Girls' Programming Network

Tic Tac Toe!

Tutors Only

This project was created by GPN Australia for GPN sites all around Australia!

This workbook and related materials were created by tutors at:

Sydney, Canberra and Perth

Girls' Programming Network

If you see any of the following tutors don't forget to thank them!!

Writers

Amanda Hogan
Isabella Hogan
Renee Noble

A massive thanks to our sponsors for supporting us!

ATLASSIAN

Workbook 1

Part 1: Welcome to Tic Tac Toe!

1.4: Printing the Board

```
# Copy your previous code here...
print("Welcome to Tic-Tac-Toe!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
print("-------------")
print("|", board[0], "|", board[1], "|", board[2], "|")
print("-------------")
print("|", board[3], "|", board[4], "|", board[5], "|")
print("-------------")
print("|", board[6], "|", board[7], "|", board[8], "|")
print("-------------")
```

Part 2: Enter the First Move

2.3: Check what happened!

```
# Copy your previous code here...
print("Welcome to Tic-Tac-Toe!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
print("-------------")
print("|", board[0], "|", board[1], "|", board[2], "|")
print("-------------")
print("|", board[3], "|", board[4], "|", board[5], "|")
print("-------------")
print("|", board[6], "|", board[7], "|", board[8], "|")
print("-------------")

symbol = "0"
square = input("Which square do you want your symbol to go in? ")
square _index = int(square)
board[square_index] = symbol
```

Bonus 2.5: Welcome the players

```
# Copy your previous code here...
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")
player_X = input("Who is playing crosses? ")

print("Welcome", player_O, ", your symbol is O!")
print("Welcome", player_X, ", your symbol is X!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
print("-------------")
print("|", board[0], "|", board[1], "|", board[2], "|")
print("-------------")
print("|", board[3], "|", board[4], "|", board[5], "|")
print("-------------")
print("|", board[6], "|", board[7], "|", board[8], "|")
print("-------------")

symbol = "0"
square = input("Which square do you want your symbol to go in? ")
square _index = int(square)
board[square_index] = symbol
```

Part 3: Creating a print function

3.4: Let's print the board again

```
# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")
player_X = input("Who is playing crosses? ")

print("Welcome", player_O, ", your symbol is O!")
print("Welcome", player_X, ", your symbol is X!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
print_board(board)

symbol = "0"
square = input("Which square do you want your symbol to go in? ")
square _index = int(square)
board[square_index] = symbol

print_board(board)
```

Part 4 : Taking Turns

4.3 Run your code!

```
# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")
player_X = input("Who is playing crosses? ")

print("Welcome", player_O, ", your symbol is O!")
print("Welcome", player_X, ", your symbol is X!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
print_board(board)

symbol = "0"
print("The current player is", symbol, "!")
square = input("Which square do you want your symbol to go in? ")
square _index = int(square)
board[square_index] = symbol

print_board(board)
if symbol == "0":
    symbol = "X"
else:
    symbol = "0"
```

Part 5 : Wait a while to win?

5.2 Did I win yet?

```
# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")
player_X = input("Who is playing crosses? ")

print("Welcome", player_O, ", your symbol is O!")
print("Welcome", player_X, ", your symbol is X!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
print_board(board)

symbol = "0"
while not game_over:
    print("The current player is", symbol, "!")
    square = input("Which square do you want your symbol to go in? ")
    square _index = int(square)
    board[square_index] = symbol

    print_board(board)
    if symbol == "0":
        symbol = "X"
    else:
        symbol = "0"
```

Part 6 : Winner winner tic tac dinner

6.2 Functions again

```
# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
def check_winner:

print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")
player_X = input("Who is playing crosses? ")

print("Welcome", player_O, ", your symbol is O!")
print("Welcome", player_X, ", your symbol is X!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
print_board(board)

symbol = "0"
while not game_over:
    print("The current player is", symbol, "!")
    square = input("Which square do you want your symbol to go in? ")
    square _index = int(square)
    board[square_index] = symbol

    print_board(board)
    if symbol == "0":
        symbol = "X"
    else:
        symbol = "0"
```

Part 7.1 : Option 1

7.1.4 No winners here!

Option 1: If statements

```
def check winner(board) :
    if board[0] == board[1] == board[2] != " ":
        return True
    elif board[3] == board[4] == board[5] != " ":
        return True
    elif board[6] == board[7] == board[8] != " ":
        return True
    if board[0] == board[3] == board[6] != " ":
        return True
    elif board[1] == board[4] == board[7] != " ":
        return True
    elif board[2] == board[5] == board[8] != " ":
        return True
    if board[0] == board[4] == board[8] != " ":
        return True
    elif board[2] == board[4] == board[6] != " ":
        return True
    else:
        return False
```

