**Objective:**

* Demonstrate your Python programming skills by designing and implementing a Tic Tac Toe game.

**Level 1: Designing the Game Parts**

1. Define three global constants representing the possible player moves.
   1. Define PlayerX as the string “X”
   2. Define PlayerO as the string “O”
   3. Define PlayerNone as the string “ “ (blank space)
2. Design the game board using the Python list data type.
   1. Decide how to identify (number) each square in the board. (e.g. By row & column or by sequential numbering.)
   2. Create a global variable using a Python list to store the contents of your board using your numbering sequence.
   3. Initialize the board to the starting empty state. (e.g. Each square containing “ “ PlayerNone.)
   4. Provide a listing of your code to define the game board.
3. Define and implement a function to print your game board to the output console.
   1. Provide a listing of your function code.

**Level 2: Implementing a Basic Game**

1. Design and implement code for PlayerX moves.
   1. Print a prompt to ask PlayerX to make a move
   2. Read the response form the console input
   3. Add the PlayerX move to the game board
   4. Print out the new game board state
2. Design and implement code for PlayerO moves.
   1. Print a prompt to ask PlayerO to make a move
   2. Read the response form the console input
   3. Add the PlayerO move to the game board
   4. Print out the new game board state
3. Create a loop to make PlayerX and PlayerO moves
   1. Add the PlayerX and PlayerO move code to your loop
   2. Stop the game if someone types “stop”
   3. Stop the game if 9 moves (i.e. all moves) have been made
4. Provide a listing of your basic game code.
   1. Demo your basic game to Mr. Nestor

**Level 3: Completing Your Game**

1. Define and implement a function to make sure a player move is valid.
   1. Make sure that the move is for a square that exists in your game board definition.
   2. Make sure that the square is empty (i.e. Has not already been used by a player.)
   3. Return Boolean “true” if the game is valid and “false” if it is not valid.
   4. Provide a listing of your function code.
2. Modify your PlayerX and PlayerO code to check for a valid move.
   1. Use the function your created above
   2. Ask the player to “Try again” if the move is not valid
   3. Keep looping until the player enters a valid or types “stop”
3. Provide a listing of your completed game code.
   1. Demo your completed game to Mr. Nestor

**Level 4: Extending Your Game – Win Check**

1. Extend your game code to:
   1. Define and implement a function to check for winning conditions
   2. Stop the game when either PlayerX or PlayerO wins.
   3. Print a “You Win!” message to the console output
2. Provide a listing of your completed game code.
3. Demo your completed game to Mr. Nestor

**Level 4+: Extending Your Game – PlayerO AI**

1. Extend your game code to replace a human PlayerO with an   
   artificial intelligence (AI) player :
   1. The AI just needs to make valid moves.
   2. The AI does not have to make the “best” move
2. Provide a listing of your completed game code.
3. Demo your completed game to Mr. Nestor

**Level 4++: 3D TicTacToe**

1. Extend your code to implement a 3D version of your Tic Tac Toe game.
2. Provide a listing of your completed game code.
3. Demo your completed game to Mr. Nestor