Tic Tac Toe development

Here are the various different versions of my code along with how I created them. Most of the syntax and logical errors are listed here as well

Version one

```
import random, time
win = False
board = [" ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", ]
player = True
```

These first lines of code are some of the main variables that I will use for the game
The random library will be used to determine who goes first. The time library will be used to give the program a more natural feel.

I will use the win variable to determine if the game is over. True will mean that a player has scored 3 in a row.

The board variable acts a place holder to store where the Xs and Os will be.

Lastly, the player variable determines who's go it is. I will use True to signify X's go and False to signify O's go.

```
while not win:
    choice = int(input("please enter a space"))
```

Here I am using a while loop to see if the game is still running. If the game is not over then a new input will be taken for the next players go.

```
    if(board[choice]) == " ":
                         ^
SyntaxError: invalid syntax

Process finished with exit code 1
```

Originally I had a syntax error here because I did not use the correct amount of brackets (I only used one instead of two)

```
    if(board[choice]) == " ":
        if player:
            board[choice] = "X"
            player = False
        elif not player:
            board[choice] = "O"
            player = True
    else:
        print("space taken try again")
```

This next section of code places the users choice onto the board.

The first line of code checks to see if the players chosen square has already been taken by checking the contents of that box.

If the box is empty then an X or an O is placed in the box depending on whose go it is. Then the players go is changed by changing the value of player accordingly.

The else statement will only get run if the users chosen box has already been taken. It asks them to try again.

```
def layout():
    print("    |    |    ")
    print(" "+board[1]+" | "+board[2]+" |  "+board[3]+"  ")
    print("---|---|---")
    print(" "+board[4]+" | "+board[5]+" |  "+board[6]+"  ")
    print("---|---|---")
    print(" "+board[7]+" | "+board[8]+" |  "+board[9]+"  ")
    print("    |    |    ")
```

Here is the section of code that prints the board layout. This is just a simple ascii table that prints the values of the board places in the boxes. By default this is a space. The board it creates looks like this

```
    |   |
 X  | O |  X
---|---|---
   | O |
---|---|---
 X  | O |
    |   |
please enter a space
```

In the current version there is no win condition but players can see if they have won by simply looking at the board. There is also no way to restart the game without closing the console

Finally I added in a small sleep after the turn is added to the board to make the program flow more natural and flow better

```
time.sleep(0.3)
layout()
```

Here is the completed code for the first version.

```
import random, time
win = False
board = [" ", " ", " ", " ", " ", " ", " ", " ", " ", " ", ]
player = True


def layout():
    print("   |   |    ")
    print(" "+board[1]+" | "+board[2]+" |  "+board[3]+"  ")
    print("---|---|---")
    print(" "+board[4]+" | "+board[5]+" |  "+board[6]+"  ")
    print("---|---|---")
    print(" "+board[7]+" | "+board[8]+" |  "+board[9]+"  ")
    print("   |   |    ")


layout()
while not win:
    choice = int(input("please enter a space"))
    if(board[choice]) == " ":
        if player:
            board[choice] = "X"
            player = False
        elif not player:
            board[choice] = "O"
            player = True
    else:
        print("\n space taken try again")

    time.sleep(0.1)
    layout()
```

Version Two

In this version of the code I aim to add a message that is displayed when somebody wins. I also want to add in some code to end the game when it is over.

In order for the game to end, a win condition must be met. In noughts and crosses this can be represented by certain combinations being filled on the board. In total there are 8 win conditions (3 horizontal 3 vertical and 2 diagonal). I wanted to use an if statement to check if a win condition has been met because I could not figure out how to do it with for loops.

```
if board[1] == "X" and board[2] == "X" and board[3] == "X":

        print(" \n X WINS!")
        win = True
```

This if statement checks to see if the first three boxes are filled. If they are, then the game must have been won so this gets printed and win becomes True which ends the game.

```
if board[1] == "X" and board[2] == "X" and board[3] == "X" or \
    board[4] == "X" and board[5] == "X" and board[6] == "X" or \
    board[7] == "X" and board[8] == "X" and board[9] == "X" or \
    board[1] == "X" and board[4] == "X" and board[7] == "X" or \
    board[2] == "X" and board[5] == "X" and board[8] == "X" or \
    board[3] == "X" and board[6] == "X" and board[9] == "X" or \
    board[1] == "X" and board[5] == "X" and board[9] == "X" or \
    board[3] == "X" and board[5] == "X" and board[7] == "X":
        print(" \n X WINS!")
        win = True
```

