**Outline**

Access the Python Development environment and continue the tutorial to gain an additional exposure to the Python programming language. Begin to develop an familiarity with intermediate programming concepts.

**Objectives**

* Use correct terminology to describe programming concepts;
* Describe the types of data that computers can process and store (e.g., numbers, text);
* Explain the difference between constants and variables used in programming;
* Use variables, expressions, and assignment statements to store and manipulate numbers and text in a program

**Materials**

* Python3 Development Environment at: //repl.it/
* Python Tutorial at: <http://www.letslearnpython.com/learn/>

**Accessing the Tutorial**

Accessing the Tutorial

* Go to: <http://www.letslearnpython.com/learn/>
* Read up to “Lesson 12: Input”

**Level 1: Input & Output**

1. Read through “Lesson 12: Input – What Is Input?” and “Lesson 12: Input – Example” and “Lesson 12: Input – Shortcut”.

I have read through lesson 12

1. Type the following code into the white area of the IDE and run the program. Explain what you see in the black area of the IDE.

print("Type your name:")

name = input()

print("Hi", name, "how are you?")

I see the text “Type your name:”. after I type my name and then I see “Hi Harjap how are you”.

1. Create a short program that reads numerical input from the console and does the following:
   1. Uses the input() function to read a numerical value from the console.
   2. Calculates the square root of the number
   3. Prints the result to the console output
   4. Provides appropriate prompt and message strings to go with the input and output.
   5. Provide your complete program below.

import math

def Root():

print("What number do you want to Square Root")

num = int(input())

print("The Sqaure Root of",num,"is",math.sqrt(num))

Root()

**Level 2: Tic-Tac-Toe Game**

1. Write a Python program to play a game of Toc-Tac-Toe. (You may modify a program that you found on-line to meet the expectations of this module.)
   1. The program may be either player v. computer or player 1 v. player 2.
   2. The program does not need to determine a winner
   3. The program just needs to keep track of moves and spaces in the game board
2. Provide a complete listing of your program.
   1. Your listing **MUST** include line numbers .

|  |  |
| --- | --- |
|  | import random  #GamePlay Board  Board = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  #3x3 board Positions 1-9  def showBoard():  print(Board[1], "│", Board[2], "│", Board[3])  print("━━━━━━━━━━")  print(Board[4], "│", Board[5], "│", Board[6])  print("━━━━━━━━━━")  print(Board[7], "│", Board[8], "│", Board[9])  #Checks each line for winning combination  def checkLine(player, spot1, spot2, spot3):  if Board[spot1] == player and Board[spot2] == player and Board[  spot3] == player:  return True  #Checks all posible combinations for winner  def checkWinner(player):  if checkLine(player, 1, 2, 3):  return True  if checkLine(player, 4, 5, 6):  return True  if checkLine(player, 7, 8, 9):  return True  if checkLine(player, 1, 4, 7):  return True  if checkLine(player, 2, 5, 8):  return True  if checkLine(player, 3, 6, 9):  return True  if checkLine(player, 1, 5, 9):  return True  if checkLine(player, 3, 5, 7):  return True  #Game function for moves and gameplay  while True:  showBoard()  #user input (1-9) User= X  X = input("Choose a spot (1-9):")  X = int(X)  if Board[X] != 'O' and Board[X] != 'X':  Board[X] = 'X'  else:  print("This spot is already taken")  #Checks if X won the game or not  if checkWinner('X') == True:  print("━━━━━━!!You beat the Computer, YOU WON!!━━━━━━")  break;  #Checks if game is DRAW  if checkWinner('O') == False and checkWinner('X') == False:  print('━━━━━━!!Nobody Wins, DRAW!!━━━━━━')  break;  #comuter Move Computer= O  while True:  O = random.randint(1, 9)  if Board[O] != 'O' and Board[O] != 'X':  Board[O] = 'O'  break;  #Checks if X won the game or not  if checkWinner('O') == True:  print('━━━━━━!!The Computer beat You, YOU LOST!!━━━━━━')  break;  showBoard() |

1. Explain how your program keeps track of the game board.   
   (Provide specific code references by line number.)
   1. What python types and data structures are used?
   2. How are moves by player X and player O recorded?
   3. How are free spaces recorded?

