**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 program2

**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**

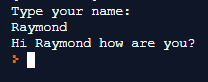
1. Read through “Lesson 12: Input – What Is Input?” and “Lesson 12: Input – Example” and “Lesson 12: Input – Shortcut”.

2. 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?")



3. Create a short program that reads numerical input from the console and does the following:

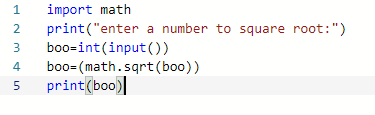
a. Uses the input() function to read a numerical value from the console.

b. Calculates the square root of the number

c. Prints the result to the console output

d. Provides appropriate prompt and message strings to go with the input and output.

e. Provide your complete program below.



**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.)

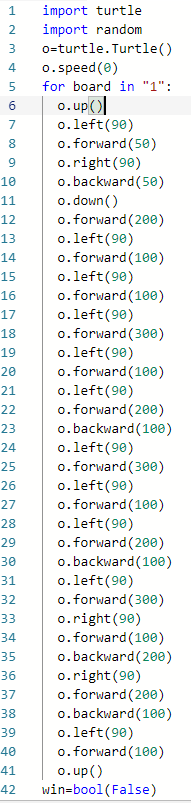
a. The program may be either player v. computer or player 1 v. player 2.

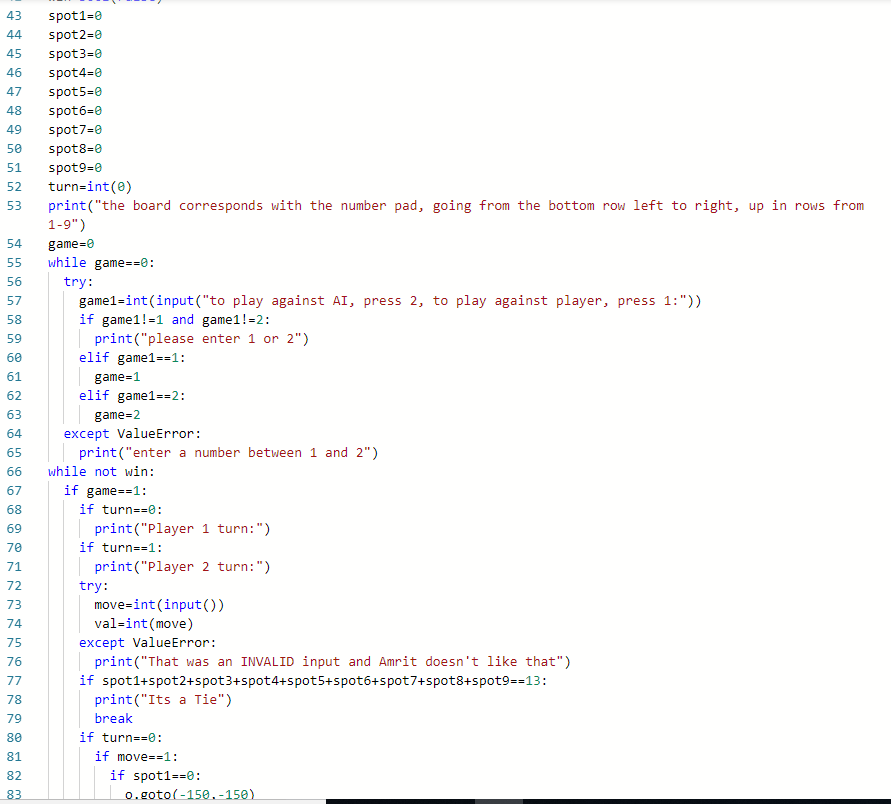
b. The program does not need to determine a winner

