

```

|0 |1 |2 |3 |4 |5 |6 |7 |8
1  #!/usr/bin/python
2  # Tic Tac Toe
3  # Code Angel
4
5
6  import sys
7  import os
8  import pygame
9  from pygame.locals import *
10 import random
11
12 # Define the colours
13 X_COLOUR = (54, 169, 225)
14 O_COLOUR = (149, 193, 31)
15 TIE_COLOUR = (130, 163, 161)
16 BACK_COLOUR = (41, 35, 92)
17 GRID_COLOUR = (45, 46, 131)
18
19
20 # Define constants
21 SCREEN_WIDTH = 640
22 SCREEN_HEIGHT = 480
23 BOX_BLOCK_SIZE = 112
24 BOARD_TOP = 64
25 LINE_WIDTH = 16
26 WINNING_LINE_WIDTH = 8
27 SCOREBOARD_MARGIN = 4
28 SCOREBOARD_HEIGHT = 36
29
30 # Setup
31 os.environ['SDL_VIDEO_CENTERED'] = '1'
32 pygame.mixer.pre_init(44100, -16, 2, 512)
33 pygame.mixer.init()
34 pygame.init()
35 game_screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
36 pygame.display.set_caption('Tic Tac Toe')
37 clock = pygame.time.Clock()
38 score_font = pygame.font.SysFont('Helvetica', 24)
39 board_font = pygame.font.SysFont('Helvetica Bold', 128)
40
41 # Load sounds
|0 |1 |2 |3 |4 |5 |6 |7 |8

```

```

42 |0 |1 |2 |3 |4 |5 |6 |7 |8
win_sound = pygame.mixer.Sound('win.ogg')
43
44
45 def main():
46
47     # Initialise variables
48     player_score = 0
49     computer_score = 0
50     ties = 0
51     pieces = 0
52     game_over = False
53     result = ''
54
55     board = [[], [], []]
56     winning_line = {'has_won': False, 'line_start': [-1, -1], 'line_end': [-1, -1]}
57
58     # Set up empty game board and toss coin
59     reset_board(board)
60     coin_toss = get_coin_toss()
61     player_turn = get_player_turn(coin_toss)
62     heads_tails_message = True
63
64     # Main game loop
65     while True:
66
67         for event in pygame.event.get():
68
69             # If game is not over
70             if game_over is False:
71
72                 # If it is the player's turn - get the row and column of mouse click
73                 if player_turn is True:
74                     if event.type == MOUSEBUTTONDOWN:
75                         mouse_x, mouse_y = event.pos
76
77                         row = get_row_clicked(mouse_y)
78                         column = get_column_clicked(mouse_x)
79
80                         # Mouse is clicked on the board
81                         if row >= 0 and column >= 0:
82

```

```

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

```

```

83         |0  |1  |2  |3  |4  |5  |6  |7  |8
84         # Free space clicked
85         if board[row][column] == '-':
86             board[row][column] = 'X'
87
88         pieces += 1
89         check_winning_line(board, 'X', winning_line)
90
91         # Player wins
92         if winning_line.get('has_won') is True:
93             player_score += 1
94             game_over = True
95             result = 'player win'
96             win_sound.play()
97
98         # 9 pieces played - game tied
99         elif pieces == 9:
100             ties += 1
101             game_over = True
102             result = 'tie'
103
104         # Now it is the computer's turn
105         else:
106             player_turn = False
107
108         heads_tails_message = False
109
110     # Game is over - wait for RETURN key to play again
111     else:
112         key_pressed = pygame.key.get_pressed()
113         if key_pressed[pygame.K_RETURN]:
114
115             pieces = 0
116             game_over = False
117
118             reset_board(board)
119             coin_toss = get_coin_toss()
120             player_turn = get_player_turn(coin_toss)
121             heads_tails_message = True
122
123     if event.type == QUIT:
124         pygame.quit()