I then used this same method to check the other 7 win conditions. I then repeated these same statements to see if Os has won.

```
elif board[1] == "O" and board[2] == "O" and board[3] == "O" or \
    board[4] == "O" and board[5] == "O" and board[6] == "O" or \
    board[7] == "O" and board[8] == "O" and board[9] == "O" or \
    board[1] == "O" and board[4] == "O" and board[7] == "O" or \
    board[2] == "O" and board[5] == "O" and board[8] == "O" or \
    board[3] == "O" and board[6] == "O" and board[9] == "O" or \
    board[1] == "O" and board[5] == "O" and board[9] == "O" or \
    board[3] == "O" and board[5] == "O" and board[7] == "O":

        print(" \n O WINS!")
        win = True
```

During this process I came across a syntax error. I did not place the backslash at the end of the line to force the condition to be passed onto the next line.

```
if board[1] == "X" and board[2] == "X" and board[3] == "X" or
    board[4] == "X" and board[5] == "X" and board[6] == "X" or
```

```
    if board[1] == "X" and board[2] == "X" and board[3] == "X" or
                                                               ^
SyntaxError: invalid syntax

Process finished with exit code 1
```

The program threw a syntax error since it could not end the if condition.

Here is the completed code for version two of the program.

```
import random, time
win = False
board = [" ", " ", " ", " ", " ", " ", " ", " ", " ", " ", ]
player = True

def layout():
    print("   |   |    ")
    print(" "+board[1]+" | "+board[2]+" |  "+board[3]+"  ")
    print("---|---|---")
    print(" "+board[4]+" | "+board[5]+" |  "+board[6]+"  ")
    print("---|---|---")
    print(" "+board[7]+" | "+board[8]+" |  "+board[9]+"  ")
    print("   |   |    ")

layout()
while not win:
    choice = int(input("please enter a space"))
    if(board[choice]) == " ":
        if player:
            board[choice] = "X"
            player = False
        elif not player:
            board[choice] = "O"
            player = True
    else:
        print("\n space taken try again")

    time.sleep(0.1)
    layout()
    if board[1] == "X" and board[2] == "X" and board[3] == "X" or \
        board[4] == "X" and board[5] == "X" and board[6] == "X" or \
        board[7] == "X" and board[8] == "X" and board[9] == "X" or \
        board[1] == "X" and board[4] == "X" and board[7] == "X" or \
        board[2] == "X" and board[5] == "X" and board[8] == "X" or \
        board[3] == "X" and board[6] == "X" and board[9] == "X" or \
        board[1] == "X" and board[5] == "X" and board[9] == "X" or \
        board[3] == "X" and board[5] == "X" and board[7] == "X":
            print(" \n X WINS!")
            win = True

    elif board[1] == "O" and board[2] == "O" and board[3] == "O" or \
        board[4] == "O" and board[5] == "O" and board[6] == "O" or \
        board[7] == "O" and board[8] == "O" and board[9] == "O" or \
        board[1] == "O" and board[4] == "O" and board[7] == "O" or \
        board[2] == "O" and board[5] == "O" and board[8] == "O" or \
        board[3] == "O" and board[6] == "O" and board[9] == "O" or \
        board[1] == "O" and board[5] == "O" and board[9] == "O" or \
        board[3] == "O" and board[5] == "O" and board[7] == "O":

            print(" \n O WINS!")
            win = True
```

Version 3

In version 3 of the program I aim to make the program more user friendly by only allowing valid inputs as well as adding more options when to restart the game. I also want to add the ability to check if the game has been drawn.

```
win = False
board = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", ]
player = True
counter = 0
```

Here I have changed the board layout to that it starts with numbers that are easier to understand than blank spaces. The counter variable is used to determine if there is a draw.

```
while not win:
    user_input = input("Enter a number from 1-9: ")
    match = re.match("^[1-9]$", user_input)
```