Part 7.2 : Option 2

7.1.4 No winners here!

Option 2: For loop and lists

```
def check winner(board) :
    winning combos = [
        # Rows
        (0,1,2),
        (3,4,5),
        (6,7,8),
        # Columns
        (0,3,6),
        (1,4,7),
        (2,5,8),
        # Diagonals
        (0,4,8),
        (2,4,6)
    ]

    for combo in winning_combos:
        combo_part_0 = combo[0]
        combo_part_1 = combo[1]
        combo_part_2 = combo[2]
        symbol_0 = board[combo_part_0]
        symbol_1 = board[combo_part_1]
        symbol_2 = board[combo_part_2]
        if symbol_0 == symbol_1 == symbol_2 == " ":
        return True
    return False
```

Part 8 : Declare the winner

8.2 Declare who won

```
# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
# Be aware that students may have used the Option 2 code here
def check_winner(board) :
    if board[0] == board[1] == board[2] != " ":
        return True
    elif board[3] == board[4] == board[5] != " ":
        return True
    elif board[6] == board[7] == board[8] != " ":
        return True
    if board[0] == board[3] == board[6] != " ":
        return True
    elif board[1] == board[4] == board[7] != " ":
        return True
    elif board[2] == board[5] == board[8] != " ":
        return True
    if board[0] == board[4] == board[8] != " ":
        return True
    elif board[2] == board[4] == board[6] != " ":
        return True
    else:
        return False


print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")
player_X = input("Who is playing crosses? ")

print("Welcome", player_O, ", your symbol is O!")
print("Welcome", player_X, ", your symbol is X!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
print_board(board)

symbol = "0"
while not game_over:
    print("The current player is", symbol, "!")
    square = input("Which square do you want your symbol to go in? ")
```

```
    square_index = int(square)
    board[square_index] = symbol

    print_board(board)
    game_over = check_winner(board)
    if game_over:
        print(symbol, "won! Congratulations!")
    if symbol == "0":
        symbol = "X"
    else:
        symbol = "0"
```

Extensions

All extensions commented with which

```
import random

# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
# Be aware that students may have used the Option 2 code here
def check_winner(board) :
    if board[0] == board[1] == board[2] != " ":
        return True
    elif board[3] == board[4] == board[5] != " ":
        return True
    elif board[6] == board[7] == board[8] != " ":
        return True
    if board[0] == board[3] == board[6] != " ":
        return True
    elif board[1] == board[4] == board[7] != " ":
        return True
    elif board[2] == board[5] == board[8] != " ":
        return True
    if board[0] == board[4] == board[8] != " ":
        return True
    elif board[2] == board[4] == board[6] != " ":
        return True
    else:
        return False


print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")
player_X = input("Who is playing crosses? ")

print("Welcome", player_O, ", your symbol is O!")
print("Welcome", player_X, ", your symbol is X!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
print_board(board)

# Extension 9
symbol = random.choice("X","O")
```

```
# Extension 12
if symbol == "O":
    current_player = player_O
else:
    current_player = player_X
print(symbol,"player will go first!")
free_squares = [0,1,2,3,4,5,6,7,8]
counter = 0
while not game_over:
    print("The current player is", current_player, "Who is playing as", symbol,"!")
#Extension 12

    # Extension 13
    if current_player == "computer":
        square = random.choice(free_squares)
    else:
        square = input("Which square do you want your symbol to go in? ")
    square_index = int(square)

    #  Extension 9
    if square_index not in free_squares:
        print("That wasn't a valid move!")
        continue
    board[square_index] = symbol
    counter+=1

    print_board(board)
    free_squares.remove(square_index)
    game_over = check_winner(board)
    if game_over:
        print(current_player, "won! Congratulations!")
    elif counter == 9: # Extension 10
        print("It's a tie!")
        break


    if symbol == "0":
        current_player = player_X
        symbol = "X"
    else:
        current_player = player_O
        symbol = "0"
```

Workbook 2

Part 1: Adding a basic computer player

1.4: Let the computer choose

- At the top of the file, make sure the student is importing the random function!

```
import random
```

- And then in the game loop, the **`square_index`** will be set like this:

```
square = input("Which square do you want to choose? ")
square_index = int(square)
if symbol == human_symbol:
    square = input("Which square do you want to choose? ")
    square_index = int(square)
else:
    square_index = random.choice(free_squares)
```

BONUS 1.5: Stop the silly humans

- Students will just need to add an if statement like this, in bold:

```
if symbol == human_symbol:
    square = input("Which square do you want to choose? ")
    square_index = int(square)

    if square_index not in free_squares:
        print("You can't place a symbol on that tile, it's already taken!")
        continue
else:
    square_index = random.choice(free_squares)
```