```

```

124         sys.exit()
125
126     # Computer turn
127     if player_turn is False and game_over is False:
128
129         calculate_computer_move(board)
130         pieces += 1
131         check_winning_line(board, 'O', winning_line)
132
133     # Computer wins
134     if winning_line.get('has_won') is True:
135         computer_score += 1
136         game_over = True
137         result = 'computer win'
138         win_sound.play()
139
140     # 9 pieces played - game tied
141     elif pieces == 9:
142         winning_line['has_won'] = True
143         ties += 1
144         game_over = True
145         result = 'tie'
146
147     # Now it is the player's turn
148     else:
149         player_turn = True
150
151     # Draw screen, board and pieces
152     game_screen.fill(BACK_COLOUR)
153     draw_board()
154     draw_pieces(board)
155
156     if game_over is True:
157         draw_winning_line(winning_line)
158         display_game_end_message(result)
159
160     display_scores(player_score, computer_score, ties)
161
162     if heads_tails_message is True:
163         display_heads_tails_message(coin_toss)
164

```

```

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

```

```

165         |0  |1  |2  |3  |4  |5  |6  |7  |8
166         pygame.display.update()
167         clock.tick(60)
168
169     # Draw the board
170     def draw_board():
171         grid_size = calculate_grid_size()
172         board_left = calculate_board_left()
173         first_vertical_line_x = board_left + BOX_BLOCK_SIZE
174         second_vertical_line_x = board_left + BOX_BLOCK_SIZE + LINE_WIDTH + BOX_BLOCK_SIZE
175
176         first_vertical_line_rect = pygame.Rect(first_vertical_line_x, BOARD_TOP, LINE_WIDTH, grid_size)
177         pygame.draw.rect(game_screen, GRID_COLOUR, first_vertical_line_rect)
178
179         second_vertical_line_rect = pygame.Rect(second_vertical_line_x, BOARD_TOP, LINE_WIDTH, grid_size)
180         pygame.draw.rect(game_screen, GRID_COLOUR, second_vertical_line_rect)
181
182         first_horizontal_line_y = BOARD_TOP + BOX_BLOCK_SIZE
183         second_vertical_line_y = BOARD_TOP + BOX_BLOCK_SIZE + LINE_WIDTH + BOX_BLOCK_SIZE
184
185         first_horizontal_line_rect = pygame.Rect(board_left, first_horizontal_line_y, grid_size, LINE_WIDTH)
186         pygame.draw.rect(game_screen, GRID_COLOUR, first_horizontal_line_rect)
187
188         second_horizontal_line_rect = pygame.Rect(board_left, second_vertical_line_y, grid_size, LINE_WIDTH)
189         pygame.draw.rect(game_screen, GRID_COLOUR, second_horizontal_line_rect)
190
191
192     # Draw the pieces on the board
193     def draw_pieces(board):
194
195         # Loop through all of the board spaces
196         for row in range(3):
197             for col in range(3):
198                 x_o = board[row][col]
199
200                 # If there is a piece in that location, draw it
201                 if x_o == 'X' or x_o == 'O':
202                     if x_o == 'X':
203                         text = board_font.render(x_o, True, X_COLOUR)
204                     else:
205                         text = board_font.render(x_o, True, O_COLOUR)

```

```

|0  |1  |2  |3  |4  |5  |6  |7  |8
206
207         text_rect = text.get_rect()
208         board_left = calculate_board_left()
209         text_rect.centerx = board_left + BOX_BLOCK_SIZE * (col + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * col
210         text_rect.centery = BOARD_TOP + BOX_BLOCK_SIZE * (row + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * row
211
212         game_screen.blit(text, text_rect)
213
214
215     # Calculate grid size
216     def calculate_grid_size():
217         return BOX_BLOCK_SIZE * 3 + LINE_WIDTH * 2
218
219
220     # Calculate board left
221     def calculate_board_left():
222         grid_size = calculate_grid_size()
223         return (SCREEN_WIDTH - grid_size) / 2
224
225
226     # Check column clicked based on the mouse x coordinate
227     def get_column_clicked(x):
228         board_left = calculate_board_left()
229
230         column = -1
231
232         col_1_left = board_left
233         col_1_right = col_1_left + BOX_BLOCK_SIZE
234         col_2_left = col_1_right + LINE_WIDTH
235         col_2_right = col_2_left + BOX_BLOCK_SIZE
236         col_3_left = col_2_right + LINE_WIDTH
237         col_3_right = col_3_left + BOX_BLOCK_SIZE
238
239         # If the mouse x coordinate is in the left hand column
240         if col_1_left < x < col_1_right:
241             column = 0
242
243         # If the mouse x coordinate is in the middle column
244         elif col_2_left < x < col_2_right:
245             column = 1
246
|0  |1  |2  |3  |4  |5  |6  |7  |8

