**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 done this.

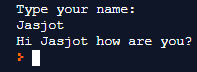
2. Type the following code into the white area of the IDE and run the program. Explain what you see in the black area of the IDE.

print("Type your name:")

name = input()

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

This is what I see:



What I see in this is that the code first prints: “Type our name”. Then, it waits for the user input. When I type my input, it prints the exact input in between “Hi” and “how are you?”.

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

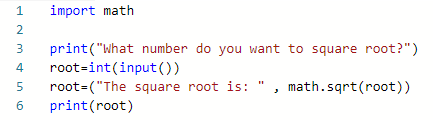
a. Uses the input() function to read a numerical value from the console.

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.)

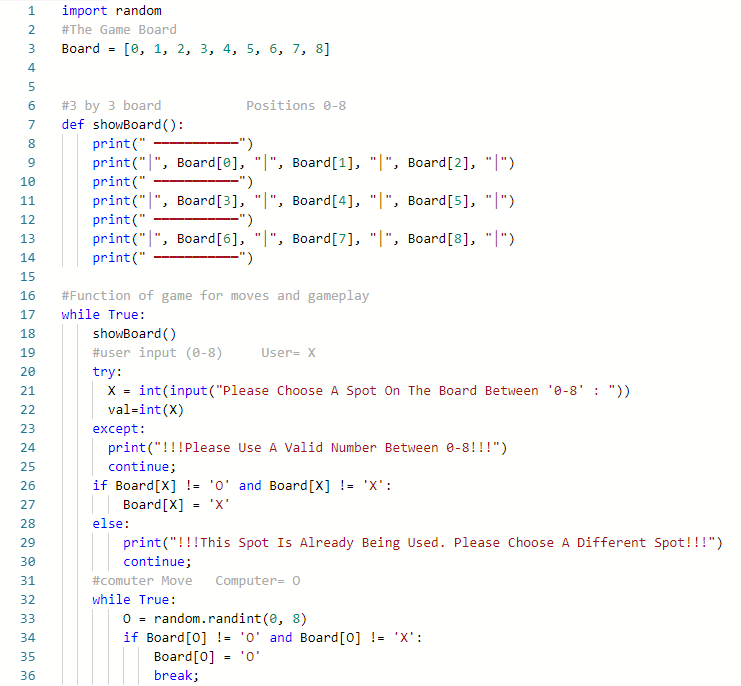
a. The program may be either player v. computer or player 1 v. player 2.

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?

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

c. How are free spaces recorded?

In line 3, I made a list named “Board” in my program to keep track of the game board. The list had indexes which were from 0-8 and I used these for my Tic Tac Toe board. I did this by making a function called “showBoard” (line 7) and it had indexes 0-2 at the top row (line 9), 3-5 at the middle row (line 11) and 6-8 at the bottom row (line 13). I did this so there was an area for the user and player to choose where they wanted to place their positions on the board. For the grid around the board, I had vertical as well as horizontal lines for the border as well as the rows and columns of the board. These lines are strings and they were printed in the “showBoard” function by using the command “print()” (lines 8-14). In the game, the user will be “X” while the computer will be “O”. The user will have to choose a number between 0-8 to begin and this will replace the respective index on the board with the “X” (lines 20-29). The computer is “O” and the place for this on the board is generated randomly. In the code, there is a random number generator and it will randomly generate a number between 0-8 while making sure that the place is not already taken. Then, it will replace that index with “O” on the board (lines 32-35). Any free spaces are shown on the board as numbers which don’t have an “X” or an “O” on them. This means that the user or the computer is able to go on that particular spot. But if that space is occupied by an “X” or an “O”, then no one will be able to put it in that spot (line 26-29 or 34-35).

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)?

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

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

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

The user has the choice of placing their “X” on an index from 0-8 on the board (line 20-22). When the player has chosen a number and wrote it, the number is then changed into an integer with the “int(input)()” (line 21). This allows it to be identified as an index of the game board. The number then goes through a check which is just the program checking if the number/index that is chosen by the user is a valid move. To do this, the program checks if that desired index is already an “X” or an “O” (line 26-29). If the program determines that the index is not already chosen, then the selected index becomes an “X”, with the help of the “if” condition (lines 26-27). If the index/spot is already used then the program prints “!!! Please Use A Valid Spot Between 0-8!!!” and the program asks the player for a different move (line 24). The move input by the user can only be valid if it is between “0-8”. If the number typed is bigger than 8 or smaller than 0, then the program prints that “!!!Please Use A Valid Number Between 0-8!!!” and it is still the user’s turn (lines 23-30). This is also the case if something is wrong with the move such as the input not being a number so the program does the same thing as above.

