Milestone Project 1: Walkthrough Steps Workbook

Below is a set of steps for you to follow to try to create the Tic Tac Toe Milestone Project game!

Some suggested tools before you get started:

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. Now on to the program!

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.

In [ ]:

```
from IPython.display import clear_output

def display_board(board):
    clear_output()
    print('['+board[7]+']['+board[8]+']['+board[9]+']')
    print('['+board[4]+']['+board[5]+']['+board[6]+']')
    print('['+board[1]+']['+board[2]+']['+board[3]+']')
```

TEST Step 1: run your function on a test version of the board list, and make adjustments as necessary

In [ ]:

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

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.

In [ ]:

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

    while marker !='x' and marker !='o':
        marker = input('Player 1: Choose marker x or o : ').lower()

    if marker == 'x':
        return ('x','o')
    else:
        return ('o','x')

    return (player1,player2)
```

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

In [ ]:

```
player1_marker, player2_marker = player_input()
```

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 [ ]:

```
player2_marker
```

In [ ]:

```
def place_marker(board, marker, position):
    board[position] = marker
    return board
```

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

In [ ]:

```
test_board = place_marker(test_board,'x',9)
display_board(test_board)
```

Step 4: Write a function that takes in a board and a mark (X or O) and then checks to see if that mark has won.

In [ ]:

```
def win_check(board, mark):

    return ((board[1]==board[2]==board[3]==mark) or
            (board[4]==board[5]==board[6]==mark) or
            (board[7]==board[8]==board[9]==mark) or
            (board[1]==board[4]==board[7]==mark) or
            (board[2]==board[5]==board[8]==mark) or
            (board[3]==board[6]==board[9]==mark) or
            (board[1]==board[5]==board[9]==mark) or
            (board[3]==board[5]==board[7]==mark))
```

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

In [ ]:

```
win_check(test_board,'x')
```

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 [2]:

```
import random

def choose_first():
    if random.randint(1,2) == 1:
        return 'Player 1'
    else:
        return 'Player 2'
```

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 board[position] == ' '
```

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 x in range(1,10):
        if space_check(board,i):
            return False

    return True
```

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 it's a free position. If it is, then return the position for later use.

In [ ]:

```
def player_choice(board):
    position_available = False
    while position_available == False:
        position = int(input('Enter the board position [1-9] for your marker : '
))
        position_available = space_check(board,position)

    return position
```

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():
    play_again = input('Would you like to play the game again (y/n) : ')
    return play_again == 'y'
```

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

In [ ]:

```
from IPython.display import clear_output
import random

def display_board(board):
    clear_output()
    print('['+board[7]+']['+board[8]+']['+board[9]+']')
    print('['+board[4]+']['+board[5]+']['+board[6]+']')
    print('['+board[1]+']['+board[2]+']['+board[3]+']')

def player_input():
    marker = ''
    while marker !='x' and marker !='o':
        marker = input('Player 1: Choose marker x or o : ').lower()
    if marker == 'x':
        return ('x','o')
    else:
        return ('o','x')
    return (player1,player2)

def place_marker(board, marker, position):
    board[position] = marker
    return board

def win_check(board, mark):

    return ((board[1]==board[2]==board[3]==mark) or
            (board[4]==board[5]==board[6]==mark) or
            (board[7]==board[8]==board[9]==mark) or
            (board[1]==board[4]==board[7]==mark) or
            (board[2]==board[5]==board[8]==mark) or
            (board[3]==board[6]==board[9]==mark) or
            (board[1]==board[5]==board[9]==mark) or
            (board[3]==board[5]==board[7]==mark))

def choose_first():
    if random.randint(1,2) == 1:
        return 'Player 1'
    else:
        return 'Player 2'

def space_check(board, position):
    return board[position] == ' '

def full_board_check(board):
    for x in range(1,10):
        if space_check(board,x):
            return False
    return True

def player_choice(board):
    position_available = False
    while position_available == False:
        position = int(input('Enter the board position [1-9] for your marker : '
))
        position_available = space_check(board,position)
    return position

def replay():
    play_again = input('Would you like to play the game again (y/n) : ')
    return play_again == 'y'


print('Welcome to Tic Tac Toe!')

while True:
    #Set the game up here
    the_board = ['#',' ',' ',' ',' ',' ',' ',' ',' ',' ']
    display_board(the_board)

    player1_marker, player2_marker = player_input()

    turn = choose_first()
    print(turn+' will go first !')

    play_game = input('Ready to play ? (y/n) : ')
    if play_game == 'y':
        game_on = True
    else:
        game_on = False

    while game_on:

        if turn == 'Player 1':
            display_board(the_board)
            position = player_choice(the_board)
            place_marker(the_board,player1_marker,position)

            if win_check(the_board,player1_marker):
                display_board(the_board)
                print('Player 1 has won !')
                game_on = False
            else:
                if full_board_check(the_board):
                    display_board(the_board)
                    print('Tie Game !')
                    game_on = False
                else:
                    turn = 'Player 2'
        else:
            display_board(the_board)
            position = player_choice(the_board)
            place_marker(the_board,player2_marker,position)

            if win_check(the_board,player2_marker):
                display_board(the_board)
                print('Player 2 has won !')
                game_on = False
            else:
                if full_board_check(the_board):
                    display_board(the_board)
                    print('Tie Game !')
                    game_on = False
                else:
                    turn = 'Player 1'
    if not replay():
        break
```

```
[ ][ ][ ]
[ ][ ][ ]
[ ][ ][ ]
```

Good Job!

In [ ]:

```

```