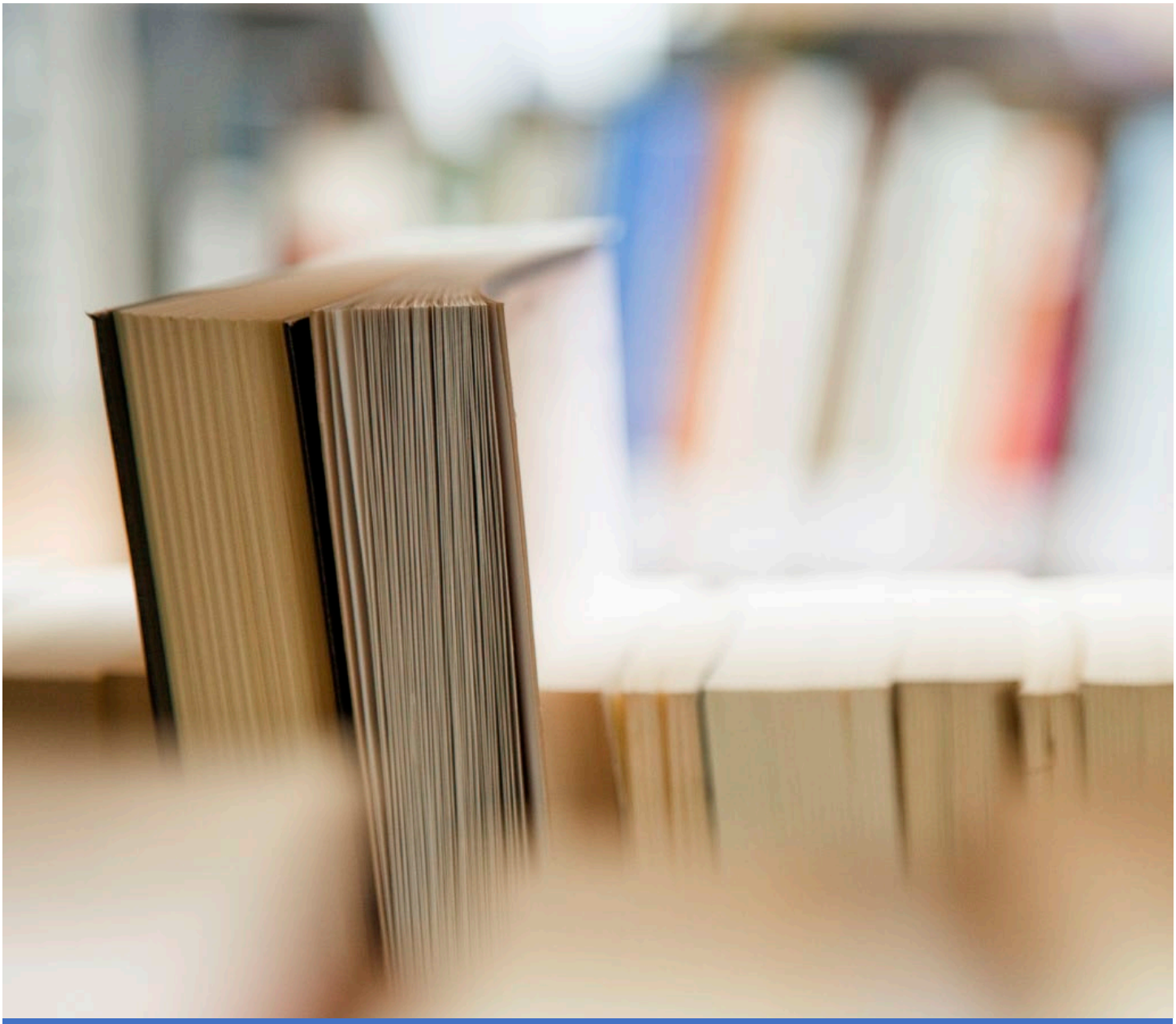


# Tic-Tac-Toe game

Coursework 3

Tony, YIT19488399

4/19/22

Artificial Intelligence

• "A brief explanation of the strategy or algorithm that you use to determine the move for AI player."

The algorithm chosen in my Tic-Tac-Toe game is MINIMAX because of Lab week 6 on the AI module. This is a great opportunity to apply a minimax search algorithm to this game with two players: MAX and MIN.

