# PYTHON MINI PROJECT – BUILD A TIC TAC TOE GAME FOR 2 HUMAN PLAYERS

## 100 Points

## Due: April 8 2021

**Objective:** To create a Tic Tac Toe game using Python where 2 users can place "X" and "O" in a 3*3 board. Here is the google game : https://www.google.com/search?q=tic+tac+toe

**RULES:**

1. 2 players should be able to play the game (both sitting at the same computer)
2. The board should be printed out every time a player makes a move
3. You should be able to accept input of the player position and then place a symbol on the board.

## Rubric:

Step 1 – 9 – **63 Points ( 7 Points for each step)**

Step 10 – **12 Points**

Comments throughout the Program – **5 Points**

Fully functional game for 2 players – **20 Points**

Please note that completing this assignment stepwise will give you points. You can also demo this program to the TA.

**To submit in Canvas:** Python file (.py or .ipynb) with all the above steps detailed.

**Note:** To receive step marks for each part of the assignment, your assignment need to have enough documentation/ details related to each step. Please submit your final Python file accordingly.

**Overview:**

- You will use the numpad to match numbers to the grid on a tic tac toe board:

| | | |
|---|---|---|
| **7** | **8** | **9** |
| **4** | **5** | **6** |
| **1** | **2** | **3** |

- Use a function to ask the user "Where would you like to place your X or O". And based on the user input (number), from the first user, X will be placed on that number and when the second user inputs a number, O will be placed in there.

- To take input from a user:
  player1 = input("Please pick a marker 'X' or 'O'")
- Note that input() takes in a string. If you need an integer value, use
  position = int(input('Please enter a number'))
- To clear the screen between moves:
  from IPython.display import clear_output
  clear_output()
- Note that clear_output() will only work in jupyter. To clear the screen in other IDEs, consider: print('\n'*100)
- This scrolls the previous board up out of view.
- Use the walkthrough_steps guide below for you to help guide you along with the functions you will need to create.

**Walk Through Steps:**

1. **Step 1:** Write a function that can print out a board. Set up your board as a list, where each index 1-9 corresponds with a number on a number pad, so you get a 3 by 3 board representation.

```
from IPython.display import clear_output

def display_board(board):
    clear_output()  # Remember, this only works in jupyter!

    print('   |   |')
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
    -------- #use print for creating rest of the board
    print('   |   |')
```

**TEST Step 1:** run your function on a test version of the board list, and make adjustments as necessary. Below is an example of test function for the board

```
test_board = ['#','X','O','X','O','X','O','X','O','X']
display_board(test_board)
```

2. **Step 2:** Write a function that can take in a player input and assign their marker as 'X' or 'O'. Think about using ***while*** loops to continually ask until you get a correct answer.

```
def player_input():
    marker = ''

    while not (marker == 'X' or marker == 'O'):
        marker = input('Player 1: Do you want to be X or O?

    if marker == 'X':
        return ()
    else:
        return ()
    #edit this function to return the correct output
```

**TEST Step 2:** run the function to make sure it returns the desired output

```
In [5]: player_input()
        Player 1: Do you want to be X or O? O
Out[5]: ('O', 'X')
```

3. **Step 3:** Write a function that takes in the board list object, a marker ('X' or 'O'), and a desired position (number 1-9) and assigns it to the board.

```
In [ ]: def place_marker(board, marker, position):
            #write your function here
```

**TEST Step 3:** run the place marker function using test parameters and display the modified board

```
In [16]: place_marker(test_board,'$',7)
         display_board(test_board)
             |   |
          $  | $ | X
             |   |
         -----------
             |   |
          O  | X | O
             |   |
         -----------
             |   |
          X  | O | X
```

4. **Step 4:** Write a function that takes in a board and checks to see if someone has won.

```
def win_check(board,mark):

    return # this line of code if for across the top
           ((board[7] == mark and board[8] == mark and board[9] == mark)
            or  # across the middle
            or  # across the bottom
            or # down the middle
            or # down the middle
            or # down the right side
            or # diagonal
            or # diagonal
```

**TEST Step 4:** run the win_check function against our test_board - it should return True

```
In [9]: win_check(test_board,'X')
Out[9]: True
```

5. **Step 5:** Write a function that uses the random module to randomly decide which player goes first. You may want to lookup random.randint() Return a string of which player went first.

```
In [ ]: import random

        def choose_first():
            #write your if construct here
```

6. **Step 6:** Write a function that returns a Boolean indicating whether a space on the board is freely available.

```
In [ ]: def space_check(board, position):

            return #write your function
```

7. **Step 7:** Write a function that checks if the board is full and returns a boolean value. True if full, False otherwise

```
In [ ]: def full_board_check(board):
            for #specify your iterable object here
                if #check the condition
                    return False
            return True
```

8. **Step 8:** Write a function that asks for a player's next position (as a number 1-9) and then uses the function from step 6 to check if its a free position. If it is, then return the position for later use.

```
def player_choice(board):
    position = 0

    while position not in [1,2,3,4,5,6,7,8,9] or not #function for space check:
        #write your program here

    return position
```

9. **Step 9:** Write a function that asks the player if they want to play again and returns a Boolean True if they do want to play again.

```
In [ ]: def replay():

            return input #give the string
```

10. **Step 10:** Here comes the hard part! Use while loops and the functions you've made to run the game!

```
print('Welcome to Tic Tac Toe!')

#while True:
    # Set the game up here
    #pass

    #while game_on:
        #Player 1 Turn


        # Player2's turn.

            #pass

    #if not replay():
        #break
```