```

```

247 |0 |1 |2 |3 |4 |5 |6 |7 |8
248 # If the mouse x coordinate is in the right hand column
249 elif col_3_left < x < col_3_right:
250     column = 2
251
252     return column
253
254 # Check row clicked based on the mouse y coordinate
255 def get_row_clicked(y):
256
257     row = -1
258
259     row_1_top = BOARD_TOP
260     row_1_bottom = row_1_top + BOX_BLOCK_SIZE
261     row_2_top = row_1_bottom + LINE_WIDTH
262     row_2_bottom = row_2_top + BOX_BLOCK_SIZE
263     row_3_top = row_2_bottom + LINE_WIDTH
264     row_3_bottom = row_3_top + BOX_BLOCK_SIZE
265
266     # If the mouse y coordinate is in the top row
267     if row_1_top < y < row_1_bottom:
268         row = 0
269
270     # If the mouse y coordinate is in the middle row
271     elif row_2_top < y < row_2_bottom:
272         row = 1
273
274     # If the mouse y coordinate is in the bottom row
275     elif row_3_top < y < row_3_bottom:
276         row = 2
277
278     return row
279
280
281 # Reset the board list
282 def reset_board(board):
283     for row in range(3):
284         board[row] = ['- ', '- ', '- ']
285
286
287 # Computer Turn
288 |0 |1 |2 |3 |4 |5 |6 |7 |8

```

```

288 |0 |1 |2 |3 |4 |5 |6 |7 |8
288 def calculate_computer_move(board):
289     block_move = False
290     middle_free = False
291
292     # Check winning positions
293     win_move, win_row, win_col = check_row(board, 'O')
294
295     if win_move is False:
296         win_move, win_row, win_col = check_column(board, 'O')
297
298         if win_move is False:
299             win_move, win_row, win_col = check_diagonal_1(board, 'O')
300
301             if win_move is False:
302                 win_move, win_row, win_col = check_diagonal_2(board, 'O')
303
304     if win_move is True:
305         board[win_row][win_col] = 'O'
306
307     # If no winning positions, check blocking positions
308     else:
309         block_move, block_row, block_col = check_row(board, 'X')
310
311         if block_move is False:
312             block_move, block_row, block_col = check_column(board, 'X')
313
314             if block_move is False:
315                 block_move, block_row, block_col = check_diagonal_1(board, 'X')
316
317                 if block_move is False:
318                     block_move, block_row, block_col = check_diagonal_2(board, 'X')
319
320         if block_move is True:
321             board[block_row][block_col] = 'O'
322
323     # If no winning positions or blocking positions, check if middle is free
324     if win_move is False and block_move is False:
325         middle_free = check_middle(board)
326
327         if middle_free is True:
328             board[1][1] = 'O'

```



```

|0  |1  |2  |3  |4  |5  |6  |7  |8
329
330     # If no winning positions or blocking positions or middle, pick random free space
331     if win_move is False and block_move is False and middle_free is False:
332         random_row, random_column = get_random_space(board)
333         board[random_row][random_column] = 'O'
334
335
336     # Two pieces in a row with one space available
337     def check_row(board, piece):
338
339         play_row = -1
340         play_col = -1
341         make_move = False
342
343         for row in range(3):
344             if board[row] == [piece, piece, '-']:
345                 play_row = row
346                 play_col = 2
347                 make_move = True
348             elif board[row] == [piece, '-', piece]:
349                 play_row = row
350                 play_col = 1
351                 make_move = True
352             elif board[row] == ['- ', piece, piece]:
353                 play_row = row
354                 play_col = 0
355                 make_move = True
356
357         return make_move, play_row, play_col
358
359
360     # Two pieces in a column with one space available
361     def check_column(board, piece):
362
363         play_row = -1
364         play_col = -1
365         space_row = -1
366         make_move = False
367
368         for col in range(3):
369             space_count = 0