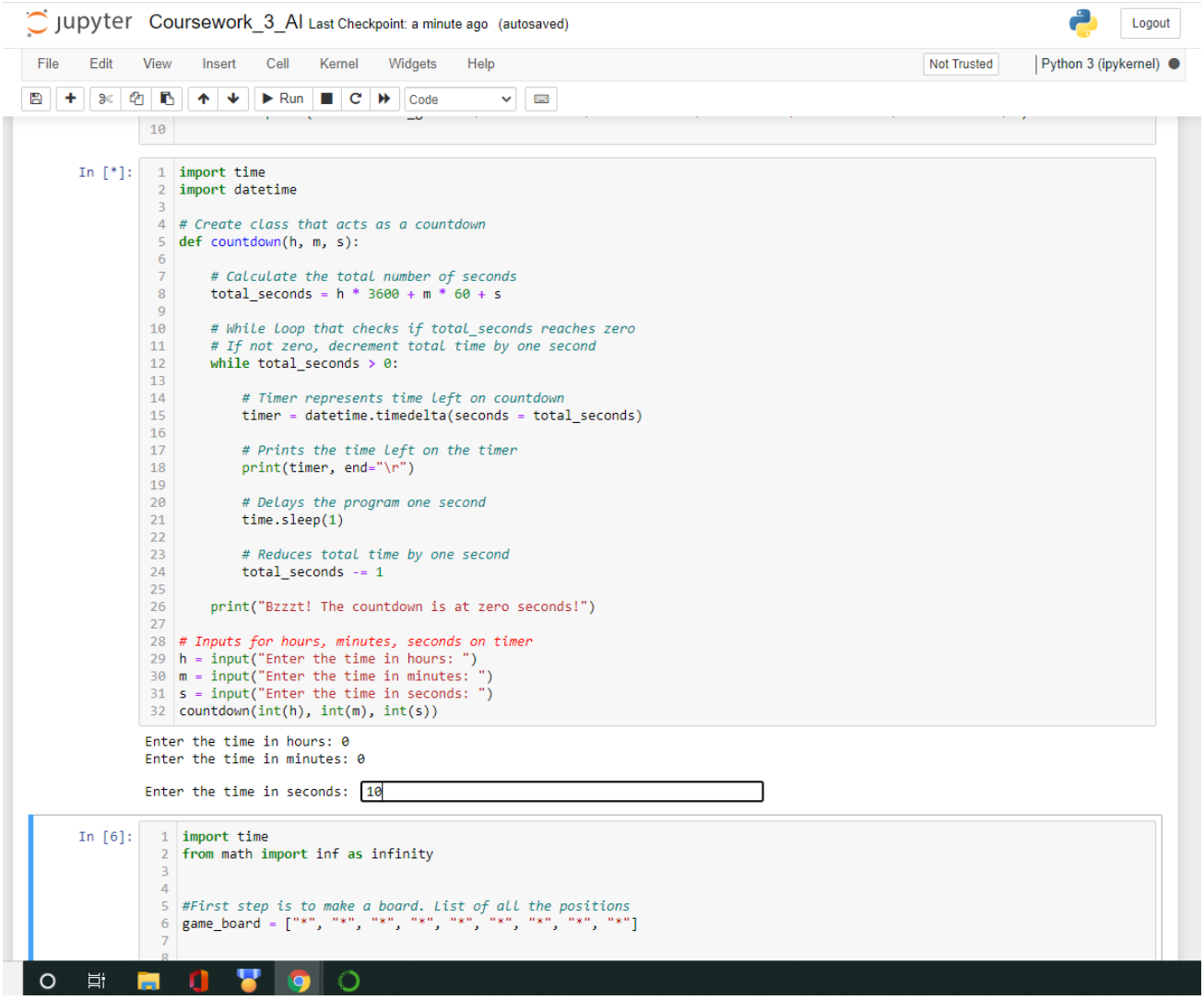
Max is maximizing score value in each move and Min minimizes score value in each move. If both players play optimally, it will be a draw at the end of the game. Game tree representation is complex and large to draw. See the link for a better understanding of the tree. https://minimax-visualizer.herokuapp.com/. Basically, minimax determines the optimal strategy and movement for each player every turn, and therefore it is a search algorithm. It checks all the possible positions on the game board in each movement and decides the best one independently for each player.

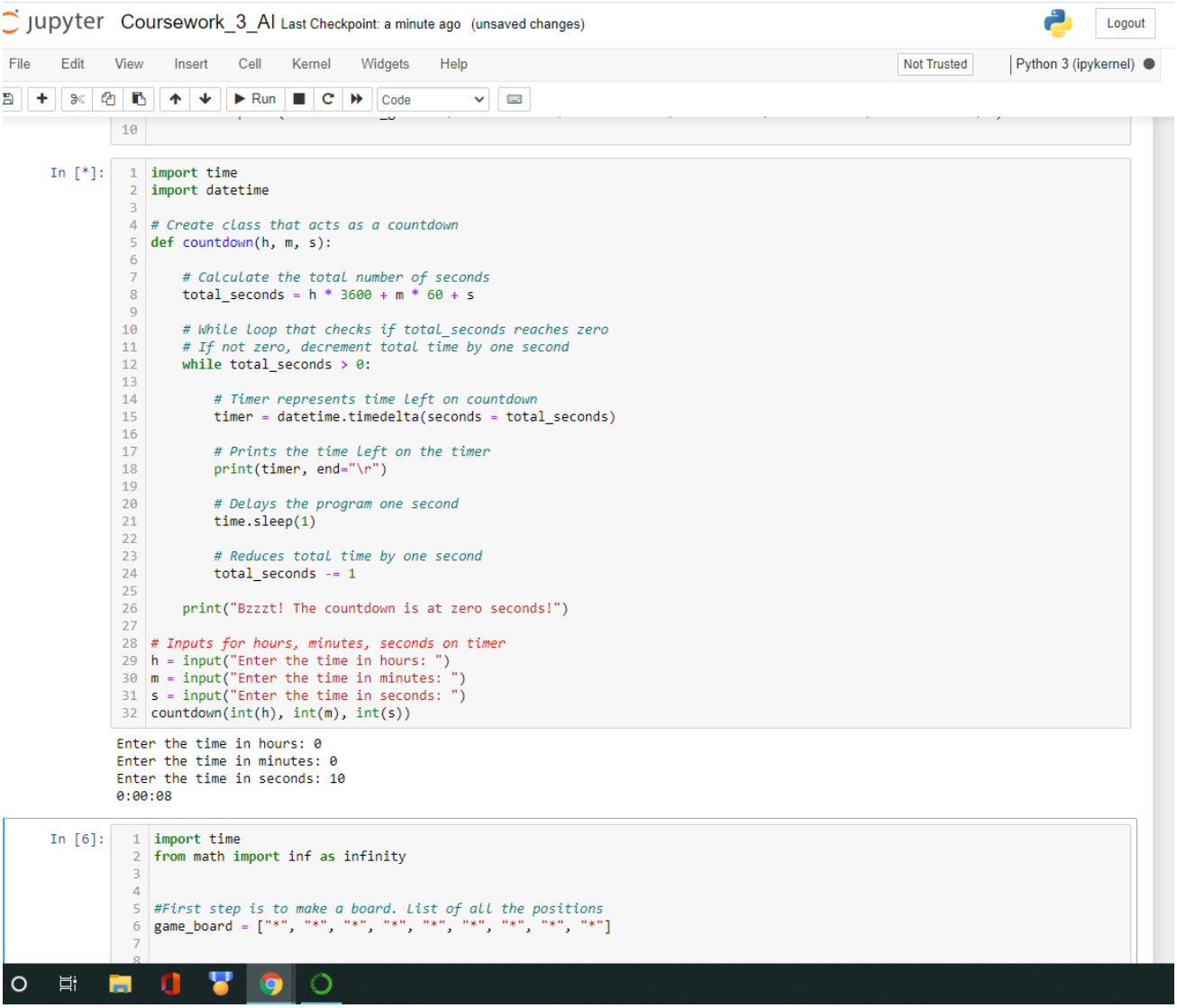
Minimax is a recursive algorithm. Also, can be a backtracking algorithm from the terminal state or node.

