > adity > tic.py > ...
1 import math
2
3 # Define constants for players
4 X = 'X'
5 O = 'O'
6 EMPTY = ' '
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SERIAL MONITOR
Python: tic
Player X's turn
X | | O

X | |

| |

Player O's turn
Enter row and column (comma-separated): 2.2
Invalid input. Try again.
Enter row and column (comma-separated): 2,2
X | | O

X | |

| | O

Player X's turn
X | X | O

X | |

Ln 3, Col 31 Spaces: 4 UTF-8 CRLF Python 3.9.5 64-bit Go Live Spell Prettier
Search
ENG IN
19:59
12-10-2023

Project outcome:

Our project on designing and analysing search techniques and game-playing techniques has yielded significant insights and practical results in the realm of artificial intelligence and game development. Throughout the project, we focused on two key aspects: improving search algorithms for decision-making and enhancing game-playing techniques for both traditional and modern games. Hence, our project satisfies the CO2 statement (design and analyse search techniques and game-playing techniques).

Project conclusion:

We have learnt from this project how the Python Tic-Tac-Toe project has effectively illustrated the ability of programming to create an iconic and engaging game. We have gained knowledge of and experience with basic Python programming concepts—such as data structures, loops, conditional statements, and user input handling—through this project.

We now have a great deal of expertise managing player input, creating engaging game logic, and making sure the user experience is seamless. In addition, we now know how to manage ties, apply win condition checks, and present the current state of the game in an intuitive way.

Appendix

Github Link: - TIC TAC TOE