There is now a more robust input validation method that uses a regex range to test if the given input was a range from 1-9.

```
    if board[choice] != user_input:
        print("No")
        continue
```

If the board space has already been taken, The word No will be printed to the console. This is tested by the user input not being the same as the available position

```
    counter += 1
```

When a go has been successfully made, the counter variable increases to test for a draw.

```
elif counter == 9:
    print("Its a Draw")
    win = True
```

When there have been 9 successful goes the game must have been a draw. The game then end even if there is no winner

Here is the completed code for version 3

```
import re, time
#imports the necessary labraries needed. Regex for input validation and time for fluidity

win = False
board = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", ]
player = True
counter = 0
#main variables used in the program. Win is a bollean used to determine if the game is ended. board is a list that is the map for the grid of a game. it is global.
#counter is a stepper variabke used to tell if the game is a draw by counting to 9 sucessful goes.
#player is used to determine who's go it is. If player is true, then it is x's turn.


#function used to print out the board onto the terminal. There are no parameters. List undexes are used to represent each point on the grid
def layout():
    print("   |   |    ")
    print(" " + board[1] + " | " + board[2] + " |  " + board[3] + "  ")
    print("---|---|---")
    print(" " + board[4] + " | " + board[5] + " |  " + board[6] + "  ")
    print("---|---|---")
    print(" " + board[7] + " | " + board[8] + " |  " + board[9] + "  ")
    print("   |   |   ")



layout()

while not win:
    user_input = input("Enter a number from 1-9: ")
    match = re.match("^[1-9]$", user_input)
#as long as the game is still running the game needs an input from 1-9. match determines whether the input in within the given regex range.
#this will keep running until the game is over

    if match:
        choice = int(user_input)
#sets the user choice to the validated input. User input is now garenteed to work as it is an int

        if board[choice] != user_input:
            print("No")
            continue
#this will print no if the spot has already been taken as the user input cannot congregate with the board place
        if player:
            board[choice] = "X"
            player = False
            counter += 1
            #On player X's go: set the coordinate to the letter X, Set the game to O's turn and add one to the draw counter
        elif not player:
            board[choice] = "O"
            player = True
            counter += 1
            #on player O's turn: ser the coordinate to the letter O, Set the game to X's turn and add one to the draw counter
```

```
else:
    print("\n please only enter a number for a valid space")
#this will be run when an error occurs and an incorrect number is given e.g. 56
time.sleep(0.1)
layout()
#wait a short time and then reprint the new board with the new turn


if board[1] == "X" and board[2] == "X" and board[3] == "X" or \
    board[4] == "X" and board[5] == "X" and board[6] == "X" or \
    board[7] == "X" and board[8] == "X" and board[9] == "X" or \
    board[1] == "X" and board[4] == "X" and board[7] == "X" or \
    board[2] == "X" and board[5] == "X" and board[8] == "X" or \
    board[3] == "X" and board[6] == "X" and board[9] == "X" or \
    board[1] == "X" and board[5] == "X" and board[9] == "X" or \
    board[3] == "X" and board[5] == "X" and board[7] == "X":
    print(" \n X WINS!")
    win = True
    #these are all the win conditions for X. If any of them are met the the game ends and X is the winner.

elif board[1] == "O" and board[2] == "O" and board[3] == "O" or \
    board[4] == "O" and board[5] == "O" and board[6] == "O" or \
    board[7] == "O" and board[8] == "O" and board[9] == "O" or \
    board[1] == "O" and board[4] == "O" and board[7] == "O" or \
    board[2] == "O" and board[5] == "O" and board[8] == "O" or \
    board[3] == "O" and board[6] == "O" and board[9] == "O" or \
    board[1] == "O" and board[5] == "O" and board[9] == "O" or \
    board[3] == "O" and board[5] == "O" and board[7] == "O":
    print(" \n O WINS!")
    win = True
    #These are the win conditions for O. If any of them are met, the game ends and O becomes the winner.


elif counter == 9:
    print("Its a Draw")
    win = True
#If the counter variable Hits nine and no win criteria are met, the game must be a draw. The game then ends as win is now True.
```