```
|0   |1   |2   |3   |4   |5   |6   |7   |8
#!/usr/bin/python
# Tic Tac Toe
# Code Angel


import sys
import os
import pygame
from pygame.locals import *
import random

# Define the colours
X_COLOUR = (54, 169, 225)
O_COLOUR = (149, 193, 31)
TIE_COLOUR = (130, 163, 161)
BACK_COLOUR = (41, 35, 92)
GRID_COLOUR = (45, 46, 131)


# Define constants
SCREEN_WIDTH = 640
SCREEN_HEIGHT = 480
BOX_BLOCK_SIZE = 112
BOARD_TOP = 64
LINE_WIDTH = 16
WINNING_LINE_WIDTH = 8
SCOREBOARD_MARGIN = 4
SCOREBOARD_HEIGHT = 36

# Setup
os.environ['SDL_VIDEO_CENTERED'] = '1'
pygame.mixer.pre_init(44100, -16, 2, 512)
pygame.mixer.init()
pygame.init()
game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption('Tic Tac Toe')
clock = pygame.time.Clock()
score_font = pygame.font.SysFont('Helvetica', 24)
board_font = pygame.font.SysFont('Helvetica Bold', 128)

# Load sounds
|0   |1   |2   |3   |4   |5   |6   |7   |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
win_sound = pygame.mixer.Sound('win.ogg')


def main():

    # Initialise variables
    player_score = 0
    computer_score = 0
    ties = 0
    pieces = 0
    game_over = False
    result = ''

    board = [[], [], []]
    winning_line = {'has_won': False, 'line_start': [-1, -1], 'line_end': [-1, -1]}

    # Set up empty game board and toss coin
    reset_board(board)
    coin_toss = get_coin_toss()
    player_turn = get_player_turn(coin_toss)
    heads_tails_message = True

    # Main game loop
    while True:

        for event in pygame.event.get():

            # If game is not over
            if game_over is False:

                # If it is the player's turn - get the row and column of mouse click
                if player_turn is True:
                    if event.type == MOUSEBUTTONUP:
                        mouse_x, mouse_y = event.pos

                        row = get_row_clicked(mouse_y)
                        column = get_column_clicked(mouse_x)

                        # Mouse is clicked on the board
                        if row >= 0 and column >= 0:

|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
                            # Free space clicked
                            if board[row][column] == '-':
                                board[row][column] = 'X'

                                pieces += 1
                                check_winning_line(board, 'X', winning_line)

                                # Player wins
                                if winning_line.get('has_won') is True:
                                    player_score += 1
                                    game_over = True
                                    result = 'player win'
                                    win_sound.play()

                                # 9 pieces played - game tied
                                elif pieces == 9:
                                    ties += 1
                                    game_over = True
                                    result = 'tie'

                                # Now it is the computer's turn
                                else:
                                    player_turn = False

                        heads_tails_message = False

            # Game is over - wait for RETURN key to play again
            else:
                key_pressed = pygame.key.get_pressed()
                if key_pressed[pygame.K_RETURN]:

                    pieces = 0
                    game_over = False

                    reset_board(board)
                    coin_toss = get_coin_toss()
                    player_turn = get_player_turn(coin_toss)
                    heads_tails_message = True

            if event.type == QUIT:
                pygame.quit()
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
                sys.exit()

        # Computer turn
        if player_turn is False and game_over is False:

            calculate_computer_move(board)
            pieces += 1
            check_winning_line(board, 'O', winning_line)

            # Computer wins
            if winning_line.get('has_won') is True:
                computer_score += 1
                game_over = True
                result = 'computer win'
                win_sound.play()

            # 9 pieces played - game tied
            elif pieces == 9:
                winning_line['has_won'] = True
                ties += 1
                game_over = True
                result = 'tie'

            # Now it is the player's turn
            else:
                player_turn = True

        # Draw screen, board and pieces
        game_screen.fill(BACK_COLOUR)
        draw_board()
        draw_pieces(board)

        if game_over is True:
            draw_winning_line(winning_line)
            display_game_end_message(result)

