**T3 – LE BAO DUY NGUYEN (102449993)**

**(extension included)**

from random import randrange

*# static game data - doesn't change (hence immutable tuple data type)*

WIN\_SET = ((0, 1, 2), (3, 4, 5), (6, 7, 8), (0, 3, 6),

(1, 4, 7), (2, 5, 8), (0, 4, 8), (2, 4, 6))

*# global variables for game data*

board = [' '] \* 9

current\_player = '' *# 'x' or 'o' for first and second player*

ai\_choices = {'r': 'Random', 'a': 'Average Ai', 's': 'Smart AI'}

ai\_choice = None

ai\_vs\_ai = False

players = {'x': 'Human', 'o': 'Super AI'}

winner = None

move = None

firstTurn = True

quitting = False

*# aesthetics...*

HR = '-' \* 40

*#==============================================================================*

*# Game model functions*

def reset\_game\_data():

'''Resets the game data in the global variables to the defaults'''

global board, current\_player, ai\_choice, ai\_choices, ai\_vs\_ai, players, winner, move, firstTurn, quitting

board = [' '] \* 9

current\_player = '' *# 'x' or 'o' for first and second player*

ai\_choices = {'r': 'Random', 'a': 'Average Ai', 's': 'Smart AI'}

ai\_choice = None

ai\_vs\_ai = False

players = {'x': 'Human', 'o': 'Super AI'}

winner = None

move = None

firstTurn = True

quitting = False

def check\_move():

'''This function will return True if ``move`` is valid (in the board range

and free cell), or print an error message and return False if not valid.

``move`` is an int board position [0..8].

'''

global move

try:

move = int(move)

if board[move] == ' ':

return True

else:

print('>> Sorry - that position is already taken!')

return False

except:

print('>> %s is not a valid position! Must be int between 0 and 8.' % move)

return False

def check\_set\_for\_player(set, player):

count = 0

move = -1

for x, index in enumerate(set):

if board[index] == player:

count += 1

elif board[index] is not 'x' and board[index] is not 'o':

move = index

if x is 2 and count is not 2:

*# if the count is at two when we have ennumerated through the set then we set move*

*# to -1 to indicate to return false*

move = -1

else:

move = -1

return move

def check\_for\_result():

'''Checks the current board to see if there is a winner, tie or not.

Returns a 'x' or 'o' to indicate a winner, 'tie' for a stale-mate game, or

simply False if the game is still going.

'''

for row in WIN\_SET:

if board[row[0]] == board[row[1]] == board[row[2]] != ' ':

return board[row[0]] *# return an 'x' or 'o' to indicate winner*

if ' ' not in board:

return 'tie'

return None

*#==============================================================================*

*# agent (human or AI) functions*

def get\_human\_move():

'''Get a human players raw input. Returns None if a number is not entered.'''

return input('[0-8] >> ')

def get\_ai\_move():

'''Get the AI's next move '''

*# A simple dumb random move - valid or NOT!*

*# Note: It is the models responsibility to check for valid moves...*

return randrange(9) *# [0..8]*

def get\_average\_ai\_move():

global current\_player

if current\_player == 'x':

otherPlayer = 'o'

else:

otherPlayer = 'x'

for set in WIN\_SET:

*# Check if other player is about to win using check\_set().*

chk = check\_set\_for\_player(set, otherPlayer)

if chk is not -1:

return chk

*# Then make the move to stop the other player from winning*

*#else choose a random option*

return randrange(9) *# [0..8]*

def get\_smart\_ai\_move():

global current\_player, firstTurn

if current\_player == 'x':

otherPlayer = 'o'

else:

otherPlayer = 'x'

for set in WIN\_SET:

*# Check if this player is about to win using check\_set()*

chk = check\_set\_for\_player(set, current\_player)

if chk is not -1:

return chk

*# Then make the move that allows you to win*

*# Check if other player is about to win using check\_set().*

chk = check\_set\_for\_player(set, otherPlayer)

if chk is not -1:

return chk

*# Then make the move to stop the other player from winning*

*# if its the first turn, return the middle*

if firstTurn:

firstTurn = False *# set this to False so it only tries this once.*

return 4

*# If neither condition*

*# Then make a random move from available spaces*

return randrange(9) *# [0..8]*

*#==============================================================================*

*# Standard trinity of game loop methods (functions)*

def process\_input():

'''Get the current players next move.'''

