Tic-Tac-Toe Game using Python:

Introduction:

Tic-Tac-Toe is a classic 2-player game where players take turns marking a 3×3 grid with "X" or "O". The first player to place three of their marks in a horizontal, vertical, or diagonal row wins the game. If the board fills up without a winner, the game ends in a tie.

This Python-based version of Tic-Tac-Toe runs in the terminal and includes:

Project Highlights

- **Player Interaction**: The game prompts for player names and alternates turns.
- **Board Display**: The grid updates in real-time with each move.
- **Game Logic**: Includes win condition checks and handles draw scenarios.
- **Input Validation**: Prevents illegal moves and catches invalid entries.
- **Fun Elements**: Emojis and prompts enhance user experience.

What You'll Learn

- How to structure a game using **functions**.
- Managing and updating a **2D list** (the game board).
- Using **loops**, **conditional logic**, and **global variables**.
- Implementing **error handling** with try/except.
- Designing a basic **turn-based game flow**.

This project is a great stepping stone for beginners who want to build logical thinking and understand how to turn game rules into working code.

Function Notes

initialize_game()

- Purpose: Sets up the game state.
- What it does:
  - Asks for both player names.
  - Creates an empty 3x3 board using a list of lists.
  - Defines all possible winning sequences (rows, columns, diagonals).
  - Sets current_player to Player 1.
- Global Variables Used:
  - player1, player2, board, winning_sequences, current_player

CODE:

```
def initialize_game():
    #Initialize game variables
    global player1, player2, board, winning_sequences, current_player
    player1 = input("Enter Player 1 Name: ").strip()
    player2 = input("Enter Player 2 Name: ").strip()
    board = [[" " for _ in range(3)] for _ in range(3)]
winning_sequences = [
        [(0, 0), (0, 1), (0, 2)],  # Row 1
        [(1, 0), (1, 1), (1, 2)],  # Row 2
        [(2, 0), (2, 1), (2, 2)],  # Row 3
```

```
        [(0, 0), (1, 0), (2, 0)],  # Column 1
        [(0, 1), (1, 1), (2, 1)],  # Column 2
        [(0, 2), (1, 2), (2, 2)],  # Column 3
        [(0, 0), (1, 1), (2, 2)],  # Diagonal 1
        [(0, 2), (1, 1), (2, 0)]   # Diagonal 2
    ]
current_player = player1
```

display_board()

- Purpose: Shows the current status of the game board.
- What it does:
  - Prints each row of the board with separators (| and -) to simulate a grid.
- Visual Benefit:
  - Helps players see the state of the game clearly after each move.

CODE:

```
def display_board():
    #Print the current board state
    print("\nCurrent Board:")
    for row in board:
        print(" | ".join(row))
        print("-" * 9)
```

take_input()

- Purpose: Gets and validates the move from the current player.
- What it does:
  - Prompts the current player for row and column values.
  - Validates the input:
    - Checks if row/col are within range (0–2).

        - Ensures the selected cell is empty.
    - Places the appropriate symbol ("X" or "O") in the selected cell.
- Error Handling:
    - Catches invalid input (non-integer or out of range).
    - Informs the user and re-prompts.

CODE:

```
def take_input():
    #Take input from the current player and update the board.
    global current_player
    while True:
        try:
            row = int(input(f"{current_player}'s Turn! Enter Row (0,1,2): "))
            col = int(input(f"{current_player}'s Turn! Enter Column (0,1,2): "))
            if 0 <= row < 3 and 0 <= col < 3 and board[row][col] == " ":
                board[row][col] = "X" if current_player == player1 else "O"
                return
            else:
                print("❌ Invalid move, try again.")
        except ValueError:
            print("⚠ Enter valid numbers (0, 1, or 2)!")
```

check_winner()

- Purpose: Checks whether the current player has won.
- What it does:
    - Loops through all predefined winning sequences.
    - Checks if the symbols in any of those sequences are all "X" or all "O".
    - Returns the name of the winning player if matched.
    - Returns None if no win is found.

- Logic:
    - Uses list comprehension and tuple unpacking for clean checking.

CODE:

```
def check_winner():
    #Check if the current player has won the game
    for sequence in winning_sequences:
        cells = [board[r][c] for r, c in sequence]
        if cells[0] == cells[1] == cells[2] and cells[0] != " ":
            return current_player
    return None
```

play_game()

- Purpose: Controls the game loop and manages turns.
- What it does:
    - Initializes the game.
    - Loops through a maximum of 9 turns.
    - After every move, it:
        - Displays the board.
        - Checks for a winner.
        - Switches players if no one has won yet.
    - If no one wins after 9 moves, declares a tie.
- Game Flow:
    - Uses current_player to track turns.
    - Alternates between player1 and player2 using a simple if-else.