        display_scores(player_score, computer_score, ties)

        if heads_tails_message is True:
            display_heads_tails_message(coin_toss)

|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
        pygame.display.update()
        clock.tick(60)


# Draw the board
def draw_board():
    grid_size = calculate_grid_size()
    board_left = calculate_board_left()
    first_vertical_line_x = board_left + BOX_BLOCK_SIZE
    second_vertical_line_x = board_left + BOX_BLOCK_SIZE + LINE_WIDTH + BOX_BLOCK_SIZE

    first_vertical_line_rect = pygame.Rect(first_vertical_line_x, BOARD_TOP, LINE_WIDTH, grid_size)
    pygame.draw.rect(game_screen, GRID_COLOUR, first_vertical_line_rect)

    second_vertical_line_rect = pygame.Rect(second_vertical_line_x, BOARD_TOP, LINE_WIDTH, grid_size)
    pygame.draw.rect(game_screen, GRID_COLOUR, second_vertical_line_rect)

    first_horizontal_line_y = BOARD_TOP + BOX_BLOCK_SIZE
    second_vertical_line_y = BOARD_TOP + BOX_BLOCK_SIZE + LINE_WIDTH + BOX_BLOCK_SIZE

    first_horizontal_line_rect = pygame.Rect(board_left, first_horizontal_line_y, grid_size, LINE_WIDTH)
    pygame.draw.rect(game_screen, GRID_COLOUR, first_horizontal_line_rect)

    second_horizontal_line_rect = pygame.Rect(board_left, second_vertical_line_y, grid_size, LINE_WIDTH)
    pygame.draw.rect(game_screen, GRID_COLOUR, second_horizontal_line_rect)


# Draw the pieces on the board
def draw_pieces(board):

    # Loop through all of the board spaces
    for row in range(3):
        for col in range(3):
            x_o = board[row][col]

            # If there is a piece in that location, draw it
            if x_o == 'X' or x_o == 'O':
                if x_o == 'X':
                    text = board_font.render(x_o, True, X_COLOUR)
                else:
                    text = board_font.render(x_o, True, O_COLOUR)
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8

                text_rect = text.get_rect()
                board_left = calculate_board_left()
                text_rect.centerx = board_left + BOX_BLOCK_SIZE * (col + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * col
                text_rect.centery = BOARD_TOP + BOX_BLOCK_SIZE * (row + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * row

                game_screen.blit(text, text_rect)


# Calculate grid size
def calculate_grid_size():
    return BOX_BLOCK_SIZE * 3 + LINE_WIDTH * 2


# Calculate board left
def calculate_board_left():
    grid_size = calculate_grid_size()
    return (SCREEN_WIDTH - grid_size) / 2


# Check column clicked based on the mouse x coordinate
def get_column_clicked(x):
    board_left = calculate_board_left()

    column = -1

    col_1_left = board_left
    col_1_right = col_1_left + BOX_BLOCK_SIZE
    col_2_left = col_1_right + LINE_WIDTH
    col_2_right = col_2_left + BOX_BLOCK_SIZE
    col_3_left = col_2_right + LINE_WIDTH
    col_3_right = col_3_left + BOX_BLOCK_SIZE

    # If the mouse x coordinate is in the left hand column
    if col_1_left < x < col_1_right:
        column = 0

    # If the mouse x coordinate is in the middle column
    elif col_2_left < x < col_2_right:
        column = 1

|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
    # If the mouse x coordinate is in the right hand column
    elif col_3_left < x < col_3_right:
        column = 2

    return column


# Check row clicked based on the mouse y coordinate
def get_row_clicked(y):

    row = -1

    row_1_top = BOARD_TOP
    row_1_bottom = row_1_top + BOX_BLOCK_SIZE
    row_2_top = row_1_bottom + LINE_WIDTH
    row_2_bottom = row_2_top + BOX_BLOCK_SIZE
    row_3_top = row_2_bottom + LINE_WIDTH
    row_3_bottom = row_3_top + BOX_BLOCK_SIZE

    # If the mouse y coordinate is in the top row
    if row_1_top < y < row_1_bottom:
        row = 0

    # If the mouse y coordinate is in the middle row
    elif row_2_top < y < row_2_bottom:
        row = 1

    # If the mouse y coordinate is in the bottom row
    elif row_3_top < y < row_3_bottom:
        row = 2

    return row


# Reset the board list
def reset_board(board):
    for row in range(3):
        board[row] = ['-', '-', '-']


# Computer Turn
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
def calculate_computer_move(board):
    block_move = False
    middle_free = False

    # Check winning positions
    win_move, win_row, win_col = check_row(board, 'O')

    if win_move is False:
        win_move, win_row, win_col = check_column(board, 'O')

        if win_move is False:
            win_move, win_row, win_col = check_diagonal_1(board, 'O')

            if win_move is False:
                win_move, win_row, win_col = check_diagonal_2(board, 'O')

    if win_move is True:
        board[win_row][win_col] = 'O'