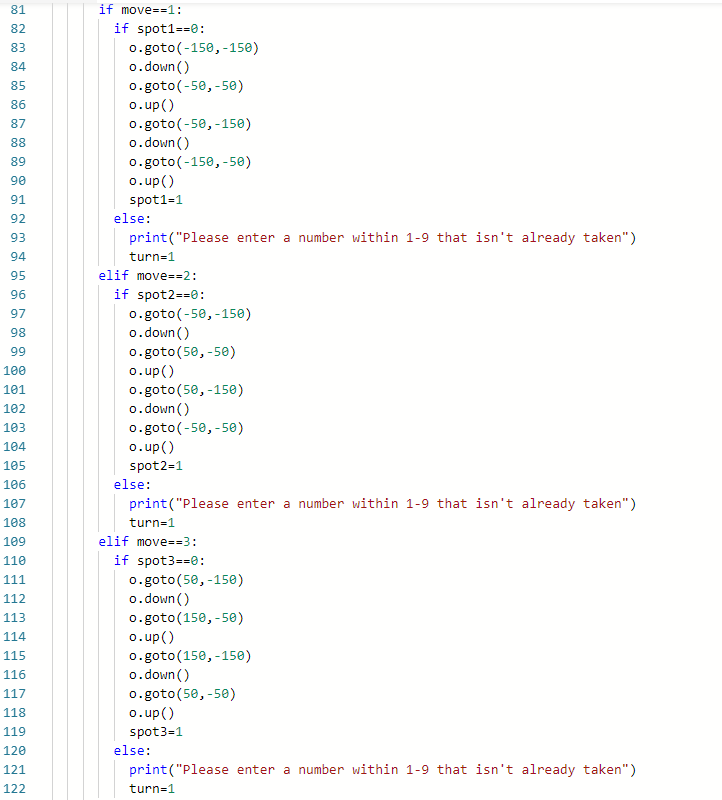
c. The program just needs to keep track of moves and spaces in the game board

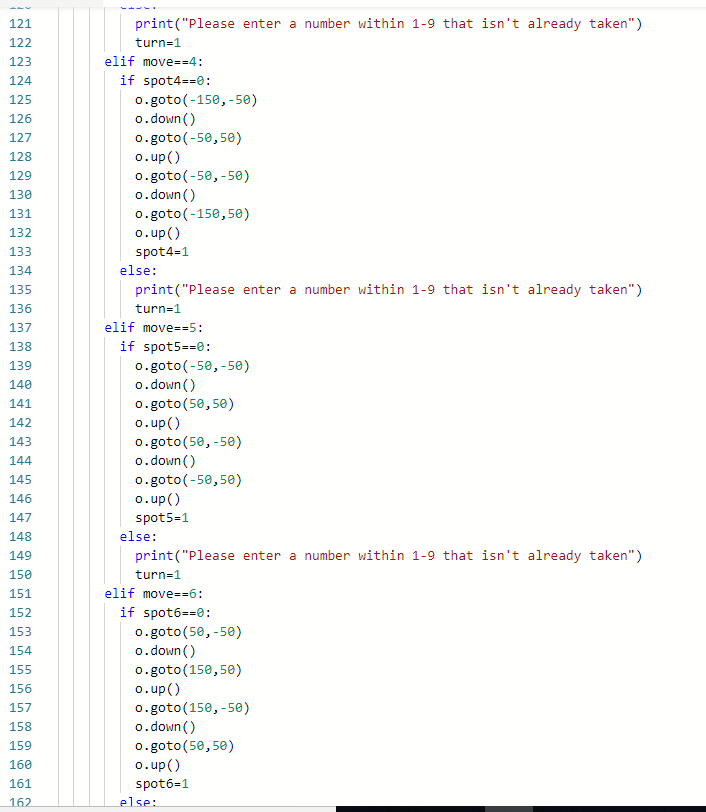
2. Provide a complete listing of your program.