To keep track of my game board, I made a list titled Board (see Line 3), the list had indexes starting from 0-9. I used the indexes 1-9 for my game board. To do this, I made a function called showBoard (see Lines 7-12), in this function I placed the indexes 1-3 in the top row, 4-6 in the middle row and 7-9 in the bottom row. To add the grid shaped lines, I included vertical lines between the indexes contained the rows and I includer horizontal lines between the rows to complete the grid. The lines are shown as strings. To do all this I used the print() command in the showBoard function. To record the X moves, the number player x inputs is the index they want to please their x so that index gets replaced by the letter X (see Lines 44-49). To record the O moves, the number that is generated by the random number generator is the index that the computer wants to place O, so that index becomes O (see Lines 62-66). The indexes are only changed if the index is not already X or O (see Lines 46-49 and 64-66). The free spaces remain in 1-9 order with the index number being shown, so it is easy to play moves (see Lines 7-12). The game board is updated and shown after every computer and Player move (see Line 42).

1. Explain how moves and commands are input from the console.  
   (Provide specific code references by line number.)
   1. How does the player tell the program about the move location (row, column)?
   2. How does the program verify that the move location is valid?
   3. How does the program verify that the space is free?
   4. What does the program do if there is something wrong with the move?

The player is given choice of picking a number between 1 and 9, all these are spots on the Tic-Tac-Toe grid (see Line 44). When the player chooses the number, the number is then converted into an integer using the help of int(input()) (see Line 45), so it can be identified as an index of Board (see Line 3). The number then goes through a check which checks if the number/index that is selected is a valid move. To do this, the program checks if that index is an X or an O, if the index is not than the selected index becomes X, with the help if the “if” condition (see Lines 46-47). If the index is already used than the program prints “This spot is already taken” and the program asks the player for a different move (see Lines 48-49). The move inputted by the player can only be valid if it is between 1 and 9. If the number typed is bigger than 9 and smaller than 1, then the program stops, and an error is displayed. If something is wrong with the move like the input is not a number than the program stops and an error appears.

1. Explain how the program keeps track of gameplay.  
   (Provide specific code references by line number.)
   1. How does the program switch between player X and player O moves?
   2. How does the program keep asking for moves?
   3. How does the program decide when to stop asking for moves?

The program for moves is in a while loop which always true unless it is stooped with “break;” (see Line 41). This means that the moves switch between player x and o and thee loop keeps on going. For example, the loop starts, and player x is asked for a move and then player o goes, after this the loop restarts and player x goes again and then player o. The code that askes the player x for moves is in the loop so the player is asked every time the loop restarts (see Lines 41-71). The program stops when there is a winner or when there are no moves left. To check for a winner, the function “checkLine” checks each line for a winning combination (see Lines 16-19). The function “checkWinner” checks the whole board for any possible winning combinations (see Lines 22-38). The “checkWinner” function is placed after each move by x and o, to check for a winner every move (see Lines 52-54 and 69-71). If there is a winner the game loop stops with the help of break; and if x wins "━━━━━━!!You beat the Computer, YOU WON!!━━━━━━" (see Line 53) is printed and if o wins '━━━━━━!!The Computer beat You, YOU LOST!!━━━━━━' (see Line 70) is printed. The program only stops if a player wins or there are no moves left. To keep track of the gameplay, the showBoard function is placed in the loop. This shows the board after each move (see Line 42).

**Level 3: Basic Enhancements**

1. Explain, in plain words, a strategy for determining if player “x” or player “O” has won the game after a move is made.

