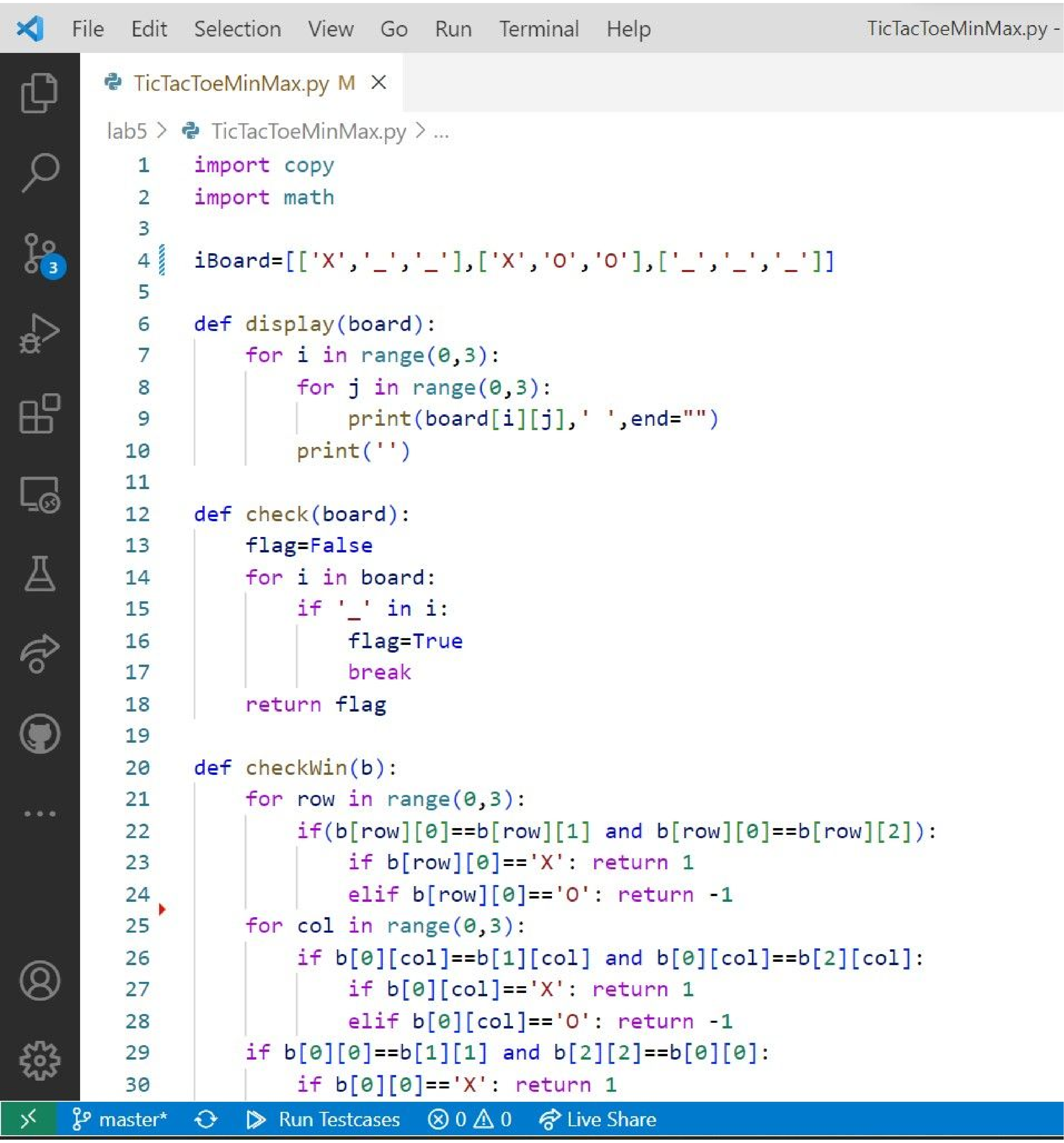

File Edit Selection View Go Run Terminal Help
TicTacToeMinMax.py -
TicTacToeMinMax.py M
lab5 > TicTacToeMinMax.py > ...