*# save the next move into a global variable*

global move, ai\_choice

if current\_player == 'x':

move = get\_human\_move()

elif ai\_choice is 's':

move = get\_smart\_ai\_move()

elif ai\_choice is 'a':

move = get\_average\_ai\_move()

elif ai\_choice is 'r':

move = get\_ai\_move()

else:

move = get\_ai\_move() *# Defaults to the random AI*

def process\_ai\_vs\_ai\_input():

'''Get the current players next move, where there are two ai battling.'''

*# save the next move into a global variable*

global move, ai\_choice

if current\_player == 'x':

move = get\_smart\_ai\_move() *# Always Smart AI vs another AI*

elif ai\_choice is 's':

move = get\_smart\_ai\_move()

elif ai\_choice is 'a':

move = get\_average\_ai\_move()

elif ai\_choice is 'r':

move = get\_ai\_move()

else:

move = get\_ai\_move() *# Defaults to the random AI*

def update\_model():

'''If the current players input is a valid move, update the board and check

the game model for a winning player. If the game is still going, change the

current player and continue. If the input was not valid, let the player

have another go.

'''

global winner, current\_player

if check\_move():

*# do the new move (which is stored in the global 'move' variable)*

board[move] = current\_player

*# check board for winner (now that it's been updated)*

winner = check\_for\_result()

*# change the current player (regardless of the outcome)*

if current\_player == 'x':

current\_player = 'o'

else:

current\_player = 'x'

else:

print('Try again')

def render\_board():

'''Display the current game board to screen.'''

print(' %s | %s | %s' % tuple(board[:3]))

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

print(' %s | %s | %s' % tuple(board[3:6]))

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

print(' %s | %s | %s' % tuple(board[6:]))

*# pretty print the current player name*

if winner is None:

print('The current player is: %s' % players[current\_player])

*#==============================================================================*

def show\_human\_help():

'''Show the player help/instructions. '''

tmp = '''

To make a move enter a number between 0 - 8 and press enter.

The number corresponds to a board position as illustrated:

0 | 1 | 2

---------

3 | 4 | 5

---------

6 | 7 | 8

'''

print(tmp)

print(HR)

def run\_human\_vs\_ai\_game():

'''Runs a Human Vs AI game'''

show\_human\_help()

*# by default the human player starts. This could be random or a choice.*

global current\_player

current\_player = 'x'

*# show the initial board and the current player's move*

render\_board()

*# Standard game loop structure*

while winner is None:

process\_input()

update\_model()

render\_board()

def run\_ai\_vs\_ai\_game():

'''Runs a game between a Smart AI and a selected AI'''

*# by default 'x' starts*

global current\_player

current\_player = 'x'

*# Standard game loop structure*

while winner is None:

process\_ai\_vs\_ai\_input()

update\_model()

*# Render the Final Board State*

render\_board()

*#==============================================================================*

*# Separate the running of the game using a \_\_name\_\_ test. Allows the use of this*

*# file as an imported module*

*#==============================================================================*

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

*# Welcome ...*

print('Welcome to the amazing+awesome tic-tac-toe! \n')

while not quitting:

*# Choose to play or have the AI fight it out*

print('Do you want the smart AI to fight for your honor?')

choice = raw\_input ('[Y/N] >> ')

if choice is 'Y' or choice is 'y':

ai\_vs\_ai = True

else:

ai\_vs\_ai = False

*# Select the AI opponent playing second*

print('\nSelect the opponent')

for key in ai\_choices.keys():

print(key, ai\_choices[key])

ai\_choice = raw\_input('>> ')

if ai\_vs\_ai:

run\_ai\_vs\_ai\_game()

else:

run\_human\_vs\_ai\_game()

*# Some pretty messages for the result*

print(HR)

if winner == 'tie':

print('TIE!')

elif winner in players:

print('%s is the WINNER!!!' % players[winner])

print(HR)

print('Do you wish to Play Again?')

tmp = raw\_input('[Y/N]>> ')

if tmp is 'Y' or tmp is 'y':

reset\_game\_data()

else:

quitting = True

print('Goodbye, Thank you for playing.')

**RESULT**

**A screenshot of text

Description automatically generated**