A strategy for this would need to include all possible ways a player could win. In Tic-Tac-Toe, the player only wins if the player has 3 of his characters in a straight line and connects. If the spots on the board are numbered 1-9, with 1 being the top left and 9 being the bottom right and the numbers increasing going left to right and top to bottom. There are 3 ways to win, the first is to have 3 character in a row for the 3 rows in the game. The 3 rows would be 1 2 3 , 4 5 6 and 7 8 9. The second way to win is by placing 3 characters in the 3 columns and the 3 columns are 1 4 7 , 2 5 8 and 3 6 9. The last way to win is by placing 3 characters in the to diagonals of the 3x3 board and the diagonal combinations are 1 5 9 and 3 5 7. With all this information, we can determine strategy of determining a winner. To do this we need to a function to look at the gameboard after each move as that’s when the board changes. The function must check all the possible winning combinations I listed above, and the combinations must only contain one character. If there is a winning combination, the function should end the game and print a result. This function must be placed after every move. In short, to determine a winner, we need to check the board for ALL wining combinations.

1. Provide a function called “checkWinForX” that returns the Boolean value of “True” if player “x” won the game.

|  |  |
| --- | --- |
|  | #Checks each line for winning combination  def checkLine(player, spot1, spot2, spot3):  if Board[spot1] == player and Board[spot2] == player and Board[  spot3] == player:  return True  #Checks all posible combinations for winner  def checkWinner(player):  if checkLine(player, 1, 2, 3):  return True  if checkLine(player, 4, 5, 6):  return True  if checkLine(player, 7, 8, 9):  return True  if checkLine(player, 1, 4, 7):  return True  if checkLine(player, 2, 5, 8):  return True  if checkLine(player, 3, 6, 9):  return True  if checkLine(player, 1, 5, 9):  return True  if checkLine(player, 3, 5, 7):  return True |

1. Modify your program to check and print a message, and stop the game of player “x” or player “O” wins the game.

|  |  |
| --- | --- |
|  | #Checks each line for winning combination  def checkLine(player, spot1, spot2, spot3):  if Board[spot1] == player and Board[spot2] == player and Board[  spot3] == player:  return True  #Checks all posible combinations for winner  def checkWinner(player):  if checkLine(player, 1, 2, 3):  return True  if checkLine(player, 4, 5, 6):  return True  if checkLine(player, 7, 8, 9):  return True  if checkLine(player, 1, 4, 7):  return True  if checkLine(player, 2, 5, 8):  return True  if checkLine(player, 3, 6, 9):  return True  if checkLine(player, 1, 5, 9):  return True  if checkLine(player, 3, 5, 7):  return True  #Checks if X won the game or not  if checkWinner('X') == True:  print("━━━━━━!!You beat the Computer, YOU WON!!━━━━━━")  break;  #Checks if O won the game or not  if checkWinner('O') == True:  print('━━━━━━!!The Computer beat You, YOU LOST!!━━━━━━')  break; |

1. Demonstrate your enhanced game to Mr. Nestor for credit for this level.

**Level 4: AI Enhancements**

1. Explain, in plain words, a strategy for suggesting the best move for player “x” or player “O” to make when it is their turn to move.

For Tic Tac Toe, there are 5 stages in determining the best move for the player. The first stage is to check if the player is in a winning position and could win in the next move. If this is the case, then the best move would be to place in the spot where the player would win. All the positions where the player could win in the next move will be needed to check so all the winning combinations and their combination of 3 characters needed to be checked. If the player is not in a winning position, the second stage comes into play, the stage is that if the opponent is in a winning position than the best move becomes to block the winning position. If this is the case, then all the winning positions are check along with the position of the 3 character in those positions, if there are 2 opponent characters in a winning position than the best move becomes to the 3rd position in the winning position to block the opponent from winning. If this is not the case, then the 3rd stage is for the best move is to place the character in the middle. The middle would be the best move because the middle has the most winning positions and possibilities. If this is not the case then the 4th stage would be to place the character in the corners because there are a lot of wining positions and possibilists. If this is not the case, then the last and 5th stage comes into play, which is to place anywhere on the sides. The function for this needs to be in a loop where it goes through all the 5 stages and stops if one of these stages are true. All these stages will result in the best move for the player.