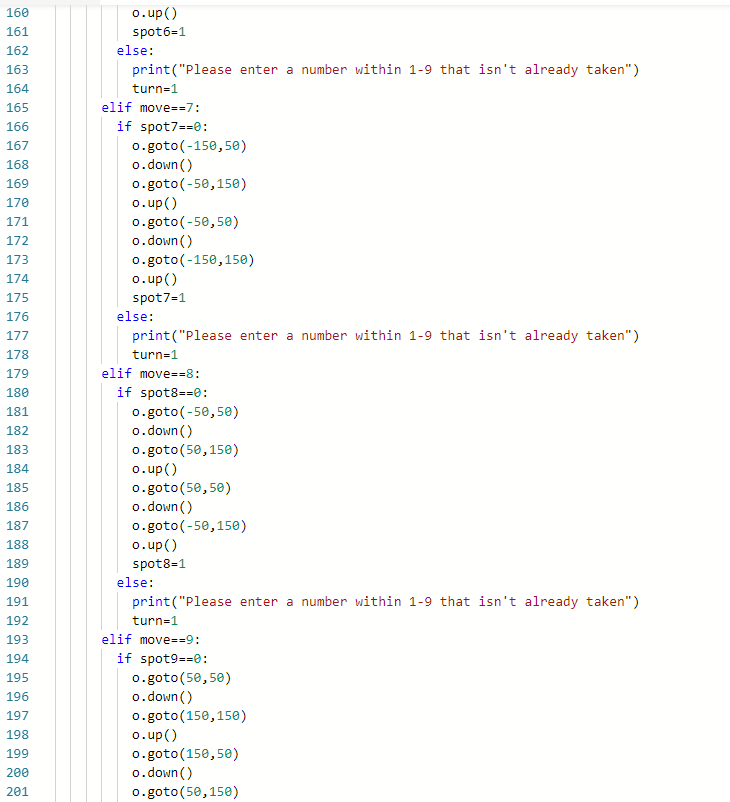
a. Your listing **MUST** include line numbers .