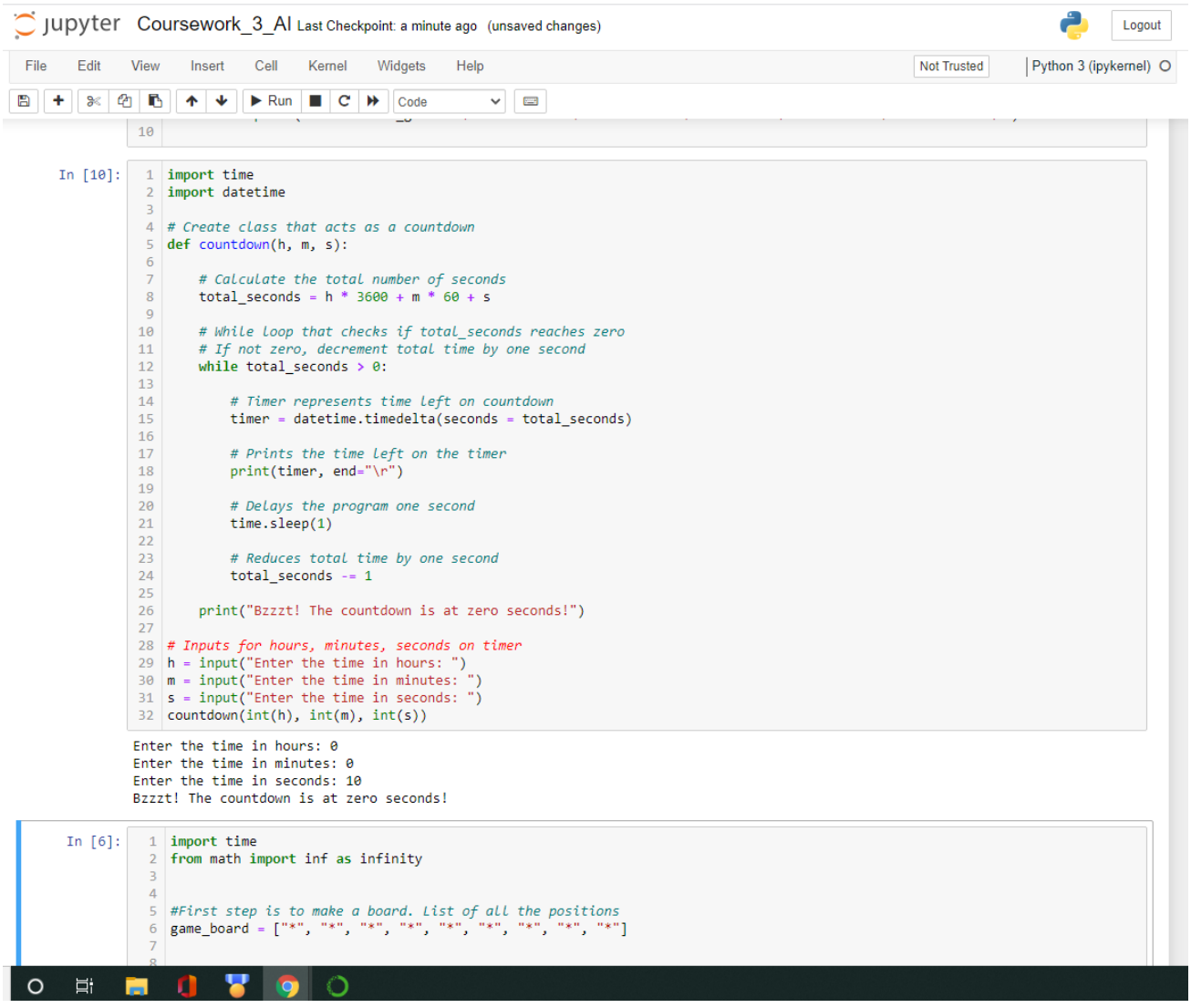
AI player looks intelligent and smart, but it is code implemented especially. Correct indentations with right loops and if statements are the secret of this simple code. Nothing intelligent! Only lines of code! Human NEVER can win against an AI player, called robot in my code. The best end in this game for the human player is a draw.