Full code for lesson 1

```
# Start your code here# Copy your previous code here...
import random

# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
# Be aware that students may have used the Option 2 code here
def check_winner(board) :
    if board[0] == board[1] == board[2] != " ":
        return True
    elif board[3] == board[4] == board[5] != " ":
        return True
    elif board[6] == board[7] == board[8] != " ":
        return True
    if board[0] == board[3] == board[6] != " ":
        return True
    elif board[1] == board[4] == board[7] != " ":
        return True
    elif board[2] == board[5] == board[8] != " ":
        return True
    if board[0] == board[4] == board[8] != " ":
        return True
    elif board[2] == board[4] == board[6] != " ":
        return True
    else:
        return False


print("Welcome to Tic-Tac-Toe!")
comp_symbol = "X"
human_symbol = "0"
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")

print("Welcome", player_O, ", your symbol is 0!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
print_board(board)

# Extension 9
symbol = random.choice(["X","0"])
# Extension 12
if symbol == "0":
```

```
    current_player = player_O
else:
    current_player = player_X
print(symbol,"player will go first!")
free_squares = [0,1,2,3,4,5,6,7,8]
counter = 0
while not game_over:
    print("The current player is", current_player, "Who is playing as", symbol,"!")
#Extension 12

    # Extension 13
    if current_player == "computer":
        square = random.choice(free_squares)
    else:
        square = input("Which square do you want your symbol to go in? ")
    square_index = int(square)

    #  Extension 9
    if square_index not in free_squares:
        print("That wasn't a valid move!")
        continue
    board[square_index] = symbol
    counter+=1

    print_board(board)
    free_squares = []
    for i, square in enumerate(board):
        if square == " ":
            free_squares.append(i)
    game_over = check_winner(board)
    if game_over:
        print(current_player, "won! Congratulations!")
    elif counter == 9: # Extension 10
        print("It's a tie!")
        break


    if symbol == "0":
        current_player = player_X
        symbol = "X"
    else:
        current_player = player_O
        symbol = "0"
```

Part 2: Making It Modular

2.3 Switching turns

- Make sure the student has removed the **`free_squares`** list from the game loop. The new function should look like this:

```
def get_free_squares(board):
    free_squares = []
    for index, symbol in enumerate(board):
        if symbol == " ":
            free_squares.append(index)
    return free_squares
```

- Note that if the student completed Bonus 1.5, they will to update that bit of code to call the new function, like this:

```
if square_index not in get_free_squares(board):
    print("You can't place a symbol on that tile!")
    continue
```

The new function should look like this:

```
def get_comp_move(board):
    free_squares = get_free_squares(board)
    return random.choice(free_squares)
```

- And then in the game loop, the **`get_comp_move`** will be called like this:

```
if symbol == human_symbol:
    square = input("Which square do you want to choose? ")
    square_index = int(square)
else:
    square_index = get_comp_move(board)
```

- The new function should look like this:

```
def get_other_symbol(symbol):
    if symbol == comp_symbol:
        return human_symbol
    else:
        return comp_symbol
```

- And then in the game loop, the function will be called like this:

```
elif counter == 9:
    print("Game over! It's a tie!")
```

```
    break

symbol = get_other_symbol(symbol)
```

Full code Lesson 2

```
# Start your code here# Copy your previous code here...
import random

# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
# Be aware that students may have used the Option 2 code here
def check_winner(board) :
    if board[0] == board[1] == board[2] != " ":
        return True
    elif board[3] == board[4] == board[5] != " ":
        return True
    elif board[6] == board[7] == board[8] != " ":
        return True
    if board[0] == board[3] == board[6] != " ":
        return True
    elif board[1] == board[4] == board[7] != " ":
        return True
    elif board[2] == board[5] == board[8] != " ":
        return True
    if board[0] == board[4] == board[8] != " ":
        return True
    elif board[2] == board[4] == board[6] != " ":
        return True
    else:
        return False

def get_free_squares(board):
    free_squares = []
    for index, symbol in enumerate(board):
        if symbol == " ":
            free_squares.append(index)
    return free_squares

def get_comp_move(board):
    free_squares = get_free_squares(board)
    return random.choice(free_squares)
```

```
def get_opposite_symbol(symbol):
    if symbol == "0":
        return "X"
    else:
        return "0"

print("Welcome to Tic-Tac-Toe!")
comp_symbol = "X"
human_symbol = "0"
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")

print("Welcome", player_O, ", your symbol is 0!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
print_board(board)

# Extension 9
symbol = random.choice(["X","0"])
# Extension 12
if symbol == "0":
    current_player = player_O
else:
    current_player = player_X
print(symbol,"player will go first!")
free_squares = [0,1,2,3,4,5,6,7,8]
counter = 0
while not game_over:
    print("The current player is", current_player, "Who is playing as", symbol,"!")
#Extension 12

    # Extension 13
    if current_player == "computer":
        square = get_comp_move(board)
    else:
        square = input("Which square do you want your symbol to go in? ")
    square_index = int(square)

    #  Extension 9
    if square_index not in free_squares:
        print("That wasn't a valid move!")
        continue
    board[square_index] = symbol
    counter+=1

    print_board(board)
    free_squares = get_free_squares(board)
    game_over = check_winner(board)
    if game_over:
        print(current_player, "won! Congratulations!")
```

```
    elif counter == 9: # Extension 10
        print("It's a tie!")
        break

    symbol = get_opposite_symbol(symbol)
```

Part 3: Winning on the next turn

3.1 Play to win

- The **`get_comp_move`** function should now look like this:

```
def get_comp_move(board):
    free_squares = get_free_squares(board)

    winning_moves = []
    for square in free_squares:
        board[square] = comp_symbol
        if check_winner(board):
            winning_moves.append(square)
        board[square] = " "

    if len(winning_moves) > 0:
        return winning_moves[0]
    else:
        return random.choice(free_squares)
```

- Note, the student could also write `len(winning_moves) >= 1 or len(winning_moves) != 0`.