    # If no winning positions, check blocking positions
    else:
        block_move, block_row, block_col = check_row(board, 'X')

        if block_move is False:
            block_move, block_row, block_col = check_column(board, 'X')

            if block_move is False:
                block_move, block_row, block_col = check_diagonal_1(board, 'X')

                if block_move is False:
                    block_move, block_row, block_col = check_diagonal_2(board, 'X')

        if block_move is True:
            board[block_row][block_col] = 'O'

    # If no winning positions or blocking positions, check if middle is free
    if win_move is False and block_move is False:
        middle_free = check_middle(board)

        if middle_free is True:
            board[1][1] = 'O'
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    # If no winning positions or blocking positions or middle, pick random free space
    if win_move is False and block_move is False and middle_free is False:
        random_row, random_column = get_random_space(board)
        board[random_row][random_column] = 'O'


# Two pieces in a row with one space available
def check_row(board, piece):

    play_row = -1
    play_col = -1
    make_move = False

    for row in range(3):
        if board[row] == [piece, piece, '-']:
            play_row = row
            play_col = 2
            make_move = True
        elif board[row] == [piece, '-', piece]:
            play_row = row
            play_col = 1
            make_move = True
        elif board[row] == ['-', piece, piece]:
            play_row = row
            play_col = 0
            make_move = True

    return make_move, play_row, play_col


# Two pieces in a column with one space available
def check_column(board, piece):

    play_row = -1
    play_col = -1
    space_row = -1
    make_move = False

    for col in range(3):
        space_count = 0
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
        piece_count = 0
        for row in range(3):
            if board[row][col] == piece:
                piece_count += 1
            elif board[row][col] == '-':
                space_count += 1
                space_row = row

        if piece_count == 2 and space_count == 1:
            play_row = space_row
            play_col = col
            make_move = True

    return make_move, play_row, play_col


# Two pieces in a diagonal top left to bottom right with one space available
def check_diagonal_1(board, piece):
    play_row = -1
    play_col = -1
    piece_count = 0
    space_count = 0
    space_row_col = -1
    make_move = False

    for row_col in range(3):
        if board[row_col][row_col] == piece:
            piece_count += 1
        elif board[row_col][row_col] == '-':
            space_count += 1
            space_row_col = row_col

    if piece_count == 2 and space_count == 1:
        play_row = space_row_col
        play_col = space_row_col
        make_move = True

    return make_move, play_row, play_col


# Two pieces in a diagonal bottom left to top right with one space available
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
def check_diagonal_2(board, piece):
    play_row = -1
    play_col = -1
    piece_count = 0
    space_count = 0
    space_col = -1
    make_move = False

    for col in range(3):
        if board[2 - col][col] == piece:
            piece_count += 1
        elif board[2 - col][col] == '-':
            space_count += 1
            space_col = col

    if piece_count == 2 and space_count == 1:
        play_col = space_col
        play_row = 2 - space_col
        make_move = True

    return make_move, play_row, play_col


# Check if middle is free
def check_middle(board):
    middle_free = False

    if board[1][1] == '-':
        middle_free = True

    return middle_free


# Get random free space
def get_random_space(board):
    rand_row = random.randint(0, 2)
    rand_col = random.randint(0, 2)

    while board[rand_row][rand_col] != '-':
        rand_row = random.randint(0, 2)
        rand_col = random.randint(0, 2)
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0   |1   |2   |3   |4   |5   |6   |7   |8

    return rand_row, rand_col


# Check winning lines
def check_winning_line(board, piece, winning_line):

    if board[0] == [piece, piece, piece]:
        winning_line['has_won'] = True
        winning_line['line_start'] = [0, 0]
        winning_line['line_end'] = [0, 2]

    elif board[1] == [piece, piece, piece]:
        winning_line['has_won'] = True
        winning_line['line_start'] = [1, 0]
        winning_line['line_end'] = [1, 2]

    elif board[2] == [piece, piece, piece]:
        winning_line['has_won'] = True
        winning_line['line_start'] = [2, 0]