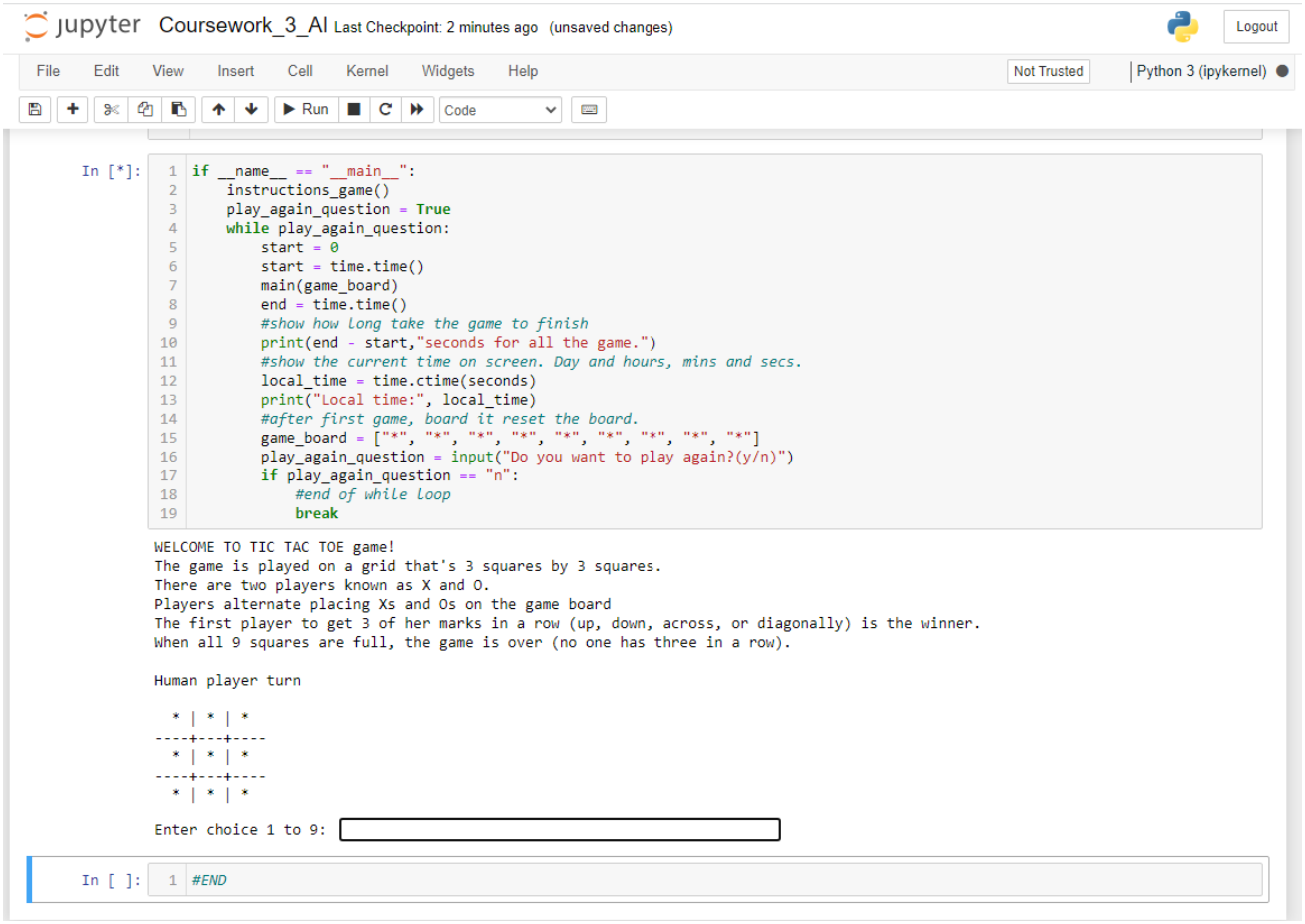
I included a timer countdown in my code for entertainment purposes only. AI player moves much faster than a human player. And if the human player needs more time to choose a cell, this is ok, and the game will continue.

