```
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
# O'neillo Program
# Dahir Abib
# Programming Fundamentals
# 15/04/2021
# Assignment 2
#
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
#Library import

# @@ = X
# "" = O


import csv
import sys
import tkinter as tk


def main():
    menu()

#printing the menu below with if else statements
#\n = new line implented

def menu():
    print("\n:::Welcome to O'Neillo Game")
    print("============================")
    print()

    choice = input("""
                        1: New Game
                        2: Restore a game
                        3: Quit

                        Select a number from the menu : """)

    if choice == "1" or choice == "one":
        register()
        print( ":: A new Game::")

        name = input("\nInsert the first player name: ")

        name2 = input("Insert the second player name: ")
#below I will be concantenating the two variables above
        print("Hi",name, "You are ''")
        print("Hi",name2," you are @@")
        print("Lets play")
    elif choice == "2" or choice == "two":
        login()
    elif choice == "3" or choice == "three":
        sys.exit
    else:
        print("You must only select either 1 or 2")
        print("Please try again")
        menu()

    import random
    import sys

    def drawBoard(board):
```

```
        # This function prints out the board that it was passed.
        HLINE1 = '   ---------------------------------------'
        VLINE1 = '''
          | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 |'
          | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |'
          | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
          | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
          | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
          | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
          | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 |
          | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 |'''


#this function prints off the grid and displays all the numbers in tge 8x8
grid
        print(HLINE1)
        for y in range(8):
            print(VLINE1)
            print(y + 1, end=' ')
            for x in range(8):
                print('| %s' % (board[x][y]), end=' ')
            print('|')
            print(VLINE1)
            print(HLINE1)
#this functions reset it to its default position
    def resetBoard(board):
        for x in range(8):
            for y in range(8):
                board[x][y] = ' '

#this function puts @@ and "" in the middle of the grid, starting position
        board[3][3] = '@@'
        board[3][4] = '""'
        board[4][3] = '@@'
        board[4][4] = '""'
#this function prints new board
    def getNewBoard():
        board = []
        for i in range(8):
            board.append([' '] * 1)

        return board
#this function checks if it is a valid move, if it is a valid move it will
go ahead
    def isValidMove(board, tile, xstart, ystart):
        if board[xstart][ystart] != ' ' or not isOnBoard(xstart, ystart):
            return False

        board[xstart][ystart] = tile  # t

        if tile == 'X':
            otherTile = 'O'
        else:
            otherTile = 'X'
#this function shows the tiles to flip
        tilesToFlip = []
        for xdirection, ydirection in [[0, 1], [1, 1], [1, 0], [1, -1], [0,
-1], [-1, -1], [-1, 0], [-1, 1]]:
            x, y = xstart, ystart
            x += xdirection
            y += ydirection
```

```
            if isOnBoard(x, y) and board[x][y] == otherTile:
#There is a piece belonging to the other player next to our piece.
                x += xdirection
                y += ydirection
                if not isOnBoard(x, y):
                    continue
                while board[x][y] == otherTile:
                    x += xdirection
                    y += ydirection
                    if not isOnBoard(x, y):
                        break
                if not isOnBoard(x, y):
                    continue
                if board[x][y] == tile:
#There are pieces to flip over. Go in the reverse direction until we reach
the original space, noting all the tiles along the way.
                    while True:
                        x -= xdirection
                        y -= ydirection
                        if x == xstart and y == ystart:
                            break
                        tilesToFlip.append([x, y])
#this function restores the empty areas
        board[xstart][ystart] = ' '
#this function shows if nothing is flipped, it is an inccorrect move
        if len(tilesToFlip) == 0:
            return False
        return tilesToFlip

    def isOnBoard(x, y):
        return x >= 0 and x <= 7 and y >= 0 and y <= 7

    def getBoardWithValidMoves(board, tile):
        dupeBoard = getBoardCopy(board)

        for x, y in getValidMoves(dupeBoard, tile):
            dupeBoard[x][y] = '.'
        return dupeBoard

    def getValidMoves(board, tile):

        validMoves = []

        for x in range(8):
            for y in range(8):
                if isValidMove(board, tile, x, y) != False:
                    validMoves.append([x, y])
        return validMoves
#this function finds the score by counting all the tiles and determines the
score.
    def getScoreOfBoard(board):
        xscore = 0
        oscore = 0
        for x in range(8):
            for y in range(8):
                if board[x][y] == 'X':
                    xscore += 1
                if board[x][y] == 'O':
                    oscore += 1
        return {'X': xscore, 'O': oscore}
```

```
    def enterPlayerTile():
#Lets the player type which tile they want to be.