1. Create a function to implement your strategy for suggesting the best move.

This function is for the computer move. This function finds the best move for the computer and plays that move. The function uses the same 5 stages to find the best move as listed above.

def computerMove():

while True:

#Checks if O is in winning position

if Board[1] == 'O' and Board[2] == 'O' and Board[3] != 'O' and Board[3] != 'X' :

Board[3] = 'O'

break;

if Board[1] == 'O' and Board[3] == 'O' and Board[2] != 'O' and Board[2] != 'X':

Board[2] = 'O'

break;

if Board[3] == 'O' and Board[2] == 'O' and Board[1] != 'O' and Board[1] != 'X':

Board[1] = 'O'

break;

if Board[4] == 'O' and Board[5] == 'O' and Board[6] != 'O' and Board[6] != 'X':

Board[6] = 'O'

break;

if Board[4] == 'O' and Board[6] == 'O' and Board[5] != 'O' and Board[5] != 'X':

Board[5] = 'O'

break;

if Board[6] == 'O' and Board[5] == 'O' and Board[4] != 'O' and Board[4] != 'X':

Board[4] = 'O'

break;

if Board[7] == 'O' and Board[8] == 'O' and Board[9] != 'O' and Board[9] != 'X':

Board[9] = 'O'

break;

if Board[7] == 'O' and Board[9] == 'O' and Board[8] != 'O' and Board[8] != 'X':

Board[8] = 'O'

break;

if Board[8] == 'O' and Board[9] == 'O' and Board[7] != 'O' and Board[7] != 'X':

Board[7] = 'O'

break;

if Board[1] == 'O' and Board[4] == 'O' and Board[7] != 'O' and Board[7] != 'X':

Board[7] = 'O'

break;

if Board[1] == 'O' and Board[7] == 'O' and Board[4] != 'O' and Board[4] != 'X':

Board[4] = 'O'

break;

if Board[4] == 'O' and Board[7] == 'O' and Board[1] != 'O' and Board[1] != 'X':

Board[1] = 'O'

break;

if Board[2] == 'O' and Board[5] == 'O' and Board[8] != 'O' and Board[8] != 'X':

Board[8] = 'O'

break;

if Board[2] == 'O' and Board[8] == 'O' and Board[5] != 'O' and Board[5] != 'X':

Board[5] = 'O'

break;

if Board[5] == 'O' and Board[8] == 'O' and Board[2] != 'O' and Board[2] != 'X':

Board[2] = 'O'

break;

if Board[3] == 'O' and Board[6] == 'O' and Board[9] != 'O' and Board[9] != 'X':

Board[9] = 'O'

break;

if Board[3] == 'O' and Board[9] == 'O' and Board[6] != 'O' and Board[6] != 'X':

Board[6] = 'O'

break;

if Board[6] == 'O' and Board[9] == 'O' and Board[3] != 'O' and Board[3] != 'X':

Board[3] = 'O'

break;

if Board[1] == 'O' and Board[5] == 'O' and Board[9] != 'O' and Board[9] != 'X':

Board[9] = 'O'

break;

if Board[1] == 'O' and Board[9] == 'O' and Board[5] != 'O' and Board[5] != 'X':

Board[5] = 'O'

break;

if Board[5] == 'O' and Board[9] == 'O' and Board[1] != 'O' and Board[1] != 'X':

Board[1] = 'O'

break;

if Board[3] == 'O' and Board[5] == 'O' and Board[7] != 'O' and Board[7] != 'X':

Board[7] = 'O'

break;

if Board[3] == 'O' and Board[7] == 'O' and Board[5] != 'O' and Board[5] != 'X':

Board[5] = 'O'

break;

if Board[5] == 'O' and Board[7] == 'O' and Board[3] != 'O' and Board[3] != 'X':

Board[3] = 'O'

break;

#Places O if X is in a winning position

if Board[1] == 'X' and Board[2] == 'X' and Board[3] != 'O' and Board[3] != 'X':

Board[3] = 'O'

break;

if Board[1] == 'X' and Board[3] == 'X' and Board[2] != 'O' and Board[2] != 'X':

Board[2] = 'O'

break;

if Board[3] == 'X' and Board[2] == 'X' and Board[1] != 'O' and Board[1] != 'X':

Board[1] = 'O'

break;

if Board[4] == 'X' and Board[5] == 'X' and Board[6] != 'O' and Board[6] != 'X':

Board[6] = 'O'

break;

if Board[4] == 'X' and Board[6] == 'X' and Board[5] != 'O' and Board[5] != 'X':