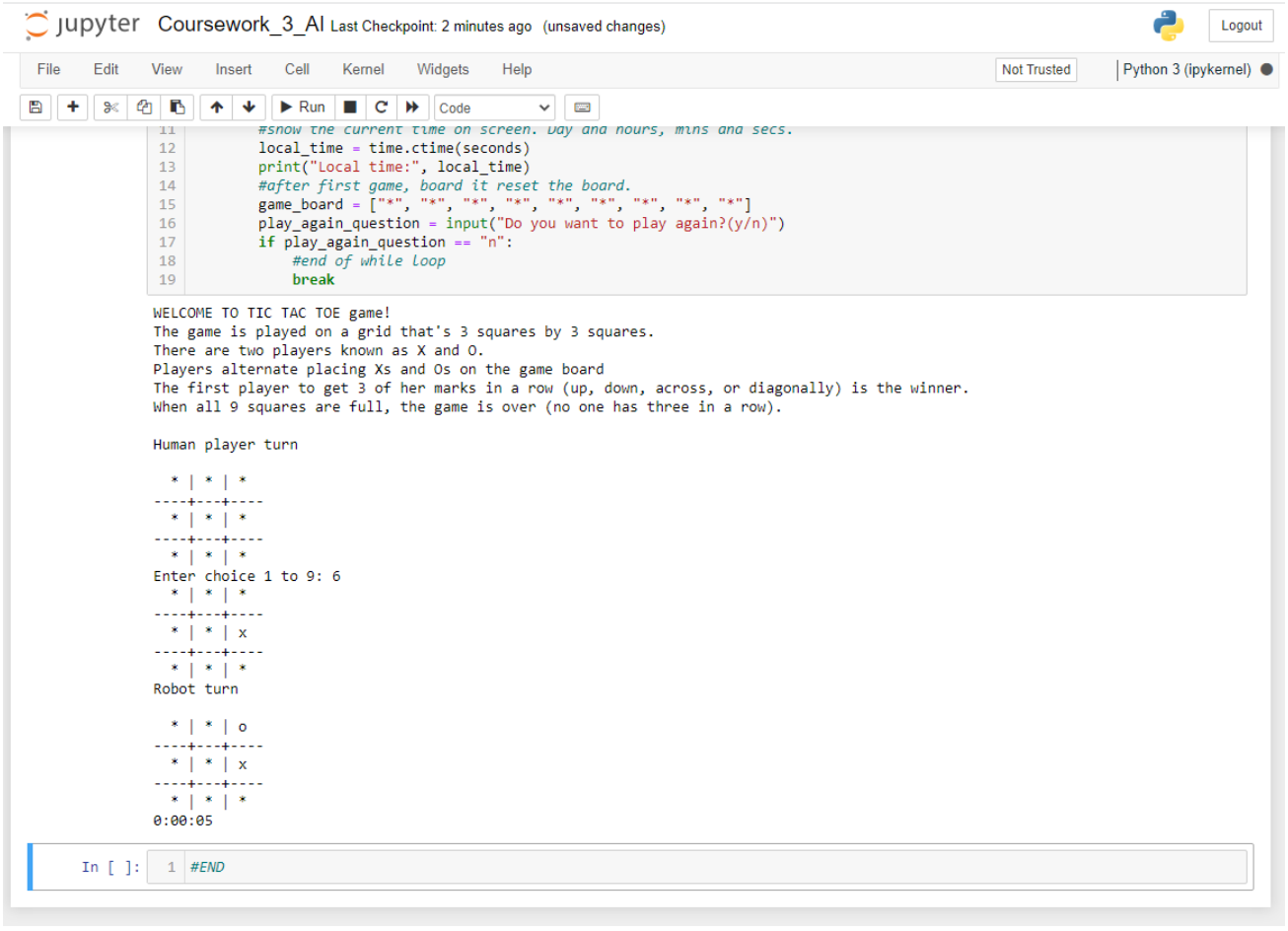
• Result of a demo run of the game. You should take screenshots of your results from Jupyter Notebooks and insert them into your report.

