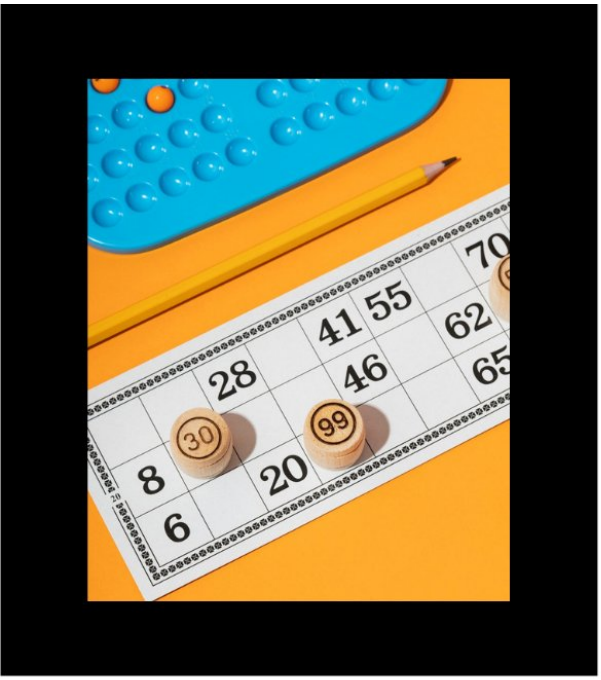
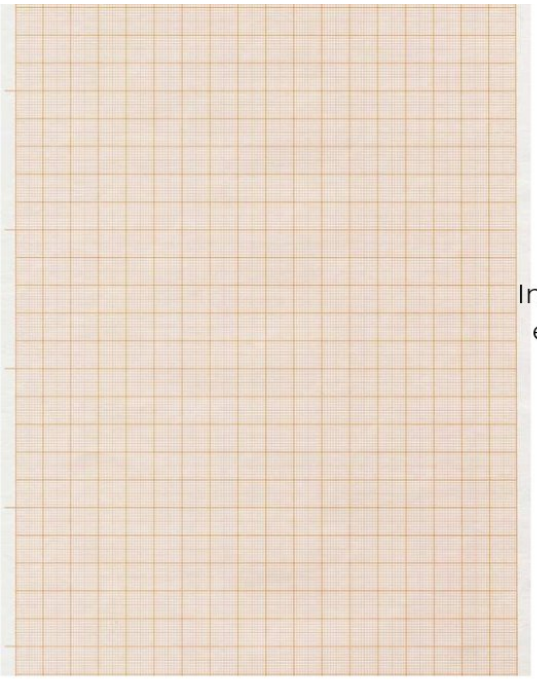
Unleashing Strategic Moves: A Comprehensive Presentation on the Tic Tac Toe Python Program



Introduction

Unleashing Strategic Moves

Welcome to the presentation on in the **Tic Tac Toe Python Program**. This comprehensive overview will delve into the intricacies of the program and its strategic implications. This project is done by Nithish, shenoz praveen, nabeel Ahmed, merylin Siva bharathi



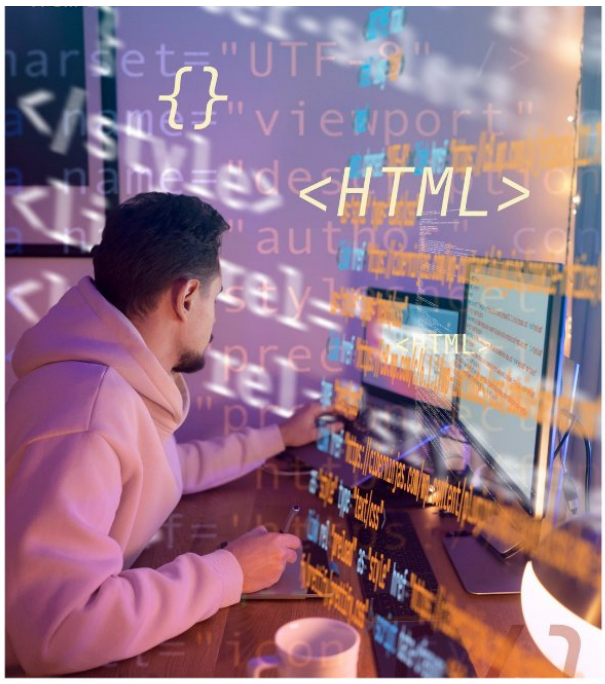
Tic Tac Toe Basics

In **Tic Tac Toe**, two players take turns marking an empty 3x3 grid with *X* and *O*. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins.

Python Implementation

Tic Tac Toe Python Program

The Tic Tac Toe Python Program leverages object-oriented programming to create a game environment. Python's simplicity and versatility make it an ideal choice for this implementation.



