class TicTacToe:

def \_\_init\_\_(self):

self.board = [[' ' for \_ in range(3)] for \_ in range(3)]

self.turn = 'X'

self.result = ''

def make\_move(self, row, col):

if not self.is\_valid\_move(row, col):

return False

self.board[row][col] = self.turn

self.check\_win()

if self.check\_win():

self.result = f"{self.turn} wins!"

return True

self.switch\_turn()

return False

def is\_valid\_move(self, row, col):

return self.board[row][col] == ' '

def check\_win(self):

for i in range(3):

if self.board[i][0] == self.board[i][1] == self.board[i][2] != ' ':

return True

if self.board[0][i] == self.board[1][i] == self.board[2][i] != ' ':

return True

if self.board[0][0] == self.board[1][1] == self.board[2][2] != ' ':

return True

if self.board[0][2] == self.board[1][1] == self.board[2][0] != ' ':

return True

return False

def switch\_turn(self):

self.turn = 'O' if self.turn == 'X' else 'X'

重构后的代码做了以下优化：

1. 将检查游戏胜利的逻辑封装在一个单独的check\_win()方法中，使得代码更加模块化，易于维护和扩展。
2. 引入了一个is\_valid\_move()方法，用于检查是否可以在给定的行和列上放置标记。这样可以避免无效的移动，提高了游戏的逻辑性和用户体验。
3. 在make\_move()方法中，先检查移动是否有效，然后再执行其他操作。这样可以避免无效的移动，并提高代码的健壮性。
4. 将游戏结果存储在result属性中，使得代码更加清晰，易于理解和跟踪游戏状态。
5. 在切换玩家标记时，使用条件运算符简化了代码，并提高了可读性。