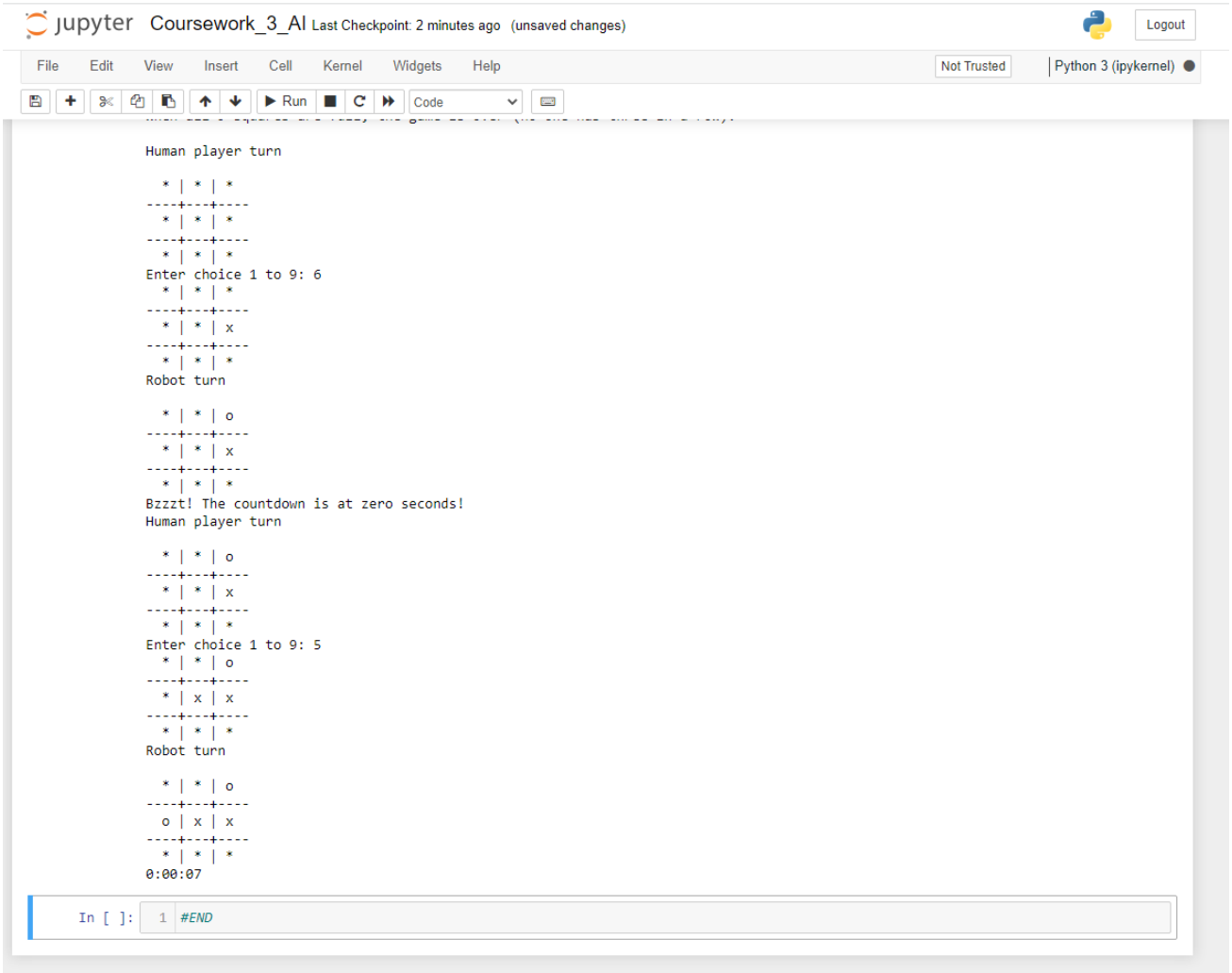


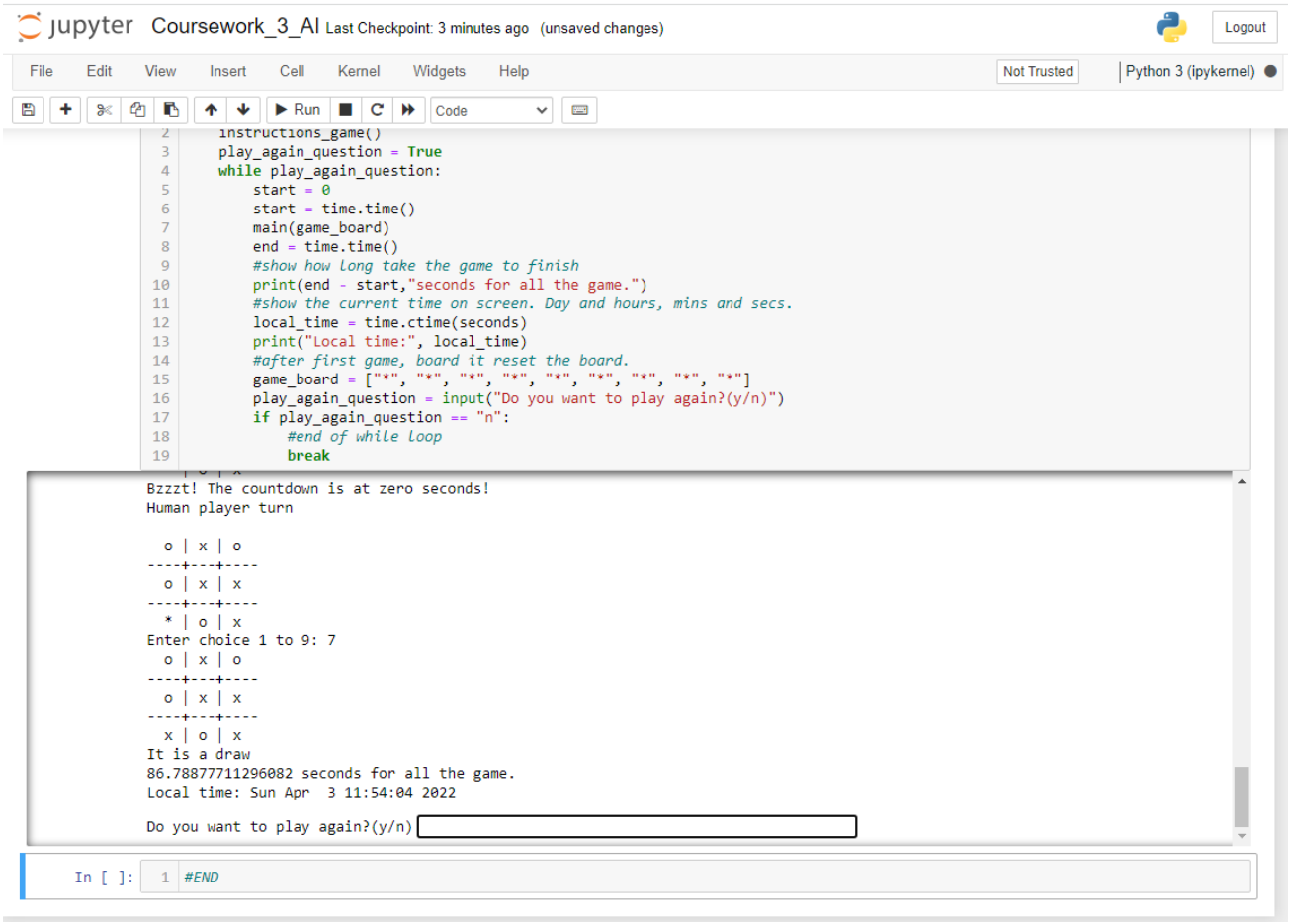

jupyter Coursework_3_AI Last Checkpoint: a minute ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [*]:
import time
import datetime
# Create class that acts as a countdown
def countdown(h, m, s):
# Calculate the total number of seconds
total_seconds = h * 3600 + m * 60 + s
# While loop that checks if total_seconds reaches zero
# If not zero, decrement total time by one second
while total_seconds > 0:
# Timer represents time left on countdown
timer = datetime.timedelta(seconds = total_seconds)
# Prints the time left on the timer
print(timer, end="\r")
# Delays the program one second
time.sleep(1)
# Reduces total time by one second
total_seconds -= 1
print("Bzzzt! The countdown is at zero seconds!")
# Inputs for hours, minutes, seconds on timer
h = input("Enter the time in hours: ")
m = input("Enter the time in minutes: ")
s = input("Enter the time in seconds: ")
countdown(int(h), int(m), int(s))
Enter the time in hours: 0
Enter the time in minutes: 0
Enter the time in seconds: 10
0:00:08
In [6]:
import time
from math import inf as infinity
#First step is to make a board. List of all the positions
game_board = ["*", "*", "*", "*", "*", "*", "*", "*", "*"]