Full Code Lesson 3

```
# Start your code here# Copy your previous code here...
import random

# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
```

```
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
# Be aware that students may have used the Option 2 code here
def check_winner(board) :
    if board[0] == board[1] == board[2] != " ":
        return True
    elif board[3] == board[4] == board[5] != " ":
        return True
    elif board[6] == board[7] == board[8] != " ":
        return True
    if board[0] == board[3] == board[6] != " ":
        return True
    elif board[1] == board[4] == board[7] != " ":
        return True
    elif board[2] == board[5] == board[8] != " ":
        return True
    if board[0] == board[4] == board[8] != " ":
        return True
    elif board[2] == board[4] == board[6] != " ":
        return True
    else:
        return False

def get_free_squares(board):
    free_squares = []
    for index, symbol in enumerate(board):
        if symbol == " ":
            free_squares.append(index)
    return free_squares

def get_comp_move(board):
    winning_moves = []
    free_squares = get_free_squares(board)
    for square in free_squares:
        board[square] = comp_symbol
        if check_winner(board):
            winning_moves.append(square)
        board[square] = " "
    if len(winning_moves) > 0:
        return winning_moves[0]
    return random.choice(free_squares)


def get_opposite_symbol(symbol):
    if symbol == "0":
        return "X"
    else:
        return "0"

print("Welcome to Tic-Tac-Toe!")
comp_symbol = "X"
```

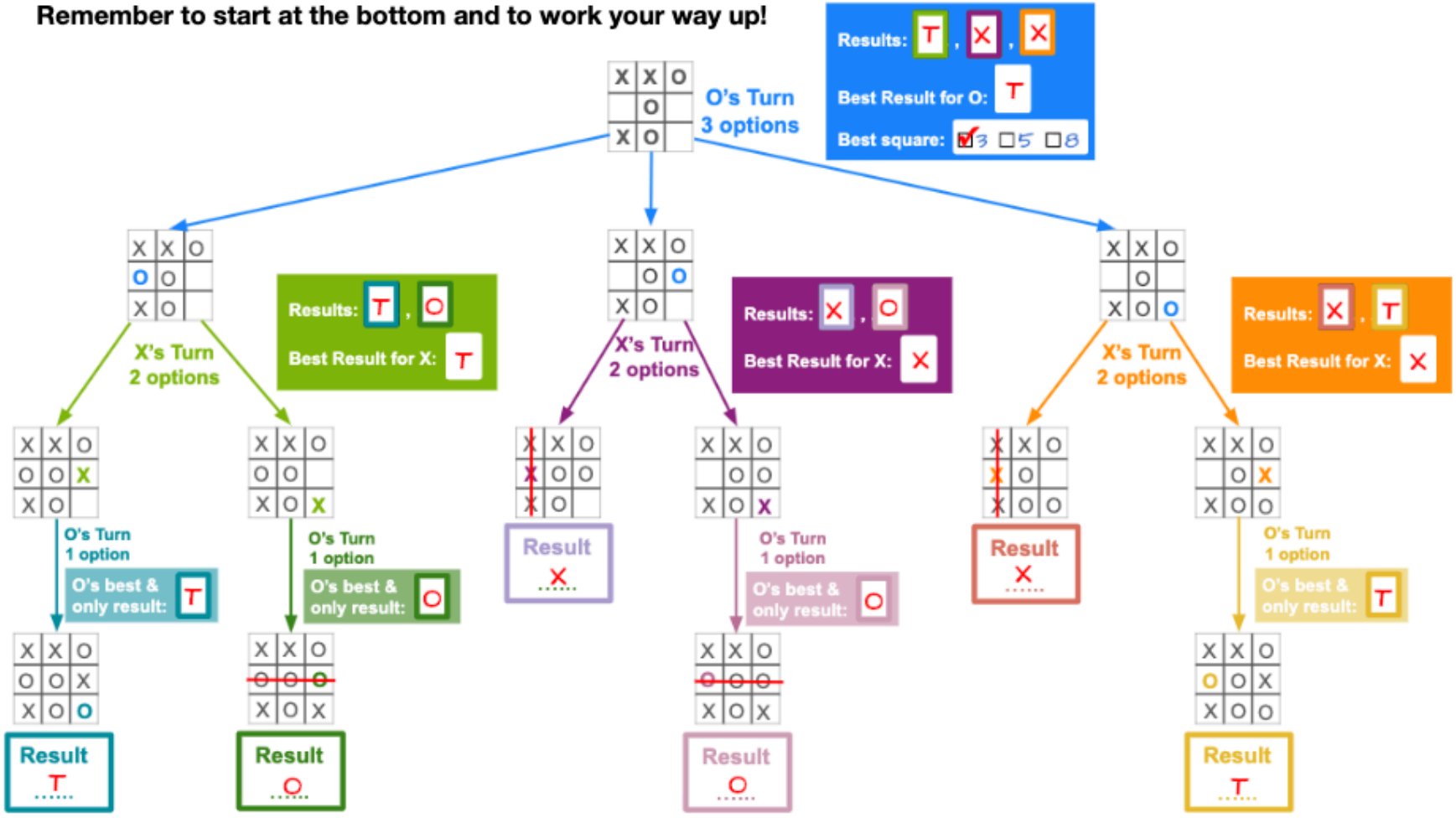
```
human_symbol = "0"
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")

print("Welcome", player_O, ", your symbol is 0!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
print_board(board)

# Extension 9
symbol = random.choice(["X","0"])
# Extension 12
if symbol == "0":
    current_player = player_O
else:
    current_player = player_X
print(symbol,"player will go first!")
free_squares = [0,1,2,3,4,5,6,7,8]
counter = 0
while not game_over:
    print("The current player is", current_player, "Who is playing as", symbol,"!")
#Extension 12

    # Extension 13
    if current_player == "computer":
        square = get_comp_move(board)
    else:
        square = input("Which square do you want your symbol to go in? ")
    square_index = int(square)

    #  Extension 9
    if square_index not in free_squares:
        print("That wasn't a valid move!")
        continue
    board[square_index] = symbol
    counter+=1

    print_board(board)
    free_squares = get_free_squares(board)
    game_over = check_winner(board)
    if game_over:
        print(current_player, "won! Congratulations!")
    elif counter == 9: # Extension 10
        print("It's a tie!")
        break

    symbol = get_opposite_symbol(symbol)
    if symbol == "0":
        current_player = player_O
    else:
        current_player = "computer"
```