        winning_line['line_end'] = [2, 2]

    elif board[0][0] == piece and board[1][0] == piece and board[2][0] == piece:
        winning_line['has_won'] = True
        winning_line['line_start'] = [0, 0]
        winning_line['line_end'] = [2, 0]

    elif board[0][1] == piece and board[1][1] == piece and board[2][1] == piece:
        winning_line['has_won'] = True
        winning_line['line_start'] = [0, 1]
        winning_line['line_end'] = [2, 1]

    elif board[0][2] == piece and board[1][2] == piece and board[2][2] == piece:
        winning_line['has_won'] = True
        winning_line['line_start'] = [0, 2]
        winning_line['line_end'] = [2, 2]

    elif board[0][0] == piece and board[1][1] == piece and board[2][2] == piece:
        winning_line['has_won'] = True
        winning_line['line_start'] = [0, 0]
        winning_line['line_end'] = [2, 2]
|0   |1   |2   |3   |4   |5   |6   |7   |8
```

```
    elif board[2][0] == piece and board[1][1] == piece and board[0][2] == piece:
        winning_line['has_won'] = True
        winning_line['line_start'] = [2, 0]
        winning_line['line_end'] = [0, 2]

    else:
        winning_line['has_won'] = False
        winning_line['line_start'] = [-1, -1]
        winning_line['line_end'] = [-1, -1]


# Draw winning line
def draw_winning_line(winning_line):
    board_left = calculate_board_left()

    start = winning_line.get('line_start')
    end = winning_line.get('line_end')

    start_x = board_left + BOX_BLOCK_SIZE * (start[1] + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * start[1]
    start_y = BOARD_TOP + BOX_BLOCK_SIZE * (start[0] + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * start[0]

    end_x = board_left + BOX_BLOCK_SIZE * (end[1] + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * end[1]
    end_y = BOARD_TOP + BOX_BLOCK_SIZE * (end[0] + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * end[0]

    pygame.draw.line(game_screen, TIE_COLOUR, (start_x, start_y), (end_x, end_y), WINNING_LINE_WIDTH)


# Display scores
def display_scores(player_score, computer_score, ties):

    # Draw rectangle
    scoreboard_background_rect = (0, 0, SCREEN_WIDTH, SCOREBOARD_HEIGHT)
    pygame.draw.rect(game_screen, GRID_COLOUR, scoreboard_background_rect)

    # Display player score
    player_text = 'Player: ' + str(player_score)
    text = score_font.render(player_text, True, X_COLOUR)
    game_screen.blit(text, [SCOREBOARD_MARGIN, SCOREBOARD_MARGIN])

    # Display computer score
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
    computer_text = 'Computer: ' + str(computer_score)
    text = score_font.render(computer_text, True, O_COLOUR)
    text_rect = text.get_rect()
    game_screen.blit(text, [SCREEN_WIDTH - text_rect.width - SCOREBOARD_MARGIN, SCOREBOARD_MARGIN])

    # Display ties
    tie_text = 'Ties: ' + str(ties)
    text = score_font.render(tie_text, True, TIE_COLOUR)
    text_rect = text.get_rect()
    game_screen.blit(text, [(SCREEN_WIDTH - text_rect.width) / 2, SCOREBOARD_MARGIN])


# Display result of heads or tails
def display_heads_tails_message(heads_tails):
    if heads_tails == 'heads':
        display_text = "It's heads - player goes first"
        text = score_font.render(display_text, True, X_COLOUR)
    else:
        display_text = "It's tails - computer goes first"
        text = score_font.render(display_text, True, O_COLOUR)

    text_rect = text.get_rect()
    x_loc = (SCREEN_WIDTH - text_rect.width) / 2
    y_loc = SCREEN_HEIGHT - SCOREBOARD_HEIGHT
    game_screen.blit(text, [x_loc, y_loc])


# Display end of game messages
def display_game_end_message(result):

    return_text = ' - press RETURN to continue'

    if result == 'player win':
        display_text = 'PLAYER wins' + return_text
        text = score_font.render(display_text, True, X_COLOUR)
    elif result == 'computer win':
        display_text = 'COMPUTER wins' + return_text
        text = score_font.render(display_text, True, O_COLOUR)
    else:
        display_text = 'Game tied' + return_text
        text = score_font.render(display_text, True, TIE_COLOUR)
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```python

    text_rect = text.get_rect()
    x_loc = (SCREEN_WIDTH - text_rect.width) / 2
    y_loc = SCREEN_HEIGHT - SCOREBOARD_MARGIN - SCOREBOARD_HEIGHT

    game_screen.blit(text, [x_loc, y_loc])


# Random coint toss - heads or tails
def get_coin_toss():
    coin_toss = random.choice(['heads', 'tails'])

    return coin_toss


# If heads, the player starts
def get_player_turn(coin_toss_result):
    if coin_toss_result == 'heads':
        player_turn = True
    else:
        player_turn = False

    return player_turn


if __name__ == '__main__':
    main()

```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
```