# [ ] This project is an implementation a Tic Tac Toe game.

# The logic of the game is in the `main` function, read it before writing any code.

# Use the description and examples under each of the following functions to implement them:

# 1) draw(board)

# 2) available(location, board)

# 3) mark(player, location, board)

# 4) check\_win(board)

# 5) check\_tie(board)

from IPython.display import clear\_output #to clear the output (specific to Jupyter notebooks and ipython)

from random import randint

def draw(board):

"""

Draw the `board` table.

The board reflects the current state of the game, a number indicates an available location.

args:

board: 3x3 table (list of lists) containing the current state of the game

returns:

None

examples:

At the beginning of the game: board = [['7', '8', '9'], ['4', '5', '6'], ['1', '2', '3']]

The printout should look like:

7 | 8 | 9

-----------

4 | 5 | 6

-----------

1 | 2 | 3

After a few marks: board = [['7', '8', 'X'], ['O', 'O', '6'], ['1', 'X', '3']]

The printout should look like:

7 | 8 | X

-----------

O | O | 6

-----------

1 | X | 3

"""

#TODO

pass

def available(location, board):

"""

Check the availability of a `location` on the current `board`

An available location on the board contains a number between 1 and 9 stored as a string.

If the location contains 'X' or 'O', the location is not available and the function should return False;

otherwise, the function should return True indicating the location is available

args:

location: a number between 1 and 9 stored as a string

board: 3x3 table (list of lists) containing the current state of the game

returns:

True if the location is available. False if the location is not available

examples:

At the beginning of the game: board = [['7', '8', '9'], ['4', '5', '6'], ['1', '2', '3']]

The printout should look like:

7 | 8 | 9

-----------

4 | 5 | 6

-----------

1 | 2 | 3

available("1", board) --> returns True

available("9", board) --> returns True

After a few marks: board = [['7', '8', 'X'], ['O', 'O', '6'], ['1', 'X', '3']]

The printout should look like:

7 | 8 | X

-----------

O | O | 6

-----------

1 | X | 3

available("1", board) --> returns True, because there is a number

available("5", board) --> returns False, because there is 'O'

available("9", board) --> returns False, because there is 'X'

"""

#TODO

pass

def mark(player, location, board):

"""

Mark `location` on the `board` with the `player` symbol.

Should replace the `location` number on the board with `X` or `O`

args:

player: player's symbol, either 'X' or 'O'

location: a number between 1 and 9 stored as a string

board: 3x3 table (list of lists) containing the current state of the game

returns:

None

examples:

At the beginning of the game: board = [['7', '8', '9'], ['4', '5', '6'], ['1', '2', '3']]

The printout should look like:

7 | 8 | 9

-----------

4 | 5 | 6

-----------

1 | 2 | 3

After mark('O', '4', board)

The printout should look like:

7 | 8 | 9

-----------

O | 5 | 6

-----------

1 | 2 | 3

After mark('X', '3', board)

The printout should look like:

7 | 8 | 9

-----------

O | 5 | 6

-----------

1 | 2 | X

After mark('O', '9', board)

The printout should look like:

7 | 8 | O

-----------

O | 5 | 6

-----------

1 | 2 | X

"""

#TODO

pass

def check\_win(board):

"""

Check if there is a winner.

A win happens if the either of the players was able to place 3 symbols ('X' or 'O') in:

a horizontal, vertical, diagonal, or anti-diagonal placement.

args:

board: 3x3 table (list of lists) containing the current state of the game

returns:

True if there is a winner. False if there is no winner yet

examples:

Horizontal win:

================

7 | O | 9

-----------

X | X | X

-----------

1 | O | 3

check\_win(board) --> returns True, because 'X' won

O | O | O

-----------

X | X | 6

-----------

X | O | 3

check\_win(board) --> returns True, because 'O' won

Vertical win:

================

7 | 8 | X

-----------

X | O | X

-----------

O | O | X

check\_win(board) --> returns True, because 'X' won

X | O | O

-----------

4 | O | 6

-----------

X | O | X

check\_win(board) --> returns True, because 'O' won

Diagonal win:

================

X | 8 | O

-----------

4 | X | X

-----------

O | O | X

check\_win(board) --> returns True, because 'X' won

O | X | O

-----------

X | O | X

-----------

1 | 2 | O

check\_win(board) --> returns True, because 'O' won

Anti-Diagonal win:

================

O | 8 | X

-----------

4 | X | X

-----------

X | O | O

check\_win(board) --> returns True, because 'X' won

7 | 8 | O

-----------

X | O | X

-----------

O | O | X

check\_win(board) --> returns True, because 'O' won

No winners yet:

================

O | 8 | 9

-----------

4 | X | X

-----------

X | O | O

check\_win(board) --> returns False

"""

#TODO

pass

def check\_tie(board):

"""

Check the game for a tie, no available locations and no winners.

args:

board: 3x3 table (list of lists) containing the current state of the game

returns:

True if there is a tie. False the board is not full yet or there is a winner

examples:

O | O | X

-----------

X | X | O

-----------

O | O | X

check\_tie(board) --> returns True

O | O | 9

-----------

X | X | 6

-----------

X | O | 3

check\_tie(board) --> returns False, because there are still available location

"""

#TODO

pass

def dashes():

"""Print a fancy line of dashes"""

print("o" + 35 \*'-' + "o")

def display(message):

"""

Print `message` in the center of a 35 characters string

args:

message: string to display

returns:

None

"""

print("|{:^35s}|".format(message))

def main():

# initializing game

board = [['7', '8', '9'], ['4', '5', '6'], ['1', '2', '3']]

# select the first player randomly

player = ['X', 'O']

turn = randint(0, 1)

win = False

tie = False

while(not win and not tie):

# switch players

turn = (turn + 1) % 2

current\_player = player[turn] # contains 'X' or 'O'

clear\_output()

# display header

dashes()

display("TIC TAC TOE")

dashes()

# display game board

print()

draw(board)

print()

# display footer

dashes()

# player select a location to mark

while True:

location = input("|{:s} Turn, select a number (1, 9): ".format(current\_player))

if available(location, board):

break # Only the user input loop, main loop does NOT break

else:

print("Selection not available!")

dashes()

# mark selected location with player symbol ('X' or 'O')

mark(current\_player, location, board)

# check for win

win = check\_win(board)

# check for tie

tie = check\_tie(board)

# Display game over message after a win or a tie

clear\_output()

# display header

dashes()

display("TIC TAC TOE")

dashes()

# display game board (Necessary to draw the latest selection)

print()

draw(board)

print()

# display footer

dashes()

display("Game Over!")

if(tie):

display("Tie!")

elif(win):

display("Winner:")

display(current\_player)

dashes()

# Run the game

main()