SOLUTION TUTORS ONLY

Your Turn!

Work out the best move for O.

Remember to start at the bottom and to work your way up!



Part 4: The best move for me!

4.4 Check your work

- The new function should look like this:

```
def best_outcome_for_symbol(player_symbol, outcomes):
    if player_symbol in outcomes:
        return player_symbol
    elif "T" in outcomes:
        return "T"
    else:
        return get_other_symbol(player_symbol)
```

- Although it might also look like this, which is also correct:

```
def best_outcome_for_symbol(player_symbol, outcomes):
    best_outcome = get_other_symbol(player_symbol)
    if player_symbol in outcomes:
        best_outcome = player_symbol
    elif "T" in outcomes:
        best_outcome = "T"
    return best_outcome
```

Full Code Lesson 4

```
# Start your code here# Copy your previous code here...
import random

# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
# Be aware that students may have used the Option 2 code here
def check_winner(board) :
    if board[0] == board[1] == board[2] != " ":
        return True
    elif board[3] == board[4] == board[5] != " ":
        return True
    elif board[6] == board[7] == board[8] != " ":
        return True
    if board[0] == board[3] == board[6] != " ":
        return True
    elif board[1] == board[4] == board[7] != " ":
        return True
    elif board[2] == board[5] == board[8] != " ":
```

```
        return True
    if board[0] == board[4] == board[8] != " ":
        return True
    elif board[2] == board[4] == board[6] != " ":
        return True
    else:
        return False

def get_free_squares(board):
    free_squares = []
    for index, symbol in enumerate(board):
        if symbol == " ":
            free_squares.append(index)
    return free_squares

def get_comp_move(board):
    winning_moves = []
    free_squares = get_free_squares(board)
    for square in free_squares:
        board[square] = comp_symbol
        if check_winner(board):
            winning_moves.append(square)
        board[square] = " "
    if len(winning_moves) > 0:
        return winning_moves[0]
    return random.choice(free_squares)


def get_opposite_symbol(symbol):
    if symbol == "0":
        return "X"
    else:
        return "0"

def best_outcome_for_symbol(symbol, outcomes):
    if symbol in outcomes:
        return symbol
    elif "T" in outcomes:
        return "T"
    else:
        return get_other_symbol(player_symbol)

print("Welcome to Tic-Tac-Toe!")
comp_symbol = "X"
human_symbol = "0"
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")

print("Welcome", player_O, ", your symbol is 0!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
```

```
print_board(board)

# Extension 9
symbol = random.choice(["X","0"])
# Extension 12
if symbol == "0":
    current_player = player_O
else:
    current_player = "computer"
print(symbol,"player will go first!")
free_squares = [0,1,2,3,4,5,6,7,8]
counter = 0
while not game_over:
    print("The current player is", current_player, "Who is playing as", symbol,"!")
#Extension 12

    # Extension 13
    if current_player == "computer":
        square = get_comp_move(board)
    else:
        square = input("Which square do you want your symbol to go in? ")
    square_index = int(square)

    #  Extension 9
    if square_index not in free_squares:
        print("That wasn't a valid move!")
        continue
    board[square_index] = symbol
    counter+=1

    print_board(board)
    free_squares = get_free_squares(board)
    game_over = check_winner(board)
    if game_over:
        print(current_player, "won! Congratulations!")
    elif counter == 9: # Extension 10
        print("It's a tie!")
        break

    symbol = get_opposite_symbol(symbol)
    if symbol == "0":
        current_player = player_O
    else:
        current_player = "computer"
```