```

```

|0  |1  |2  |3  |4  |5  |6  |7  |8
370     piece_count = 0
371     for row in range(3):
372         if board[row][col] == piece:
373             piece_count += 1
374         elif board[row][col] == '-':
375             space_count += 1
376             space_row = row
377
378     if piece_count == 2 and space_count == 1:
379         play_row = space_row
380         play_col = col
381         make_move = True
382
383     return make_move, play_row, play_col
384
385
386 # Two pieces in a diagonal top left to bottom right with one space available
387 def check_diagonal_1(board, piece):
388     play_row = -1
389     play_col = -1
390     piece_count = 0
391     space_count = 0
392     space_row_col = -1
393     make_move = False
394
395     for row_col in range(3):
396         if board[row_col][row_col] == piece:
397             piece_count += 1
398         elif board[row_col][row_col] == '-':
399             space_count += 1
400             space_row_col = row_col
401
402     if piece_count == 2 and space_count == 1:
403         play_row = space_row_col
404         play_col = space_row_col
405         make_move = True
406
407     return make_move, play_row, play_col
408
409
410 # 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
411 def check_diagonal_2(board, piece):
412     play_row = -1
413     play_col = -1
414     piece_count = 0
415     space_count = 0
416     space_col = -1
417     make_move = False
418
419     for col in range(3):
420         if board[2 - col][col] == piece:
421             piece_count += 1
422         elif board[2 - col][col] == '-':
423             space_count += 1
424             space_col = col
425
426     if piece_count == 2 and space_count == 1:
427         play_col = space_col
428         play_row = 2 - space_col
429         make_move = True
430
431     return make_move, play_row, play_col
432
433
434 # Check if middle is free
435 def check_middle(board):
436     middle_free = False
437
438     if board[1][1] == '-':
439         middle_free = True
440
441     return middle_free
442
443
444 # Get random free space
445 def get_random_space(board):
446     rand_row = random.randint(0, 2)
447     rand_col = random.randint(0, 2)
448
449     while board[rand_row][rand_col] != '-':
450         rand_row = random.randint(0, 2)
451         rand_col = random.randint(0, 2)
|0 |1 |2 |3 |4 |5 |6 |7 |8

```

```

452 |0 |1 |2 |3 |4 |5 |6 |7 |8
453     return rand_row, rand_col
454
455
456 # Check winning lines
457 def check_winning_line(board, piece, winning_line):
458
459     if board[0] == [piece, piece, piece]:
460         winning_line['has_won'] = True
461         winning_line['line_start'] = [0, 0]
462         winning_line['line_end'] = [0, 2]
463
464     elif board[1] == [piece, piece, piece]:
465         winning_line['has_won'] = True
466         winning_line['line_start'] = [1, 0]
467         winning_line['line_end'] = [1, 2]
468
469     elif board[2] == [piece, piece, piece]:
470         winning_line['has_won'] = True
471         winning_line['line_start'] = [2, 0]
472         winning_line['line_end'] = [2, 2]
473
474     elif board[0][0] == piece and board[1][0] == piece and board[2][0] == piece:
475         winning_line['has_won'] = True
476         winning_line['line_start'] = [0, 0]
477         winning_line['line_end'] = [2, 0]
478
479     elif board[0][1] == piece and board[1][1] == piece and board[2][1] == piece:
480         winning_line['has_won'] = True
481         winning_line['line_start'] = [0, 1]
482         winning_line['line_end'] = [2, 1]
483
484     elif board[0][2] == piece and board[1][2] == piece and board[2][2] == piece:
485         winning_line['has_won'] = True
486         winning_line['line_start'] = [0, 2]
487         winning_line['line_end'] = [2, 2]
488
489     elif board[0][0] == piece and board[1][1] == piece and board[2][2] == piece:
490         winning_line['has_won'] = True
491         winning_line['line_start'] = [0, 0]
492         winning_line['line_end'] = [2, 2]