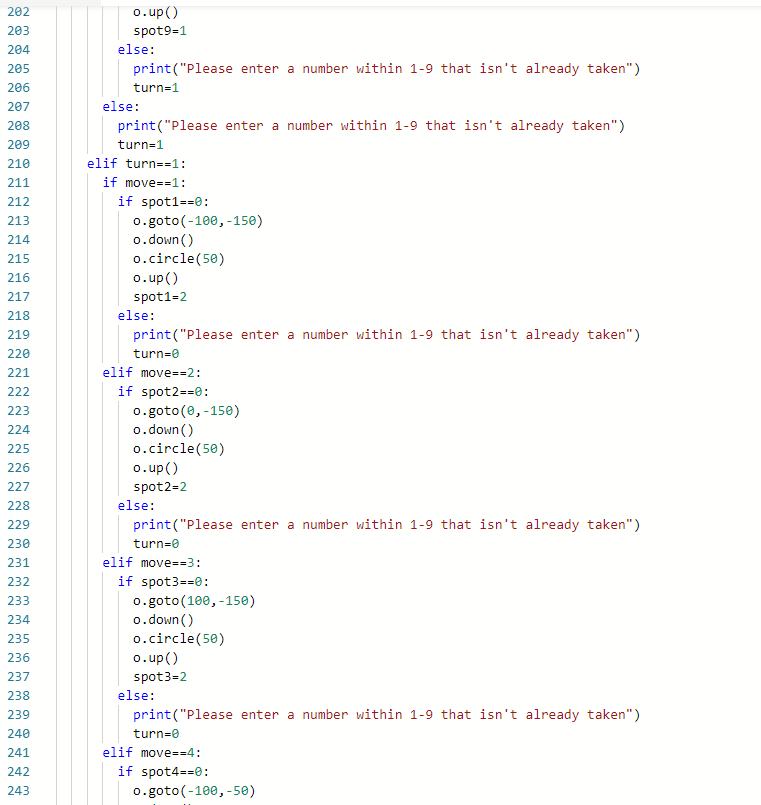


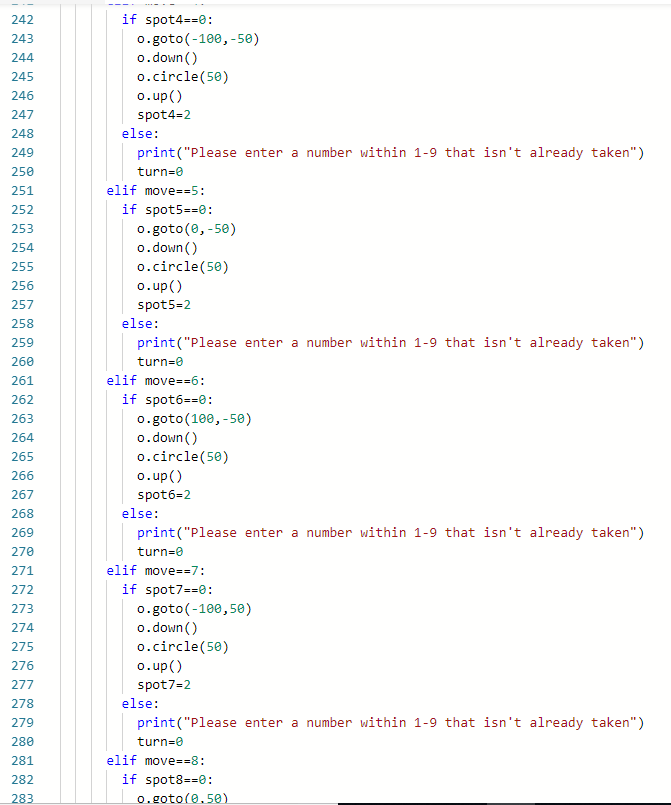


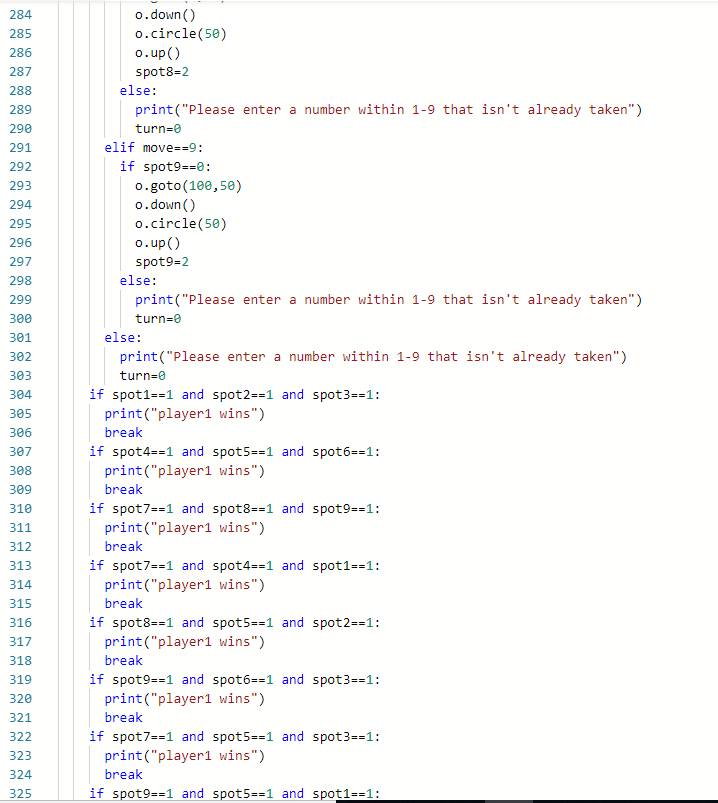




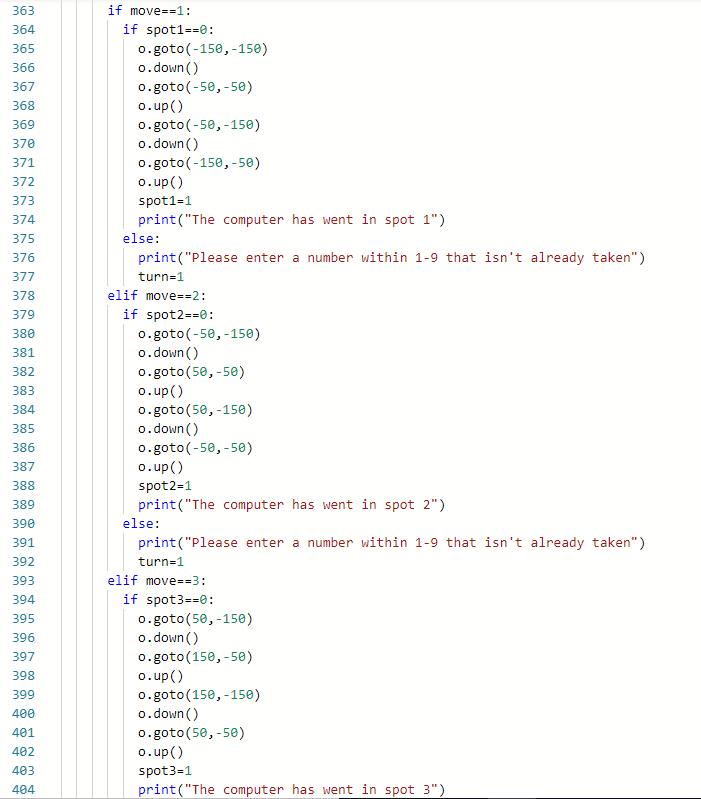


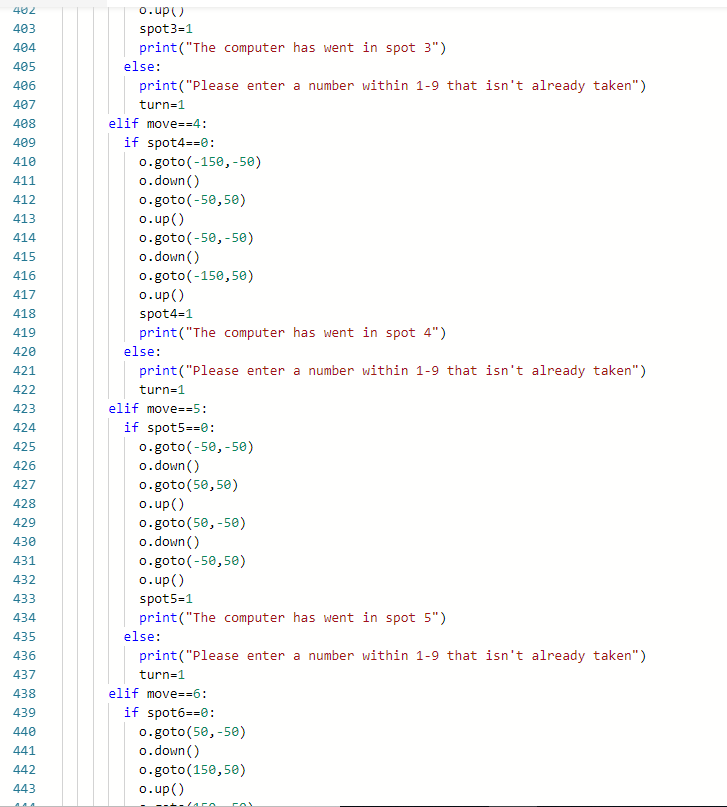


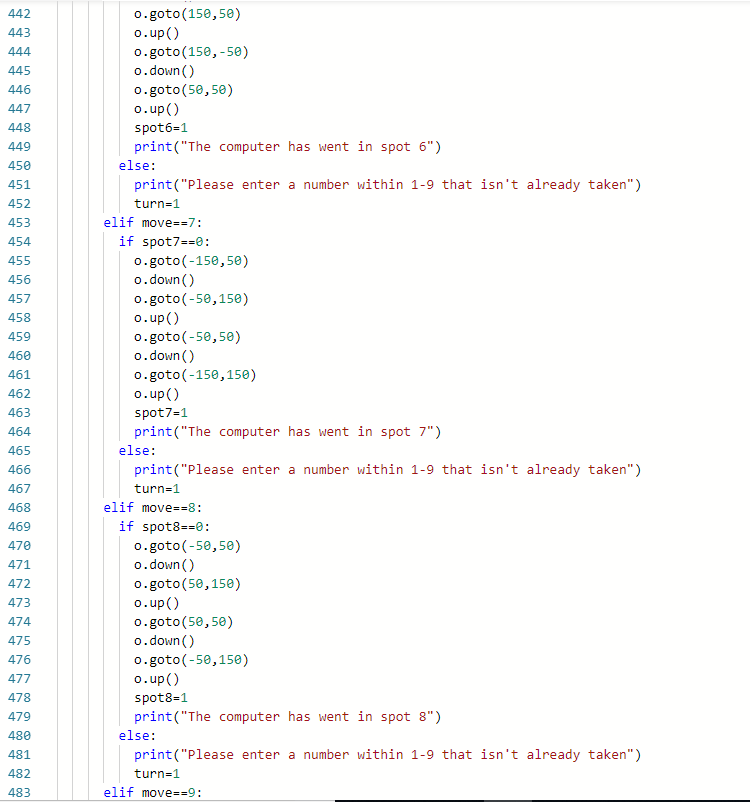


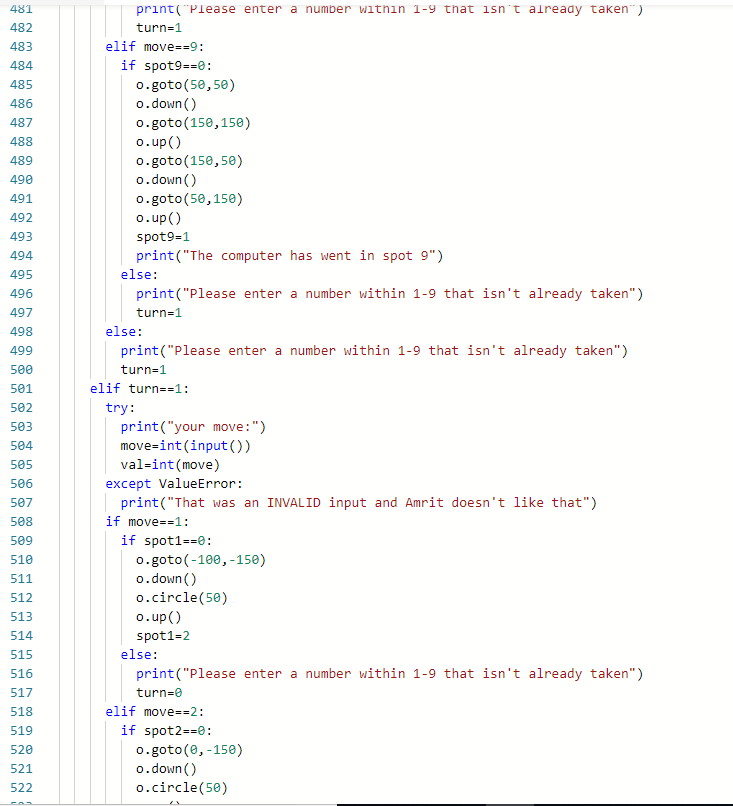


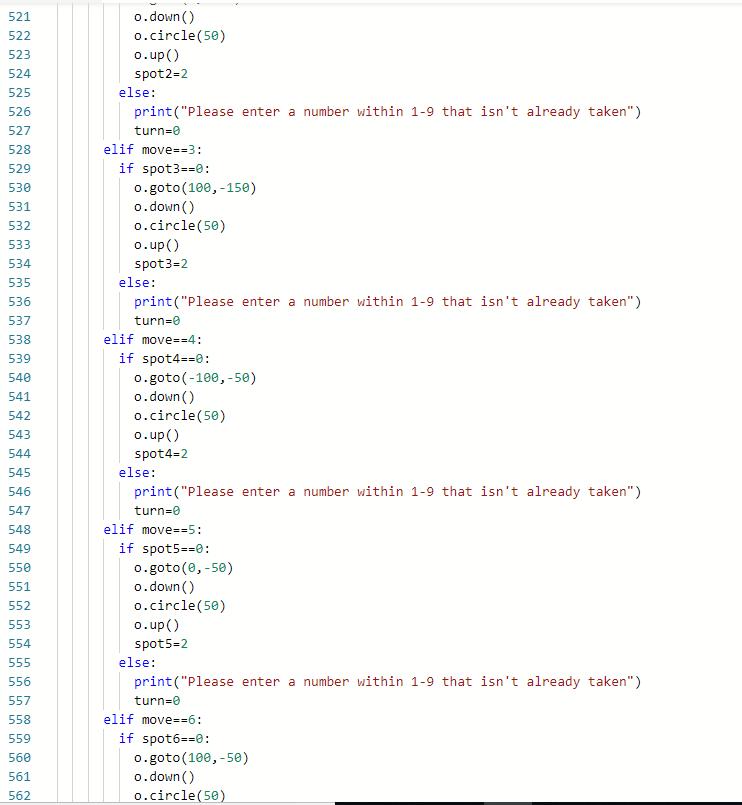


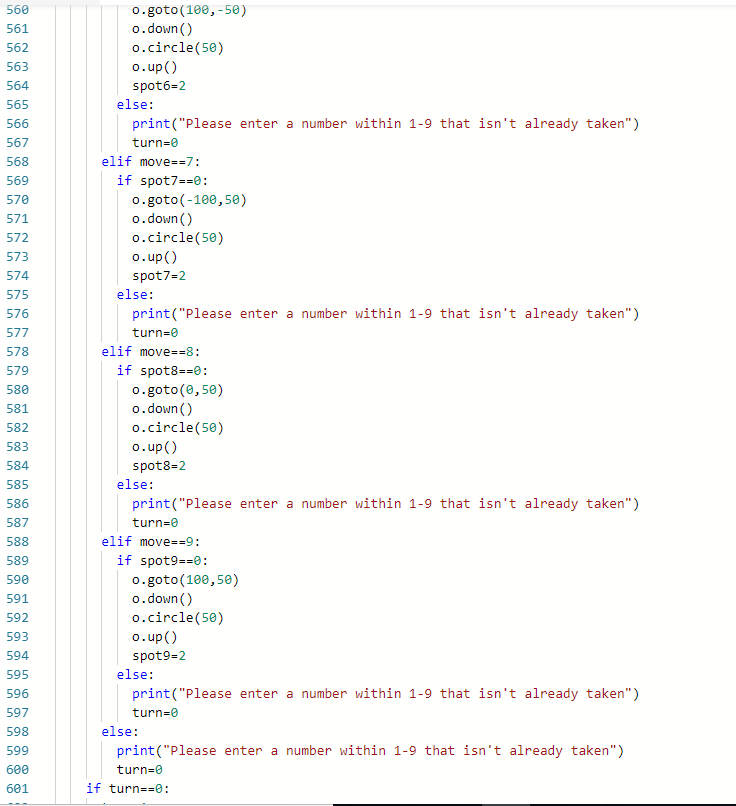


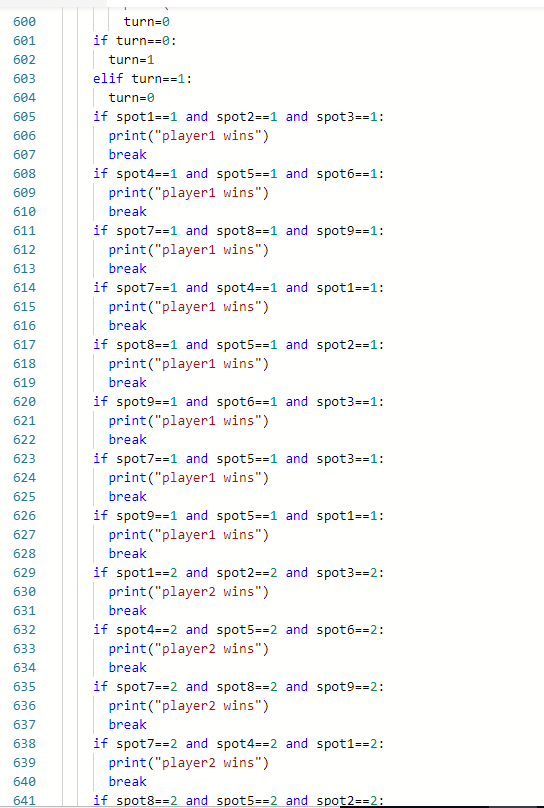


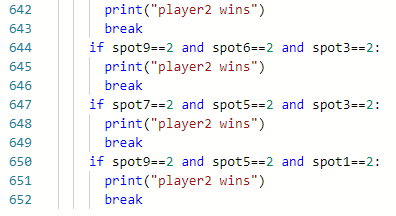












3. Explain how your program keeps track of the game board.

(Provide specific code references by line number.)

