# Tic-Tac-Toe

# Basic game loop structure

# - move\_input()

# - update\_board()

# - render\_board()

# WIN\_SET contains sets of all the spots on the board that equal a win

import random

class Tictactoe:

#constructor

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

#static set of valid 'win' moves

self.WIN\_SET = (

(0,1,2), (3,4,5), (6,7,8), #horizontal

(0,3,6), (1,4,7), (2,5,8), #vertical

(0,4,8), (2,4,6)) #diagonal

self.board = [' '] \* 9

self.current\_player = 'x'

self.player\_choice = ' '

self.players = {'x': ' ', 'o': ' ' }

self.winner = None

self.move = None

#methods

def choose\_side(self):

while True:

t.player\_choice = input("""Please choose a side: 'X' or 'O': """)

t.player\_choice = t.player\_choice.lower()

#make sure its a valid choice

if t.player\_choice == 'x' or t.player\_choice == 'o':

print("You've chosen: " + t.player\_choice)

break

else:

print("Invalid entry.")

t.player\_choice = ' '

continue

#player 'x' goes first

#player 'o' ai goes first

if t.player\_choice == 'x':

self.players['x'] = "Human"

self.players['o'] = "Super AI"

else:

self.players['x'] = "Super AI"

self.players['o'] = "Human"

def check\_move\_valid(self):

global move

if move <= 8 and move >= 0:

if t.board[move] == ' ':

return True

else:

print("That area is already filled. Please enter a different number.")

return False

else:

print("Not a valid position. Must be a number between [0-8].")

def check\_win\_result(self):

#if matching symbols in WIN\_SET (e.g. 0,4,8)

#row[0] = 0, row[1] = 4, row[2] = 8

for row in self.WIN\_SET:

if t.board[row[0]] == t.board[row[1]] == t.board[row[2]] != ' ':

print(t.board[row[0]] + " wins!")

t.winner = "true"

if ' ' not in t.board and t.winner == None:

print("Tie")

t.winner = "false"

#==============================================================================

# Human and AI functions

def human\_move(self):

return int(input("Enter a number [0-8]: "))

def ai\_move(self):

return random.randrange(9)

def ai\_move\_hard(self):

#best move if ai goes first

if t.board[4] == ' ':

return 4

else:

#get positions the player has taken so far

player\_pos = [i for i in range(len(t.board)) if t.board[i] != t.current\_player and t.board[i] != ' ']

#block the players winning move by comparing what squares the player has against what they need to win

if len(player\_pos) == 2:

for row in self.WIN\_SET:

if player\_pos[0] in row and player\_pos[1] in row:

#find the last square the player needs to win and take it!

if player\_pos[0] != row[0]:

return row[0]

elif player\_pos[1] != row[1]:

return row[1]

else:

return row[2]

else:

return random.choice(t.choose\_empty\_pos())

else:

return random.choice(t.choose\_empty\_pos())

def choose\_empty\_pos(self):

#chooses a random, empty square from the remaining squares on the board

empty\_pos = [i for i in range(len(t.board)) if t.board[i] == ' ']

return empty\_pos

#==============================================================================

# Game loop methods

def next\_move(self):

global move

if t.current\_player == 'x':

if t.players['x'] == "Human":

move = t.human\_move()

print("--Player--")

else:

t.players['x'] == "Super AI"

move = t.ai\_move\_hard()

print("--Computer--")

else:

if t.players['o'] == "Human":

move = t.human\_move()

print("--Player--")

else:

t.players['o'] == "Super AI"

move = t.ai\_move\_hard()

print("--Computer--")

def update\_model(self):

if t.check\_move\_valid():

#set the list element to match the current players symbol

t.board[move] = t.current\_player

#print(t.board[0:8])

#check to see if anyone has won

t.check\_win\_result()

#switch players

if t.current\_player == 'x':

t.current\_player = 'o'

else:

t.current\_player = 'x'

def render\_board(self):

print('-------------')

print('| %s | %s | %s |' % (self.board[0], self.board[1], self.board[2]))

print('-------------')

print('| %s | %s | %s |' % (self.board[3], self.board[4], self.board[5]))

print('-------------')

print('| %s | %s | %s |' % (self.board[6], self.board[7], self.board[8]))

print('-------------')

def position\_board(self):

print("HOW TO PLAY: Enter a number when prompted to fill the corresponding space")

print('-------------')

print('| 0 | 1 | 2 |')

print('-------------')

print('| 3 | 4 | 5 |')

print('-------------')

print('| 6 | 7 | 8 |')

print('-------------')

#==============================================================================

# Main

#==============================================================================

if \_\_name\_\_ == '\_\_main\_\_':

t = Tictactoe()

t.position\_board() #shows board for moveset

t.choose\_side() #choose a side

#game loop

while t.winner is None:

t.next\_move()

t.update\_model()

t.render\_board()