master*
Run Testcases
0 0
Live Share

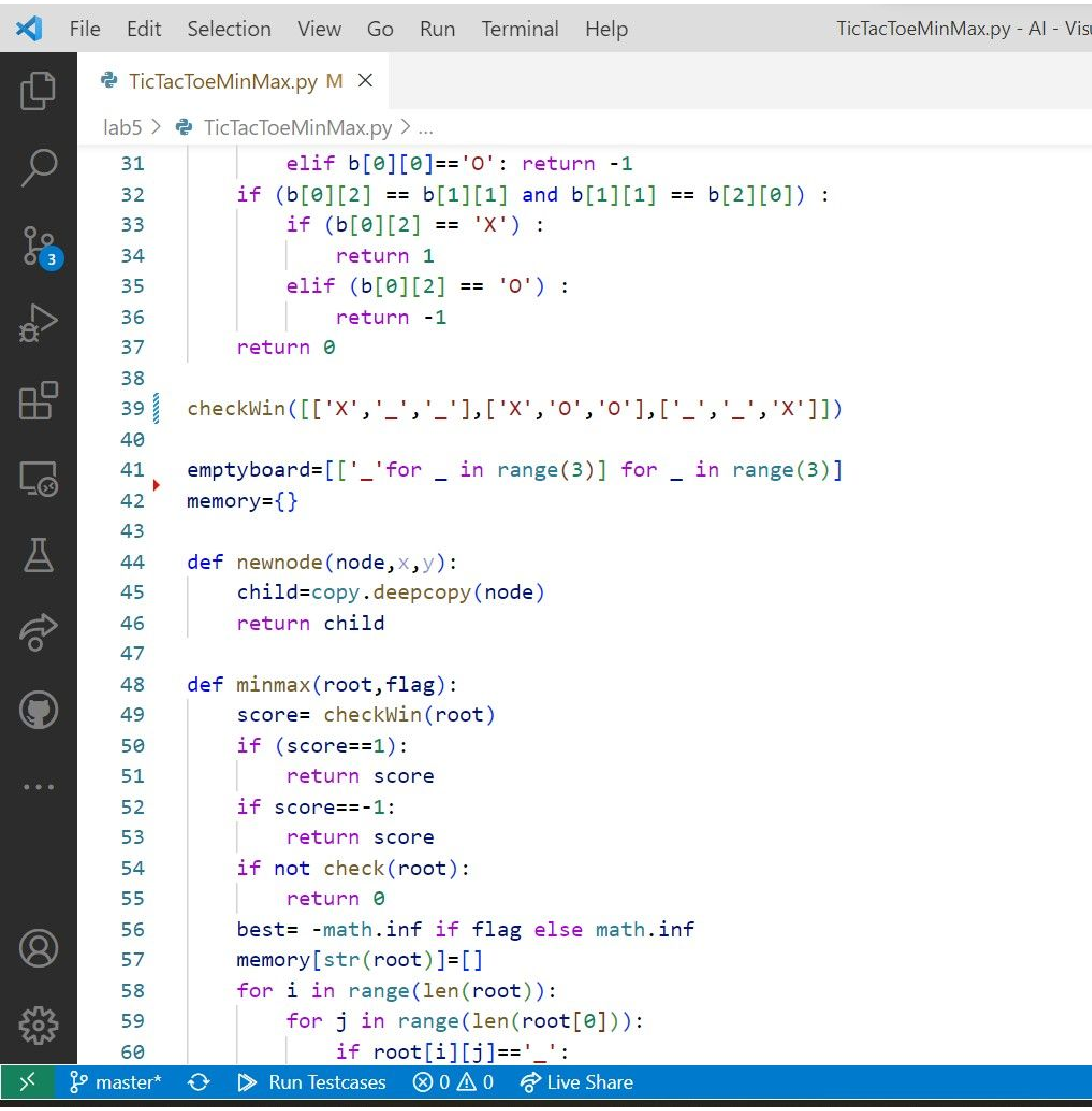
```python
import copy
import math

iBoard=[['X','_','_'],['X','O','O'],['_','_','_']]

def display(board):
    for i in range(0,3):
        for j in range(0,3):
            print(board[i][j],' ',end="")
        print('')

def check(board):
    flag=False
    for i in board:
        if '_' in i:
            flag=True
            break
    return flag

def checkWin(b):
    for row in range(0,3):
        if(b[row][0]==b[row][1] and b[row][0]==b[row][2]):
            if b[row][0]=='X': return 1
            elif b[row][0]=='O': return -1
    for col in range(0,3):
        if b[0][col]==b[1][col] and b[0][col]==b[2][col]:
            if b[0][col]=='X': return 1
            elif b[0][col]=='O': return -1
    if b[0][0]==b[1][1] and b[2][2]==b[0][0]:
        if b[0][0]=='X': return 1
```



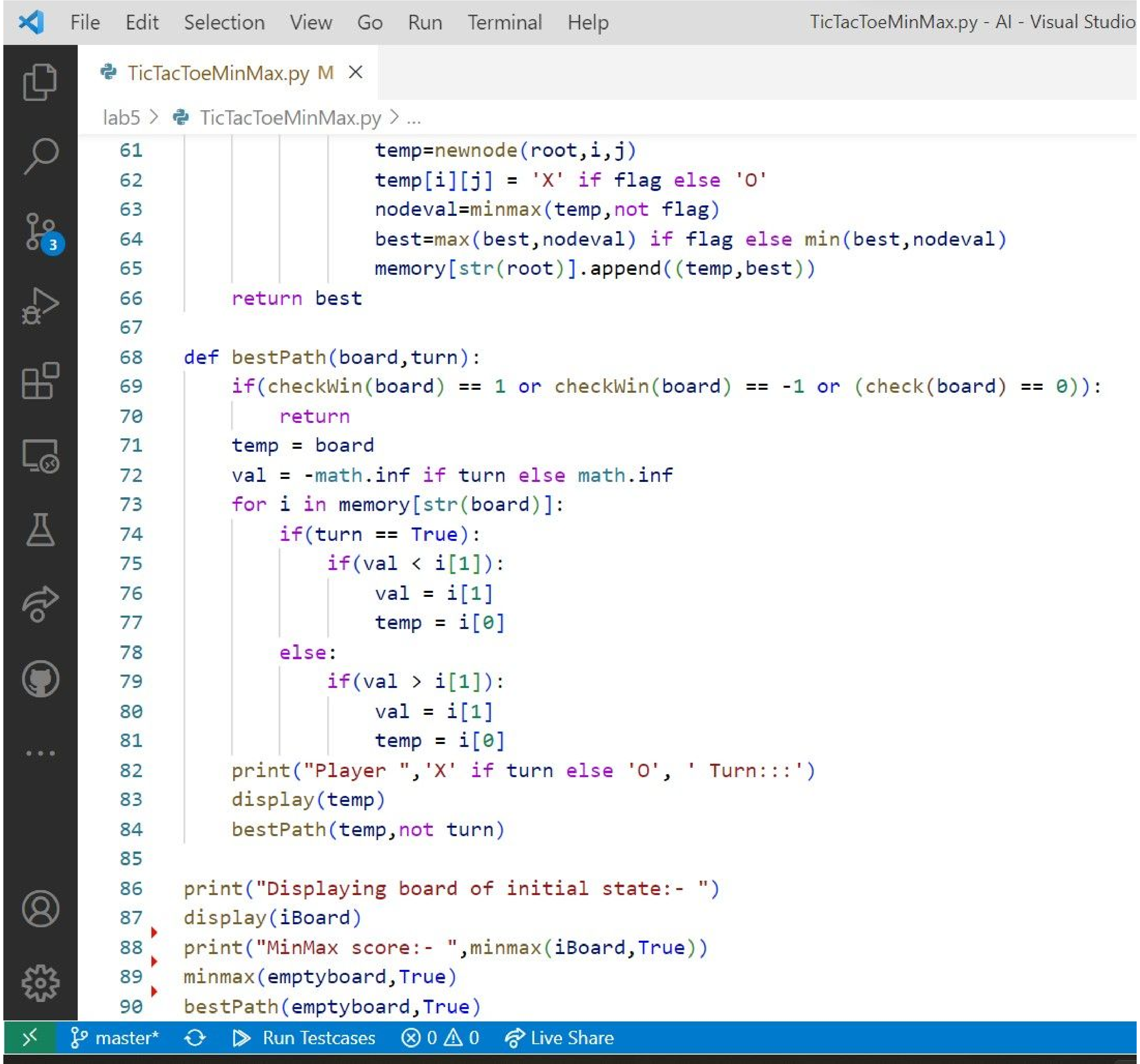
```python
        elif b[0][0]=='O': return -1
    if (b[0][2] == b[1][1] and b[1][1] == b[2][0]) :
        if (b[0][2] == 'X') :
            return 1
        elif (b[0][2] == 'O') :
            return -1
    return 0

checkWin([['X','_','_'],['X','O','O'],['_','_','X']])

emptyboard=[['_'for _ in range(3)] for _ in range(3)]
memory={}

def newnode(node,x,y):
    child=copy.deepcopy(node)
    return child

def minmax(root,flag):
    score= checkWin(root)
    if (score==1):
        return score