Board[5] = 'O'

break;

if Board[6] == 'X' and Board[5] == 'X' and Board[4] != 'O' and Board[4] != 'X':

Board[4] = 'O'

break;

if Board[7] == 'X' and Board[8] == 'X' and Board[9] != 'O' and Board[9] != 'X':

Board[9] = 'O'

break;

if Board[7] == 'X' and Board[9] == 'X' and Board[8] != 'O' and Board[8] != 'X':

Board[8] = 'O'

break;

if Board[8] == 'X' and Board[9] == 'X' and Board[7] != 'O' and Board[7] != 'X':

Board[7] = 'O'

break;

if Board[1] == 'X' and Board[4] == 'X' and Board[7] != 'O' and Board[7] != 'X':

Board[7] = 'O'

break;

if Board[1] == 'X' and Board[7] == 'X' and Board[7] != 'O' and Board[7] != 'X':

Board[4] = 'O'

break;

if Board[4] == 'X' and Board[7] == 'X' and Board[1] != 'O' and Board[1] != 'X':

Board[1] = 'O'

break;

if Board[2] == 'X' and Board[5] == 'X' and Board[8] != 'O' and Board[8] != 'X':

Board[8] = 'O'

break;

if Board[2] == 'X' and Board[8] == 'X' and Board[5] != 'O' and Board[5] != 'X':

Board[5] = 'O'

break;

if Board[5] == 'X' and Board[8] == 'X' and Board[2] != 'O' and Board[2] != 'X':

Board[2] = 'O'

break;

if Board[3] == 'X' and Board[6] == 'X' and Board[9] != 'O' and Board[9] != 'X':

Board[9] = 'O'

break;

if Board[3] == 'X' and Board[9] == 'X' and Board[6] != 'O' and Board[6] != 'X':

Board[6] = 'O'

break;

if Board[6] == 'X' and Board[9] == 'X' and Board[3] != 'O' and Board[3] != 'X':

Board[3] = 'O'

break;

if Board[1] == 'X' and Board[5] == 'X' and Board[9] != 'O' and Board[9] != 'X':

Board[9] = 'O'

break;

if Board[1] == 'X' and Board[9] == 'X' and Board[5] != 'O' and Board[5] != 'X':

Board[5] = 'O'

break;

if Board[5] == 'X' and Board[9] == 'X' and Board[1] != 'O' and Board[1] != 'X':

Board[1] = 'O'

break;

if Board[3] == 'X' and Board[5] == 'X' and Board[7] != 'O' and Board[7] != 'X':

Board[7] = 'O'

break;

if Board[3] == 'X' and Board[7] == 'X' and Board[5] != 'O' and Board[5] != 'X':

Board[5] = 'O'

break;

if Board[5] == 'X' and Board[7] == 'X' and Board[3] != 'O' and Board[3] != 'X':

Board[3] = 'O'

break;

#places O in the middle if it is Free

if Board[5] != 'O' and Board[5] != 'X':

Board[5] = 'O'

break;

#places O n the Corners if corner is Free

if Board[1] != 'O' and Board[1] != 'X':

Board[1] = 'O'

break;

if Board[3] != 'O' and Board[3] != 'X':

Board[3] = 'O'

break;

if Board[7] != 'O' and Board[7] != 'X':

Board[7] = 'O'

break;

if Board[9] != 'O' and Board[9] != 'X':

Board[9] = 'O'

break;

#Checks is sides are empty and places O if free

if Board[2] != 'O' and Board[2] != 'X':

Board[2] = 'O'

break;

if Board[6] != 'O' and Board[6] != 'X':

Board[6] = 'O'

break;

if Board[8] != 'O' and Board[8] != 'X':

Board[8] = 'O'

break;

if Board[4] != 'O' and Board[4] != 'X':

Board[4] = 'O'

break;

1. Modify your program to print a suggested move when it is each player’s turn to move.

This function suggests the best move for player X, which is the User.