Part 5: The next, next, next moves

5.5 Space to place, but no way to win!

- The function should look like this:

```
def get_move_outcomes(player_symbol, board):
    free_squares = get_free_squares(board)

    if len(free_squares) == 0:
        return "T"

    outcomes = []
    for square in free_squares:
        board[square] = player_symbol
        if check_winner(board):
            outcomes.append(player_symbol)
        else:
            opponent_symbol = get_other_symbol(player_symbol)
            best_outcome = get_move_outcomes(opponent_symbol, board)
            outcomes.append(best_outcome)
        board[square] = " "

    return best_outcome_for_symbol(player_symbol, outcomes)
```

- There is no reason why the student can't append the result directly to the list. Just let them do whatever makes the most sense to them!

```
outcomes.append(get_move_outcomes(opponent_symbol, board))
```

Full Code Lesson 5

```
# Start your code here# Copy your previous code here...
import random

# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
# Be aware that students may have used the Option 2 code here
def check_winner(board) :
```

```
    if board[0] == board[1] == board[2] != " ":
        return True
    elif board[3] == board[4] == board[5] != " ":
        return True
    elif board[6] == board[7] == board[8] != " ":
        return True
    if board[0] == board[3] == board[6] != " ":
        return True
    elif board[1] == board[4] == board[7] != " ":
        return True
    elif board[2] == board[5] == board[8] != " ":
        return True
    if board[0] == board[4] == board[8] != " ":
        return True
    elif board[2] == board[4] == board[6] != " ":
        return True
    else:
        return False

def get_free_squares(board):
    free_squares = []
    for index, symbol in enumerate(board):
        if symbol == " ":
            free_squares.append(index)
    return free_squares

def get_comp_move(board):
    winning_moves = []
    free_squares = get_free_squares(board)
    for square in free_squares:
        board[square] = comp_symbol
        if check_winner(board):
            winning_moves.append(square)
        board[square] = " "
    if len(winning_moves) > 0:
        return winning_moves[0]
    return random.choice(free_squares)

def get_opposite_symbol(symbol):
    if symbol == "0":
        return = "X"
    else:
        return "0"

def best_outcome_for_symbol(symbol, outcomes):
    if symbol in outcomes:
        return symbol
    elif "T" in outcomes:
        return "T"
    else:
        return get_other_symbol(player_symbol)
```

```
def get_move_outcomes(symbol, board):
    free_squares = get_free_squares(board)
    if len(free_squares) == 0:
        return "T"
    results = []
    for square in free_squares:
        board[square] = symbol
        if check_winner(board):
            results.append(symbol)
        else:
            symbol = get_opposite_symbol(symbol)
            results.append(get_move_outcomes(symbol, board))
        board[square] = " "
    return best_outcome_for_symbol(symbol, results)


print("Welcome to Tic-Tac-Toe!")
comp_symbol = "X"
human_symbol = "0"
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")

print("Welcome", player_O, ", your symbol is 0!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
print_board(board)

# Extension 9
symbol = random.choice([comp_symbol,human_symbol])
# Extension 12
if symbol == human_symbol:
    current_player = player_O
else:
    current_player = "computer"
print(symbol,"player will go first!")
free_squares = [0,1,2,3,4,5,6,7,8]
counter = 0
while not game_over:
    print("The current player is", current_player, "Who is playing as", symbol,"!")
#Extension 12

    # Extension 13
    if current_player == "computer":
        square = get_comp_move(board)
    else:
        square = input("Which square do you want your symbol to go in? ")
    square_index = int(square)

    #  Extension 9
    if square_index not in free_squares:
```

```
        print("That wasn't a valid move!")
        continue
    board[square_index] = symbol
    counter+=1

    print_board(board)
    free_squares = get_free_squares(board)
    game_over = check_winner(board)
    if game_over:
        print(current_player, "won! Congratulations!")
    elif counter == 9: # Extension 10
        print("It's a tie!")
        break

    symbol = get_opposite_symbol(symbol)
    if symbol == human_symbol:
        current_player = player_O
    else:
        current_player = "computer"
```