        tile = ''
        while not (tile == 'X' or tile == 'O'):
            print(name,'Do you want to be X or O? (X = @@ & O = "") ')
            tile = input().upper()
            print('okay', name2, 'you are the other symbol')


        if tile == 'X':
            return ['X', 'O']
        else:
            return ['O', 'X']
#this function picks randomly any player to go first
    def whoGoesFirst():

        if random.randint(0, 1) == 0:
            return 'player1'
        else:
            return 'player2'

#this function allows the player to play again or quit
    def playAgain():
        print('Do you want to play again? (yes or no)')
        return input().lower().startswith('y')

    def makeMove(board, tile, xstart, ystart):
        tilesToFlip = isValidMove(board, tile, xstart, ystart)

        if tilesToFlip == False:
            return False

        board[xstart][ystart] = tile
        for x, y in tilesToFlip:
            board[x][y] = tile
        return True

    def getBoardCopy(board):
        dupeBoard = getNewBoard()

        for x in range(8):
            for y in range(8):
                dupeBoard[x][y] = board[x][y]

        return dupeBoard

    def isOnCorner(x, y):
        return (x == 0 and y == 0) or (x == 7 and y == 0) or (x == 0 and y
== 7) or (x == 7 and y == 7)

#Let the player type in their move.
#Returns the move
    def getPlayerMove(board, playerTile):

        Digits1to8 = '1 2 3 4 5 6 7 8'.split()
        while True:
            print('Enter your move, or type quit to end ')
            move = input().lower()
            if move == 'quit':
                return 'quit'
```

```
            if len(move) == 2 and move[0] in Digits1to8 and move[1] in
Digits1to8:
                x = int(move[0]) - 1
                y = int(move[1]) - 1
                if isValidMove(board, playerTile, x, y) == False:
                    continue
                else:
                    break
            else:
                print('That is not a valid move. Type the x axis (1-8),
then y axis (1-8).')

        return [x, y]

    def getPlayerMove(board, playerTile):
        possibleMoves = getValidMoves(board, playerTile)

#randomize the possible moves
        random.shuffle(possibleMoves)

        for x, y in possibleMoves:
            if isOnCorner(x, y):
                return [x, y]

        bestScore = -1
        for x, y in possibleMoves:
            dupeBoard = getBoardCopy(board)
            makeMove(dupeBoard, playerTile, x, y)
            score = getScoreOfBoard(dupeBoard)[playerTile]
            if score > bestScore:
                bestMove = [x, y]
                bestScore = score
        return bestMove

#This functions shows the current score
    def showPoints(playerTile, player2Tile):
#this functions shows the score and also prints out the percentages
        scores = getScoreOfBoard(mainBoard)
        print('You have %s points. player2 has %s points.' %
(scores[playerTile], scores[player2TileTile]))

    print('Welcome!')

#This functions resets the board and you can play again
    while True:

        mainBoard = getNewBoard()
        resetBoard(mainBoard)
        playerTile, player2Tile = enterPlayerTile()
        turn = whoGoesFirst()
        print('The ' + turn + ' will go first.')

        while True:
            if turn == 'player':

                if turn:
                    validMovesBoard = getBoardWithValidMoves(mainBoard,
playerTile)
                    drawBoard(validMovesBoard)
```

```
                else:
                    drawBoard(mainBoard)
                showPoints(playerTile, player2Tile)
                move = getPlayerMove(mainBoard, playerTile)
                if move == 'quit':
                    print('Thanks for playing!')
                    sys.exit()  #ends the game
                else:
                    makeMove(mainBoard, playerTile, move[0], move[1])

                if getValidMoves(mainBoard, player2Tile) == []:
                    break
                else:
                    turn = 'player2'

            else:
                drawBoard(mainBoard)
                showPoints(playerTile, player2Tile)
                input('Press Enter to see the player2\'s move.')
                x, y = getPlayerMove(mainBoard, player2Tile)
                makeMove(mainBoard, player2Tile, x, y)

                if getValidMoves(mainBoard, playerTile) == []:
                    break
                else:
                    turn = 'player'

#final score
        drawBoard(mainBoard)
        scores = getScoreOfBoard(mainBoard)
        print('X scored %s points. O scored %s points.' % (scores['X'],
scores['O']))
        if scores[playerTile] > scores[player2Tile]:
            print('You beat the player2 by %s points! Congratulations!' %
(scores[playerTile] - scores[player2Tile]))
        elif scores[playerTile] < scores[player2Tile]:
            print('You lost. The player2 beat you by %s points.' %
(scores[player2Tile] - scores[playerTile]))
        else:
            print('The game was a tie!')

        if not playAgain():
            break
pass
def register():
    pass
def login():
    pass
main()
```