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?

b. How does the program keep asking for moves?

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

This program for the moves is in a “while loop” which is always true unless it is stopped with “break;” (line 36). This means that the moves switch between user “X” and the computer “O” and the loop keeps going. The loop starts when the user “X” is asked to choose a spot and then the computer “O” goes. After this, the loop restarts and user “X” goes again and then the computer “O”. The code that asks the user “X” for moves is in the loop so the player is asked every time the loop restarts (lines 17-36). The program stops when there are no moves left, or when there is invalid input. To keep track of the gameplay, the “showBoard()” function is placed in the loop and this shows the board after each move (line 18).

**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.

The strategy that is made for determining if player “X” and player “O” has won the game after a move is by having win combinations. To determine a winner, the board must be checked for all winning combinations. If the spots on the board are numbered 0-8, with 0 being the top left and 8 being the bottom right and the numbers are increasing across the board. Then there are 3 ways to win:

1) Have 3 characters in a row for any row on the board and the 3 rows are (0, 1, 2), (3, 4, 5) and (6, 7, 8).

2) Have 3 characters in a column for any column on the board and the 3 columns are (0, 3, 6), (1, 4, 7) and (2, 5, 9).

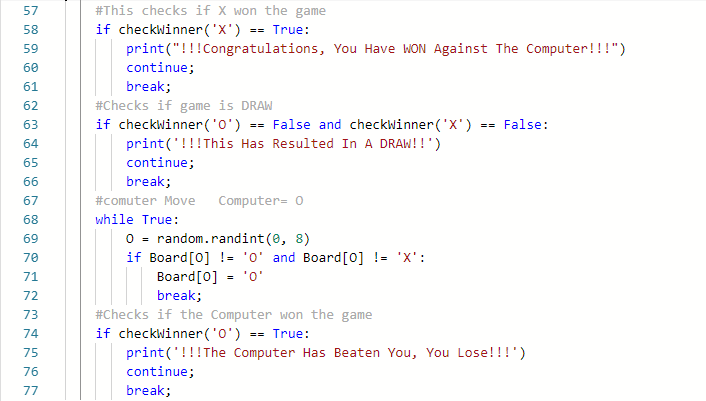
3) Have 3 characters in a diagonal line for any diagonal line on the board and the diagonal combinations are (0, 4, 8) and (2, 4, 6).

With this information, we can determine strategy for determining a winner. To do this we need to create a function to look at the game board after each move. The function must check all the possible winning combinations listed above, and the combinations must only contain one character. If there is a winning combination, the function should end the game and a result should appear. This function must be in use after every move.

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.

I have done this.

**Level 4: AI Enhancements**

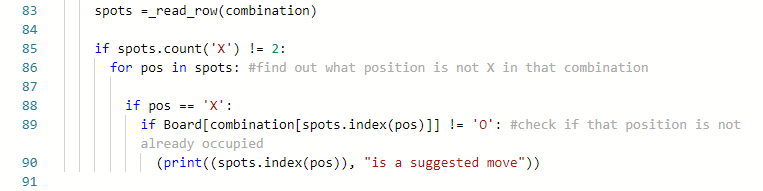
1. Explain, in plain words, a strategy for suggesting the best move for player “x” or player “O” to make when it is their turn to move.

For this Tic Tac Toe game code, in-order to suggest the best move for user “X” is by implementing spots =\_read\_row. This combination suggests that the user “X” should choose this spot as their next move. If “X” has filled at least 2 spots and the last spot is not used, a message would appear saying “(integer) is a suggested move”. The code with “If Board [combination[spots.index(pos)]]!= ‘0’” helps to strengthen the suggestions to as the suggested index can be printed on the screen. This suggestion depends on which spots the user “X” puts “X” on the board.

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.