def BestMove():

while True:

if Board[1] == 'X' and Board[2] == 'X' and Board[3] != 'O' and Board[3] != 'X':

print('You should place X in spot #3')

break;

if Board[1] == 'X' and Board[3] == 'X' and Board[2] != 'O' and Board[2] != 'X':

print('You should place X in spot #2')

break;

if Board[3] == 'X' and Board[2] == 'X' and Board[1] != 'O' and Board[1] != 'X':

print('You should place X in spot #1')

break;

if Board[4] == 'X' and Board[5] == 'X' and Board[6] != 'O' and Board[6] != 'X':

print('You should place X in spot #6')

break;

if Board[4] == 'X' and Board[6] == 'X' and Board[5] != 'O' and Board[5] != 'X':

print('You should place X in spot #5')

break;

if Board[6] == 'X' and Board[5] == 'X' and Board[4] != 'O' and Board[4] != 'X':

print('You should place X in spot #4')

break;

if Board[7] == 'X' and Board[8] == 'X' and Board[9] != 'O' and Board[9] != 'X':

print('You should place X in spot #9')

break;

if Board[7] == 'X' and Board[9] == 'X' and Board[8] != 'O' and Board[8] != 'X':

print('You should place X in spot #8')

break;

if Board[8] == 'X' and Board[9] == 'X' and Board[7] != 'O' and Board[7] != 'X':

print('You should place X in spot #7')

break;

if Board[1] == 'X' and Board[4] == 'X' and Board[7] != 'O' and Board[7] != 'X':

print('You should place X in spot #7')

break;

if Board[1] == 'X' and Board[7] == 'X' and Board[4] != 'O' and Board[4] != 'X':

print('You should place X in spot #4')

break;

if Board[4] == 'X' and Board[7] == 'X' and Board[1] != 'O' and Board[1] != 'X':

print('You should place X in spot #1')

break;

if Board[2] == 'X' and Board[5] == 'X' and Board[8] != 'O' and Board[8] != 'X':

print('You should place X in spot #8')

break;

if Board[2] == 'X' and Board[8] == 'X' and Board[5] != 'O' and Board[5] != 'X':

print('You should place X in spot #5')

break;

if Board[5] == 'X' and Board[8] == 'X' and Board[2] != 'O' and Board[2] != 'X':

print('You should place X in spot #2')

break;

if Board[3] == 'X' and Board[6] == 'X' and Board[9] != 'O' and Board[9] != 'X':

print('You should place X in spot #9')

break;

if Board[3] == 'X' and Board[9] == 'X' and Board[6] != 'O' and Board[6] != 'X':

print('You should place X in spot #6')

break;

if Board[6] == 'X' and Board[9] == 'X' and Board[3] != 'O' and Board[3] != 'X':

print('You should place X in spot #3')

break;

if Board[1] == 'X' and Board[5] == 'X' and Board[9] != 'O' and Board[9] != 'X':

print('You should place X in spot #9')

break;

if Board[1] == 'X' and Board[9] == 'X' and Board[5] != 'O' and Board[5] != 'X':

print('You should place X in spot #5')

break;

if Board[5] == 'X' and Board[9] == 'X' and Board[1] != 'O' and Board[1] != 'X':

print('You should place X in spot #1')

break;

if Board[3] == 'X' and Board[5] == 'X' and Board[7] != 'O' and Board[7] != 'X':

print('You should place X in spot #7')

break;

if Board[3] == 'X' and Board[7] == 'X' and Board[5] != 'O' and Board[5] != 'X':

print('You should place X in spot #5')

break;

if Board[5] == 'X' and Board[7] == 'X' and Board[3] != 'O' and Board[3] != 'X':

print('You should place X in spot #3')

break;

if Board[1] == 'O' and Board[2] == 'O' and Board[3] != 'O' and Board[3] != 'X' :

print('You should place X in spot #3')

break;

if Board[1] == 'O' and Board[3] == 'O' and Board[2] != 'O' and Board[2] != 'X':

print('You should place X in spot #2')

break;

if Board[3] == 'O' and Board[2] == 'O' and Board[1] != 'O' and Board[1] != 'X':

print('You should place X in spot #1')

break;

if Board[4] == 'O' and Board[5] == 'O' and Board[6] != 'O' and Board[6] != 'X':