CODE:

```
def play_game():
    #Main game loop.
    global current_player
```

```
initialize_game()
display_board()
for turn in range(9):
    take_input()
    display_board()
    winner = check_winner()
    if winner:
        print(f"🎉 Congratulations {winner}! You won the game! 🎉")
        return
    current_player = player1 if current_player == player2 else player2

print(f"🤝 {player1} and {player2}, it's a tie! 🤝")
```

EXECUTION OF CODE:

```
def initialize_game():
    #Initialize game variables
    global player1, player2, board, winning_sequences, current_player
    player1 = input("Enter Player 1 Name: ").strip()
    player2 = input("Enter Player 2 Name: ").strip()
    board = [[" " for _ in range(3)] for _ in range(3)]
   winning_sequences = [
        [(0, 0), (0, 1), (0, 2)],  # Row 1
        [(1, 0), (1, 1), (1, 2)],  # Row 2
        [(2, 0), (2, 1), (2, 2)],  # Row 3
        [(0, 0), (1, 0), (2, 0)],  # Column 1
        [(0, 1), (1, 1), (2, 1)],  # Column 2
        [(0, 2), (1, 2), (2, 2)],  # Column 3
        [(0, 0), (1, 1), (2, 2)],  # Diagonal 1
        [(0, 2), (1, 1), (2, 0)]   # Diagonal 2
    ]
    current_player = player1
def display_board():
    #Print the current board state
    print("\nCurrent Board:")
    for row in board:
        print(" | ".join(row))
        print("-" * 9)
```

```
def take_input():
    #Take input from the current player and update the board.
    global current_player
    while True:
        try:
            row = int(input(f"{current_player}'s Turn! Enter Row (0,1,2): "))
            col = int(input(f"{current_player}'s Turn! Enter Column (0,1,2): "))
            if 0 <= row < 3 and 0 <= col < 3 and board[row][col] == " ":
                board[row][col] = "X" if current_player == player1 else "O"
                return
            else:
                print("❌ Invalid move, try again.")
        except ValueError:
            print("⚠ Enter valid numbers (0, 1, or 2)!")
def check_winner():
    #Check if the current player has won the game
    for sequence in winning_sequences:
        cells = [board[r][c] for r, c in sequence]
        if cells[0] == cells[1] == cells[2] and cells[0] != " ":
            return current_player
    return None
def play_game():
    #Main game loop.
    global current_player
```

```
    initialize_game()
    display_board()
    for turn in range(9):
        take_input()
        display_board()
        winner = check_winner()
        if winner:
            print(f"🎉 Congratulations {winner}! You won the game! 🎉")
            return
        current_player = player1 if current_player == player2 else player2
print(f"🤝 {player1} and {player2}, it's a tie! 🤝")
# Start the game
play_game()
```

```
Enter Player 1 Name: Raju

Enter Player 2 Name:
```

```
Enter Player 1 Name: Raju
Enter Player 2 Name: Rani

Current Board:
  |   |
---------
  |   |
---------
  |   |
---------

Raju's Turn! Enter Row (0,1,2):
```

```
Raju's Turn! Enter Row (0,1,2): 0
Raju's Turn! Enter Column (0,1,2): 0

Current Board:
X |   |
---------
  |   |
---------
  |   |
---------


Rani's Turn! Enter Row (0,1,2): 1
Rani's Turn! Enter Column (0,1,2): 1

Current Board:
X |   |
---------
  | O |
---------
  |   |
---------

Raju's Turn! Enter Row (0,1,2):
```

```
Raju's Turn! Enter Row (0,1,2): 0
Raju's Turn! Enter Column (0,1,2): 1

Current Board:
X | X |
---------
  | O |
---------
  |   |
---------

Rani's Turn! Enter Row (0,1,2):
```

```
Rani's Turn! Enter Row (0,1,2): 1
Rani's Turn! Enter Column (0,1,2): 2

Current Board:
X | X |
---------
  | O | O
---------
  |   |
---------

Raju's Turn! Enter Row (0,1,2):
```

```
Raju's Turn! Enter Row (0,1,2): 0
Raju's Turn! Enter Column (0,1,2): 2

Current Board:
X | X | X
---------
  | O | O
---------
  |   |
---------
🎉 Congratulations Raju! You won the game! 🎉
```

```
Current Board:
  | X | X
---------
O |   |
---------
  |   |
---------
Rani's Turn! Enter Row (0,1,2): 0
Rani's Turn! Enter Column (0,1,2): 1
❌ Invalid move, try again.

Rani's Turn! Enter Row (0,1,2):
```

```
Current Board:
X | O | O
---------
O | X | X
---------
X | X | O
---------
🤝 Raju and Rani, it's a tie! 🤝
```