

## PRACTICAL NO-9

**Aim:-** Write a program to implement Tic-Tac-Toe game using python.

**Program:-**

# Tic-Tac-Toe Game Implementation

```
def print_board(board):
    for row in board:
        print(" | ".join(row))
        print("-" * 9)

def check_winner(board):
    # Check rows and columns
    for i in range(3):
        if board[i][0] == board[i][1] == board[i][2] != ' ':
            return board[i][0]
        if board[0][i] == board[1][i] == board[2][i] != ' ':
            return board[0][i]

    # Check diagonals
    if board[0][0] == board[1][1] == board[2][2] != ' ':
        return board[0][0]
    if board[0][2] == board[1][1] == board[2][0] != ' ':
        return board[0][2]

    return None

def is_full(board):
    return all(cell != ' ' for row in board for cell in row)

def play_game():
    board = [[' ' for _ in range(3)] for _ in range(3)]
    current_player = 'X'

    while True:
        print_board(board)
        print(f"Player {current_player}'s turn. Enter row and column (0-2) separated by a space:")
```

```
row, col = map(int, input().split())

if board[row][col] == ' ':
    board[row][col] = current_player
else:
    print("Cell already taken. Choose another cell.")
    continue

winner = check_winner(board)
if winner:
    print_board(board)
    print(f"Player {winner} wins!")
    break

if is_full(board):
    print_board(board)
    print("It's a draw!")
    break

current_player = 'O' if current_player == 'X' else 'X'

if __name__ == "__main__":
    play_game()
```

## Output:-

```
| | |
| | |
| | |
-----
Player X's turn. Enter row and column (0-2) separated by a space:
1 1
| | |
| X |
| | |
-----
Player O's turn. Enter row and column (0-2) separated by a space:
0 1
| O |
| X |
| | |
-----
```

## PRACTICAL NO-10

**Aim:-** Create a spell-checking application utilizing natural language processing (NLP) techniques, including syntactic and semantic analysis.

**Program:-**

```
import nltk
from textblob import TextBlob
from nltk.tokenize import word_tokenize
from nltk.corpus import words

# Initialize NLTK words list
nltk.download('words')
word_list = set(words.words())

def is_correctly_spelled(word):
    return word.lower() in word_list

def suggest_corrections(word):
    blob = TextBlob(word)
    return blob.correct()

def spell_check(text):
    tokens = word_tokenize(text)
    corrections = {}

    for word in tokens:
        if not is_correctly_spelled(word):
            suggestions = suggest_corrections(word)
            corrections[word] = suggestions

    return corrections

if __name__ == "__main__":
    user_input = input("Enter a sentence to check for spelling errors: ")
    corrections = spell_check(user_input)

    if corrections:
        for misspelled, suggestion in corrections.items():
            print(f"Misspelled word: '{misspelled}' -> Suggested correction: '{suggestion}'")
```

else:

```
print("No spelling errors found.")
```

## Output:-

Enter a sentence to check for spelling errors: Ths is a smple text with speling errors.

```
arduino Copy code  
  
Misspelled word: 'Ths' -> Suggested correction: 'This'  
Misspelled word: 'smple' -> Suggested correction: 'simple'  
Misspelled word: 'speling' -> Suggested correction: 'spelling'  
Misspelled word: 'errors' -> Suggested correction: 'errors'
```