```

```

|0 |1 |2 |3 |4 |5 |6 |7 |8
493
494     elif board[2][0] == piece and board[1][1] == piece and board[0][2] == piece:
495         winning_line['has_won'] = True
496         winning_line['line_start'] = [2, 0]
497         winning_line['line_end'] = [0, 2]
498
499     else:
500         winning_line['has_won'] = False
501         winning_line['line_start'] = [-1, -1]
502         winning_line['line_end'] = [-1, -1]
503
504
505 # Draw winning line
506 def draw_winning_line(winning_line):
507     board_left = calculate_board_left()
508
509     start = winning_line.get('line_start')
510     end = winning_line.get('line_end')
511
512     start_x = board_left + BOX_BLOCK_SIZE * (start[1] + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * start[1]
513     start_y = BOARD_TOP + BOX_BLOCK_SIZE * (start[0] + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * start[0]
514
515     end_x = board_left + BOX_BLOCK_SIZE * (end[1] + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * end[1]
516     end_y = BOARD_TOP + BOX_BLOCK_SIZE * (end[0] + 1) - BOX_BLOCK_SIZE / 2 + LINE_WIDTH * end[0]
517
518     pygame.draw.line(game_screen, TIE_COLOUR, (start_x, start_y), (end_x, end_y), WINNING_LINE_WIDTH)
519
520
521 # Display scores
522 def display_scores(player_score, computer_score, ties):
523
524     # Draw rectangle
525     scoreboard_background_rect = (0, 0, SCREEN_WIDTH, SCOREBOARD_HEIGHT)
526     pygame.draw.rect(game_screen, GRID_COLOUR, scoreboard_background_rect)
527
528     # Display player score
529     player_text = 'Player: ' + str(player_score)
530     text = score_font.render(player_text, True, X_COLOUR)
531     game_screen.blit(text, [SCOREBOARD_MARGIN, SCOREBOARD_MARGIN])
532
533     # Display computer score
|0 |1 |2 |3 |4 |5 |6 |7 |8

```

```

534 computer_text = 'Computer: ' + str(computer_score)
535 text = score_font.render(computer_text, True, O_COLOUR)
536 text_rect = text.get_rect()
537 game_screen.blit(text, [SCREEN_WIDTH - text_rect.width - SCOREBOARD_MARGIN, SCOREBOARD_MARGIN])
538
539 # Display ties
540 tie_text = 'Ties: ' + str(ties)
541 text = score_font.render(tie_text, True, TIE_COLOUR)
542 text_rect = text.get_rect()
543 game_screen.blit(text, [(SCREEN_WIDTH - text_rect.width) / 2, SCOREBOARD_MARGIN])
544
545
546 # Display result of heads or tails
547 def display_heads_tails_message(heads_tails):
548     if heads_tails == 'heads':
549         display_text = "It's heads - player goes first"
550         text = score_font.render(display_text, True, X_COLOUR)
551     else:
552         display_text = "It's tails - computer goes first"
553         text = score_font.render(display_text, True, O_COLOUR)
554
555     text_rect = text.get_rect()
556     x_loc = (SCREEN_WIDTH - text_rect.width) / 2
557     y_loc = SCREEN_HEIGHT - SCOREBOARD_HEIGHT
558     game_screen.blit(text, [x_loc, y_loc])
559
560
561 # Display end of game messages
562 def display_game_end_message(result):
563
564     return_text = ' - press RETURN to continue'
565
566     if result == 'player win':
567         display_text = 'PLAYER wins' + return_text
568         text = score_font.render(display_text, True, X_COLOUR)
569     elif result == 'computer win':
570         display_text = 'COMPUTER wins' + return_text
571         text = score_font.render(display_text, True, O_COLOUR)
572     else:
573         display_text = 'Game tied' + return_text
574         text = score_font.render(display_text, True, TIE_COLOUR)

```

```

575 |0 |1 |2 |3 |4 |5 |6 |7 |8
576     text_rect = text.get_rect()
577     x_loc = (SCREEN_WIDTH - text_rect.width) / 2
578     y_loc = SCREEN_HEIGHT - SCOREBOARD_MARGIN - SCOREBOARD_HEIGHT
579
580     game_screen.blit(text, [x_loc, y_loc])
581
582
583     # Random coin toss - heads or tails
584     def get_coin_toss():
585         coin_toss = random.choice(['heads', 'tails'])
586
587         return coin_toss
588
589
590     # If heads, the player starts
591     def get_player_turn(coin_toss_result):
592         if coin_toss_result == 'heads':
593             player_turn = True
594         else:
595             player_turn = False
596
597         return player_turn
598
599
600     if __name__ == '__main__':
601         main()
602

```

```

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

```