print('You should place X in spot #6')

break;

if Board[4] == 'O' and Board[6] == 'O' and Board[5] != 'O' and Board[5] != 'X':

print('You should place X in spot #5')

break;

if Board[6] == 'O' and Board[5] == 'O' and Board[4] != 'O' and Board[4] != 'X':

print('You should place X in spot #4')

break;

if Board[7] == 'O' and Board[8] == 'O' and Board[9] != 'O' and Board[9] != 'X':

print('You should place X in spot #9')

break;

if Board[7] == 'O' and Board[9] == 'O' and Board[8] != 'O' and Board[8] != 'X':

print('You should place X in spot #8')

break;

if Board[8] == 'O' and Board[9] == 'O' and Board[7] != 'O' and Board[7] != 'X':

print('You should place X in spot #7')

break;

if Board[1] == 'O' and Board[4] == 'O' and Board[7] != 'O' and Board[7] != 'X':

print('You should place X in spot #7')

break;

if Board[1] == 'O' and Board[7] == 'O' and Board[4] != 'O' and Board[4] != 'X':

print('You should place X in spot #4')

break;

if Board[4] == 'O' and Board[7] == 'O' and Board[1] != 'O' and Board[1] != 'X':

print('You should place X in spot #1')

break;

if Board[2] == 'O' and Board[5] == 'O' and Board[8] != 'O' and Board[8] != 'X':

print('You should place X in spot #8')

break;

if Board[2] == 'O' and Board[8] == 'O' and Board[5] != 'O' and Board[5] != 'X':

print('You should place X in spot #5')

break;

if Board[5] == 'O' and Board[8] == 'O' and Board[2] != 'O' and Board[2] != 'X':

print('You should place X in spot #2')

break;

if Board[3] == 'O' and Board[6] == 'O' and Board[9] != 'O' and Board[9] != 'X':

print('You should place X in spot #9')

break;

if Board[3] == 'O' and Board[9] == 'O' and Board[6] != 'O' and Board[6] != 'X':

print('You should place X in spot #6')

break;

if Board[6] == 'O' and Board[9] == 'O' and Board[3] != 'O' and Board[3] != 'X':

print('You should place X in spot #3')

break;

if Board[1] == 'O' and Board[5] == 'O' and Board[9] != 'O' and Board[9] != 'X':

print('You should place X in spot #9')

break;

if Board[1] == 'O' and Board[9] == 'O' and Board[5] != 'O' and Board[5] != 'X':

print('You should place X in spot #5')

break;

if Board[5] == 'O' and Board[9] == 'O' and Board[1] != 'O' and Board[1] != 'X':

print('You should place X in spot #1')

break;

if Board[3] == 'O' and Board[5] == 'O' and Board[7] != 'O' and Board[7] != 'X':

print('You should place X in spot #7')

break;

if Board[3] == 'O' and Board[7] == 'O' and Board[5] != 'O' and Board[5] != 'X':

print('You should place X in spot #5')

break;

if Board[5] == 'O' and Board[7] == 'O' and Board[3] != 'O' and Board[3] != 'X':

print('You should place X in spot #3')

break;

if Board[5] != 'O' and Board[5] != 'X':

print('You should place X in spot #5')

break;

if Board[1] != 'O' and Board[1] != 'X':

print('You should place X in spot #1')

break;

if Board[3] != 'O' and Board[3] != 'X':

print('You should place X in spot #3')

break;

if Board[7] != 'O' and Board[7] != 'X':

print('You should place X in spot #7')

break;

if Board[9] != 'O' and Board[9] != 'X':

print('You should place X in spot #9')

break;

if Board[2] != 'O' and Board[2] != 'X':

print('You should place X in spot #2')

break;

if Board[6] != 'O' and Board[6] != 'X':

print('You should place X in spot #6')

break;

if Board[8] != 'O' and Board[8] != 'X':

print('You should place X in spot #8')

break;

if Board[4] != 'O' and Board[4] != 'X':

print('You should place X in spot #4')

break;

1. Demonstrate your AI enhanced game to Mr. Nestor for credit for this level.