a. What python types and data structures are used?

The program stores data using variables and then changes the variables when the player moves in that position.

b. How are moves by player X and player O recorded?

They are recorded by changing the values of the variables to a certain value, 1 for player 1, and 2 for player 2.

c. How are free spaces recorded?

They are recorded when the value of the spot is 0, which means that nobody has gone in that position.

4. Explain how moves and commands are input from the console.

(Provide specific code references by line number.)

a. How does the player tell the program about the move location (row, column)?

They input an integer from the console indicating which move they desire.

b. How does the program verify that the move location is valid?

By checking if the input is an integer, and if it isn’t then it outputs an error message. If it is an integer above or below the values that were assigned, then an error message gets outputted.

c. How does the program verify that the space is free?

It verifies by checking if the variable value is 0.

d. What does the program do if there is something wrong with the move?

The program outputs an error message, and it is that player’s turn again.

5. Explain how the program keeps track of gameplay.

(Provide specific code references by line number.)

a. How does the program switch between player X and player O moves?

It changes the turn by switching the turn variable back and forth in line 601-604.

b. How does the program keep asking for moves?

It asks for moves by printing the line in line 69, 71, or 503.

c. How does the program decide when to stop asking for moves?

When there is a win, or a tie (line 78).

**Level 3: Basic Enhancements**

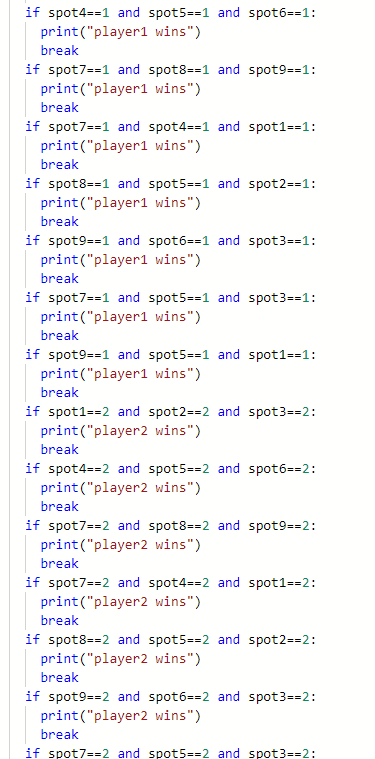
1. Explain, in plain words, a strategy for determining if player “x” or player “O” has won the game after a move is made.

It looks through the 8 possible combinations for each player, and then determines if any one matches. If none match, then they resume the program.

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



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



4. 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.

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

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

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