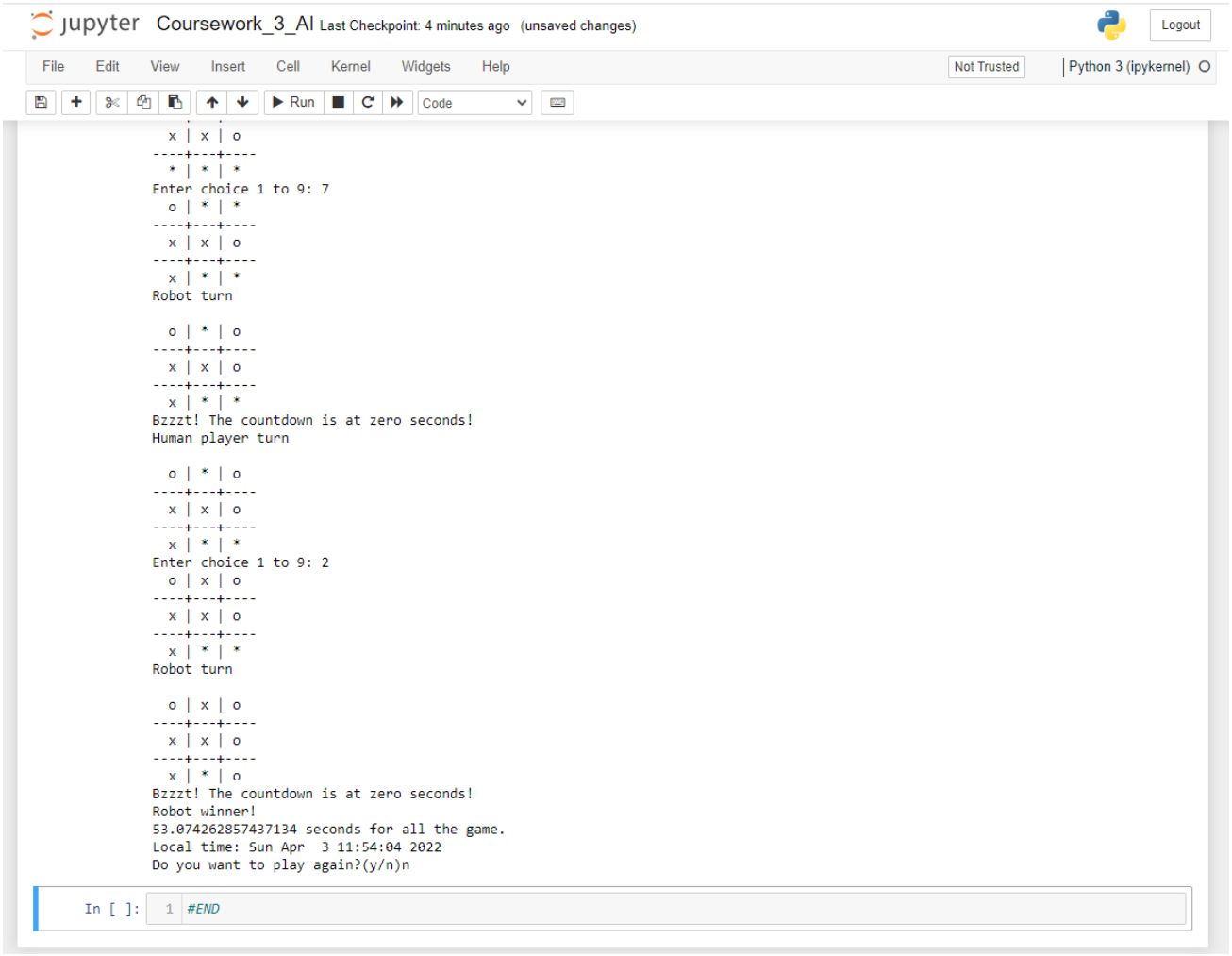

jupyter Coursework_3_AI Last Checkpoint: a minute ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [10]:
import time
import datetime
# Create class that acts as a countdown
def countdown(h, m, s):
# Calculate the total number of seconds
total_seconds = h * 3600 + m * 60 + s
# While loop that checks if total_seconds reaches zero
# If not zero, decrement total time by one second
while total_seconds > 0:
# Timer represents time left on countdown
timer = datetime.timedelta(seconds = total_seconds)
# Prints the time left on the timer
print(timer, end="\r")
# Delays the program one second
time.sleep(1)
# Reduces total time by one second
total_seconds -= 1
print("Bzzzt! The countdown is at zero seconds!")
# Inputs for hours, minutes, seconds on timer
h = input("Enter the time in hours: ")
m = input("Enter the time in minutes: ")
s = input("Enter the time in seconds: ")
countdown(int(h), int(m), int(s))
Enter the time in hours: 0
Enter the time in minutes: 0
Enter the time in seconds: 10
Bzzzt! The countdown is at zero seconds!
In [6]:
import time
from math import inf as infinity
#First step is to make a board. List of all the positions
game_board = ["*", "*", "*", "*", "*", "*", "*", "*", "*"]