Strategic Moves

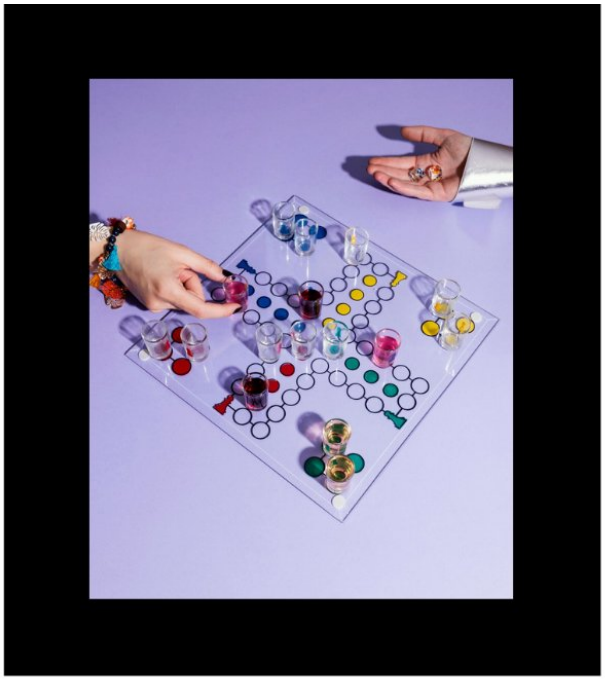
strategic moves

Understanding the strategic moves in Tic Tac Toe is crucial for success. This presentation will explore optimal strategies for both offensive and defensive gameplay, providing valuable insights.

Advanced Tactics

advanced tactics

Delving deeper into the program, we will examine advanced tactics such as creating fork opportunities, blocking opponent's forks, and understanding the concept of a 'threat'. These tactics can elevate gameplay to a new level.



Tic Tac Toe ppt

- 1. **Python Program TicTacToe Project Presentation** : By : Nithish, Nabeel ahmed, Merlyn Siva Bharathi, Shenoz Parveen
- 2. **PROJECT DESCRIPTION :** Written in Python programming language. This python project is based on CUI board-based game. All the gaming rules are the same. Reduces the manual struggle to play this game. This will provide lots of TIC TAC TOE matches without any error.
- 3. **A hash** shape square board grid. One of the players has to choose, 'O' and the other, 'X' to mark their respective cells. The player will have to input a numerical character, from 1 to 9, to select a position for X or O into the space . The game ends when one whole row/ column/ diagonal filled with his/her respective character ('O' or 'X'). If the blank spaces in the grid are all filled, and there is no winner, then the game is said to be a draw.
- 4. **SYSTEM DESIGN : Use** Case Diagram :- Flow Chart :- Start Gam e Make Move End Gam e TicTacToe Game Star t Compute Next step Who First Huma n draw Get new frame Decide Input Compute Next step Check win End

Program:

```
Def print_board(board):

For row in board:

Print(“ | “.join(row))

Print(“-“ * 9)


Def check_winner(board):

# Check rows

For row in board:

If len(set(row)) == 1 and row[0] != ‘ ‘:

Return row[0]


# Check columns

For col in range(3):

    If board[0][col] == board[1][col] == board[2][col] and board[0][col] != ‘ ‘:

Return board[0][col]


# Check diagonals

If board[0][0] == board[1][1] == board[2][2] and board[0][0] != ‘ ‘:

Return board[0][0]
```

```
 If board[0][2] == board[1][1] == board[2][0] and board[0][2] != ‘ ‘:
Return board[0][2]

Return None

Def is_board_full(board):
For row in board:
If ‘ ‘ in row:
Return False
Return True

Def is_valid_move(row, col, board):
Return 0 <= row < 3 and 0 <= col < 3 and board[row][col] == ‘ ‘

Def play_tic_tac_toe():
Board = [[‘ ‘ for _ in range(3)] for _ in range(3)]
Current_player = ‘X’

While True:
Print_board(board)
Print(f”Player {current_player}’s turn”)

Row = int(input(“Enter row (0, 1, or 2): “))
Col = int(input(“Enter column (0, 1, or 2): “))

If is_valid_move(row, col, board):
Board[row][col] = current_player
Winner = check_winner(board)
```

```
 If winner:
Print_board(board)
Print(f”Player {winner} wins!”)
Break
Elif is_board_full(board):
Print_board(board)
Print(“It’s a tie!”)
Break

        Current_player = ‘O’ if current_player == ‘X’ else ‘X’
Else:
Print(“Invalid move. Try again.”)

If __name__ == “__main__”:
Play_tic_tac_toe()
```

Output:

```
  |
-----
  |
-----
  |
-----
er X's turn
r row (0, 1, or 2): 0
r column (0, 1, or 2): 0
  |
-----
  |
-----
  |
-----
er O's turn
r row (0, 1, or 2): 0
r column (0, 1, or 2): 2
  | O
-----
  |
-----
  |
-----
er X's turn
r row (0, 1, or 2): 1
r column (0, 1, or 2): 1
  | O
-----
X |
-----
  |
-----
er O's turn
r row (0, 1, or 2): 1
r column (0, 1, or 2): 2
  | O
-----
X | O
-----
  |
-----
er X's turn
r row (0, 1, or 2): 2
r column (0, 1, or 2): 2
  | O
-----
X | O
-----
  | X
-----
er X wins!

gram finished]
```