Part 6: Computer can’t be beat

6.3 Choosing your favourite

- And finally the function should look like this

```
def get_comp_move(board):
    free_squares = get_free_squares(board)

    winning_moves = []
    tied_moves = []
    losing_moves = []

    for square in free_squares:
        board[square] = comp_symbol
        if check_winner(board):
```

```
            winning_moves.append(square)
        board[square] = " "

    if len(winning_moves) > 0:
        return winning_moves[0]
    else:
        for square in free_squares:
            board[square] = comp_symbol
            result = get_move_outcomes(human_symbol, board)
            if result == comp_symbol:
                winning_moves.append(square)
            elif result == "T":
                tied_moves.append(square)
            else:
                losing_moves.append(square)
            board[square] = " "

        if len(winning_moves) > 0:
            return winning_moves[0]
        elif len(tied_moves) > 0:
            return tied_moves[0]
        else:
            return losing_moves[0]
```

Full Code Lesson 6

```
# Start your code here# Copy your previous code here...
import random

# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
# Be aware that students may have used the Option 2 code here
def check_winner(board) :
    if board[0] == board[1] == board[2] != " ":
        return True
    elif board[3] == board[4] == board[5] != " ":
        return True
    elif board[6] == board[7] == board[8] != " ":
        return True
    if board[0] == board[3] == board[6] != " ":
        return True
    elif board[1] == board[4] == board[7] != " ":
        return True
    elif board[2] == board[5] == board[8] != " ":
        return True
```

```
    if board[0] == board[4] == board[8] != " ":
        return True
    elif board[2] == board[4] == board[6] != " ":
        return True
    else:
        return False

def get_free_squares(board):
    free_squares = []
    for index, symbol in enumerate(board):
        if symbol == " ":
            free_squares.append(index)
    return free_squares

def get_comp_move(board):
    free_squares = get_free_squares(board)

    winning_moves = []
    tied_moves = []
    losing_moves = []

    for square in free_squares:
        board[square] = comp_symbol
        if check_winner(board):
            winning_moves.append(square)
        board[square] = " "

    if len(winning_moves) > 0:
        return winning_moves[0]
    else:
        for square in free_squares:
            board[square] = comp_symbol
            result = get_move_outcomes(human_symbol, board)
            if result == comp_symbol:
                winning_moves.append(square)
            elif result == "T":
                tied_moves.append(square)
            else:
                losing_moves.append(square)
            board[square] = " "

        if len(winning_moves) > 0:
            return winning_moves[0]
        elif len(tied_moves) > 0:
            return tied_moves[0]
        else:
            return losing_moves[0]


def get_opposite_symbol(symbol):
    if symbol == "0":
        return "X"
    else:
```

```
        return "0"

def best_outcome_for_symbol(symbol, outcomes):
    if symbol in outcomes:
        return symbol
    elif "T" in outcomes:
        return "T"
    else:
        return get_opposite_symbol(symbol)

def get_move_outcomes(player_symbol, board):
    free_squares = get_free_squares(board)

    if len(free_squares) == 0:
        return "T"

    outcomes = []
    for square in free_squares:
        board[square] = player_symbol
        if check_winner(board):
            outcomes.append(player_symbol)
        else:
            opponent_symbol = get_opposite_symbol(player_symbol)
            best_outcome = get_move_outcomes(opponent_symbol, board)
            outcomes.append(best_outcome)
        board[square] = " "

    return best_outcome_for_symbol(player_symbol, outcomes)



print("Welcome to Tic-Tac-Toe!")
comp_symbol = "X"
human_symbol = "0"
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")

print("Welcome", player_O, ", your symbol is O!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
print_board(board)

# Extension 9
symbol = random.choice([comp_symbol,human_symbol])
# Extension 12
if symbol == human_symbol:
    current_player = player_O
else:
    current_player = "computer"
print(symbol,"player will go first!")
free_squares = [0,1,2,3,4,5,6,7,8]
counter = 0
while not game_over:
```

```
    print("The current player is", current_player, "Who is playing as", symbol,"!")
#Extension 12

    # Extension 13
    if current_player == "computer":
        square = get_comp_move(board)
    else:
        square = input("Which square do you want your symbol to go in? ")
    square_index = int(square)

    #  Extension 9
    if square_index not in free_squares:
        print("That wasn't a valid move!")
        continue
    board[square_index] = symbol
    counter+=1

    print_board(board)
    free_squares = get_free_squares(board)
    game_over = check_winner(board)
    if game_over:
        print(current_player, "won! Congratulations!")
    elif counter == 9: # Extension 10
        print("It's a tie!")
        break

    symbol = get_opposite_symbol(symbol)
    if symbol == human_symbol:
        current_player = player_O
    else:
        current_player = "computer"



# Copy your previous code here...
```