    if score==-1:
        return score
    if not check(root):
        return 0
    best= -math.inf if flag else math.inf
    memory[str(root)]=[]
    for i in range(len(root)):
        for j in range(len(root[0])):
            if root[i][j]=='_':
```


File Edit Selection View Go Run Terminal Help
TicTacToeMinMax.py - AI - Visual Studio
TicTacToeMinMax.py M
lab5 > TicTacToeMinMax.py > ...
master*
Run Testcases
0 0
Live Share

```python
                    temp=newnode(root,i,j)
                    temp[i][j] = 'X' if flag else 'O'
                    nodeval=minmax(temp,not flag)
                    best=max(best,nodeval) if flag else min(best,nodeval)
                    memory[str(root)].append((temp,best))
    return best

def bestPath(board,turn):
    if(checkWin(board) == 1 or checkWin(board) == -1 or (check(board) == 0)):
        return
    temp = board
    val = -math.inf if turn else math.inf
    for i in memory[str(board)]:
        if(turn == True):
            if(val < i[1]):
                val = i[1]
                temp = i[0]
        else:
            if(val > i[1]):
                val = i[1]
                temp = i[0]
    print("Player ",'X' if turn else 'O', ' Turn:::')
    display(temp)
    bestPath(temp,not turn)

print("Displaying board of initial state:- ")
display(iBoard)
print("MinMax score:- ",minmax(iBoard,True))
minmax(emptyboard,True)
bestPath(emptyboard,True)
```