BONUS 6.5: Using a dictionary

- Most of the code for the bonus is provided above, it's just a matter of the student figuring out where to slot them in, highlighted below in bold:

```
def get_comp_move(board):
    free_squares = get_free_squares(board)
    outcomes = {comp_symbol: [], "T": [], human_symbol: []}

    for square in free_squares:
        board[square] = comp_symbol
        if check_winner(board):
            outcomes[comp_symbol].append(square)
        board[square] = " "

    if len(outcomes[comp_symbol]) > 0:
        return outcomes[comp_symbol][0]
    else:
        for square in free_squares:
            board[square] = comp_symbol
            result = get_move_outcomes(human_symbol, board)
            outcomes[result].append(square)
            board[square] = " "

        if len(outcomes[comp_symbol]) > 0:
            return outcomes[comp_symbol][0]
        elif len(outcomes["T"]) > 0:
            return outcomes["T"][0]
        else:
            return outcomes[human_symbol][0]
```

BONUS 6.5: Using a dictionary

```
# Start your code here# Copy your previous code here...
import random

# Copy your previous code here...
def print_board(board):
    print("-------------")
    print("|", board[0], "|", board[1], "|", board[2], "|")
    print("-------------")
    print("|", board[3], "|", board[4], "|", board[5], "|")
    print("-------------")
    print("|", board[6], "|", board[7], "|", board[8], "|")
    print("-------------")
# Be aware that students may have used the Option 2 code here
def check_winner(board) :
    if board[0] == board[1] == board[2] != " ":
        return True
    elif board[3] == board[4] == board[5] != " ":
        return True
    elif board[6] == board[7] == board[8] != " ":
        return True
    if board[0] == board[3] == board[6] != " ":
        return True
    elif board[1] == board[4] == board[7] != " ":
        return True
    elif board[2] == board[5] == board[8] != " ":
        return True
    if board[0] == board[4] == board[8] != " ":
        return True
    elif board[2] == board[4] == board[6] != " ":
        return True
    else:
        return False

def get_free_squares(board):
    free_squares = []
    for index, symbol in enumerate(board):
        if symbol == " ":
            free_squares.append(index)
    return free_squares

def get_comp_move(board):
    free_squares = get_free_squares(board)

    winning_moves = []
    tied_moves = []
    losing_moves = []

    for square in free_squares:
        board[square] = comp_symbol
        if check_winner(board):
            winning_moves.append(square)
        board[square] = " "
```

```
    if len(winning_moves) > 0:
        return winning_moves[0]
    else:
        for square in free_squares:
            board[square] = comp_symbol
            result = get_move_outcomes(human_symbol, board)
            if result == comp_symbol:
                winning_moves.append(square)
            elif result == "T":
                tied_moves.append(square)
            else:
                losing_moves.append(square)
            board[square] = " "

        if len(winning_moves) > 0:
            return winning_moves[0]
        elif len(tied_moves) > 0:
            return tied_moves[0]
        else:
            return losing_moves[0]


def get_opposite_symbol(symbol):
    if symbol == "0":
        return "X"
    else:
        return "0"

def best_outcome_for_symbol(symbol, outcomes):
    if symbol in outcomes:
        return symbol
    elif "T" in outcomes:
        return "T"
    else:
        return get_opposite_symbol(symbol)

def get_move_outcomes(player_symbol, board):
    free_squares = get_free_squares(board)

    if len(free_squares) == 0:
        return "T"

    outcomes = []
    for square in free_squares:
        board[square] = player_symbol
        if check_winner(board):
            outcomes.append(player_symbol)
        else:
            opponent_symbol = get_opposite_symbol(player_symbol)
            best_outcome = get_move_outcomes(opponent_symbol, board)
            outcomes.append(best_outcome)
        board[square] = " "
```

```
    return best_outcome_for_symbol(player_symbol, outcomes)



print("Welcome to Tic-Tac-Toe!")
comp_symbol = "X"
human_symbol = "0"
print("Welcome to Tic-Tac-Toe!")
player_O = input("Who is playing naughts? ")

print("Welcome", player_O, ", your symbol is O!")
board = [" ", " ", " ", " ", " ", " ", " ", " ", " "]
game_over = False
print_board(board)

# Extension 9
symbol = random.choice([comp_symbol,human_symbol])
# Extension 12
if symbol == human_symbol:
    current_player = player_O
else:
    current_player = "computer"
print(symbol,"player will go first!")
free_squares = [0,1,2,3,4,5,6,7,8]
counter = 0
while not game_over:
    print("The current player is", current_player, "Who is playing as", symbol,"!")
#Extension 12

    # Extension 13
    if current_player == "computer":
        square = get_comp_move(board)
    else:
        square = input("Which square do you want your symbol to go in? ")
    square_index = int(square)

    #  Extension 9
    if square_index not in free_squares:
        print("That wasn't a valid move!")
        continue
    board[square_index] = symbol
    counter+=1

    print_board(board)
    free_squares = get_free_squares(board)
    game_over = check_winner(board)
    if game_over:
        print(current_player, "won! Congratulations!")
    elif counter == 9: # Extension 10
        print("It's a tie!")
        break

    symbol = get_opposite_symbol(symbol)
```

```
    if symbol == human_symbol:
        current_player = player_O
    else:
        current_player = "computer"
```