jupyter Coursework_3_AI Last Checkpoint: 2 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
In [*]:
if __name__ == "__main__":
instructions_game()
play_again_question = True
while play_again_question:
start = 0
start = time.time()
main(game_board)
end = time.time()
#show how long take the game to finish
print(end - start,"seconds for all the game.")
#show the current time on screen. Day and hours, mins and secs.
local_time = time.ctime(seconds)
print("Local time:", local_time)
#after first game, board it reset the board.
game_board = ["*", "*", "*", "*", "*", "*", "*", "*", "*"]
play_again_question = input("Do you want to play again?(y/n)")
if play_again_question == "n":
#end of while loop
break
WELCOME TO TIC TAC TOE game!
The game is played on a grid that's 3 squares by 3 squares.
There are two players known as X and O.
Players alternate placing Xs and Os on the game board
The first player to get 3 of her marks in a row (up, down, across, or diagonally) is the winner.
When all 9 squares are full, the game is over (no one has three in a row).
Human player turn
* | * | *
---+---+---
* | * | *
---+---+---
* | * | *
Enter choice 1 to 9:
In [ ]:
#END


jupyter Coursework_3_AI Last Checkpoint: 2 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
#show the current time on screen. Day and hours, mins and secs.
local_time = time.ctime(seconds)
print("Local time:", local_time)
#after first game, board it reset the board.
game_board = ["*", "*", "*", "*", "*", "*", "*", "*", "*"]
play_again_question = input("Do you want to play again?(y/n)")
if play_again_question == "n":
#end of while loop
break
WELCOME TO TIC TAC TOE game!
The game is played on a grid that's 3 squares by 3 squares.
There are two players known as X and O.
Players alternate placing Xs and Os on the game board
The first player to get 3 of her marks in a row (up, down, across, or diagonally) is the winner.
When all 9 squares are full, the game is over (no one has three in a row).
Human player turn
* | * | *
---+---+---
* | * | *
---+---+---
* | * | *
Enter choice 1 to 9: 6
* | * | *
---+---+---
* | * | x
---+---+---
* | * | *
Robot turn
* | * | o
---+---+---
* | * | x
---+---+---
* | * | *
0:00:05
In [ ]:
#END


jupyter Coursework_3_AI Last Checkpoint: 2 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
Human player turn
* | * | *
---+---+---
* | * | *
---+---+---
* | * | *
Enter choice 1 to 9: 6
* | * | *
---+---+---
* | * | x
---+---+---
* | * | *
Robot turn
* | * | o
---+---+---
* | * | x
---+---+---
* | * | *
Bzzzt! The countdown is at zero seconds!
Human player turn
* | * | o
---+---+---
* | * | x
---+---+---
* | * | *
Enter choice 1 to 9: 5
* | * | o
---+---+---
* | x | x
---+---+---
* | * | *
Robot turn
* | * | o
---+---+---
o | x | x
---+---+---
* | * | *
0:00:07
In [ ]:
1 #END


jupyter Coursework_3_AI Last Checkpoint: 3 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
instructions_game()
play_again_question = True
while play_again_question:
start = 0
start = time.time()
main(game_board)
end = time.time()
#show how long take the game to finish
print(end - start,"seconds for all the game.")
#show the current time on screen. Day and hours, mins and secs.
local_time = time.ctime(seconds)
print("Local time:", local_time)
#after first game, board it reset the board.
game_board = ["*", "*", "*", "*", "*", "*", "*", "*", "*"]
play_again_question = input("Do you want to play again?(y/n)")
if play_again_question == "n":
#end of while loop
break
Bzzzt! The countdown is at zero seconds!
Human player turn
o | x | o
o | x | x
* | o | x
Enter choice 1 to 9: 7
o | x | o
o | x | x
x | o | x
It is a draw
86.78877711296082 seconds for all the game.
Local time: Sun Apr 3 11:54:04 2022
Do you want to play again?(y/n)
In [ ]:
#END


jupyter Coursework_3_AI Last Checkpoint: 4 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code
x | x | o
---+---+---
* | * | *
Enter choice 1 to 9: 7
o | * | *
---+---+---
x | x | o
---+---+---
x | * | *
Robot turn
o | * | o
---+---+---
x | x | o
---+---+---
x | * | *
Bzzzt! The countdown is at zero seconds!
Human player turn
o | * | o
---+---+---
x | x | o
---+---+---
x | * | *
Enter choice 1 to 9: 2
o | x | o
---+---+---
x | x | o
---+---+---
x | * | *
Robot turn
o | x | o
---+---+---
x | x | o
---+---+---
x | * | o
Bzzzt! The countdown is at zero seconds!
Robot winner!
53.074262857437134 seconds for all the game.
Local time: Sun Apr 3 11:54:04 2022
Do you want to play again?(y/n)n
In [ ]:
1 #END

• A brief reflection. It may include but is not limited to the discussions about the topics below:

o Have you met any difficulties/bugs during this coursework, and how did you tackle them?

The most difficult part was designing the minimax algorithm because it uses concepts like a tree, recursion, node, score, etc.

Another difficulty found was the design of the game table. In the beginning, I tried to make a coloured table script executing in a new window. This is difficult in Jupyter Notebook, and I just printed it in black and white on the console at the end.

o What did you think is the most difficult part of this coursework?

It is impossible to win the robot! Human player can only aspire to the draw. Without the Minimax algorithm, the robot sometimes chooses not optimal cells of the board and the human player can win. But this is not the case with this Tic Tac Toe game.

o What have you learnt from investigating this problem?

I learned about the minimax algorithm, decision tree and other algorithms like Alpha-Beta Pruning, but I did not use it here.

And I found an interesting Tic Tac Toe game on the website. It is called the "Ultimate Tic Tac Toe" game. https://ultimate-t3.herokuapp.com/. It is much harder to code it and play it.

o Which part(s) in your code design is your favourite and why?

My favourite part is the timer because it makes the game more interesting and challenging. It remembers me of chess playing between two human players in the past and I applied the timer concept to the Tic Tac Toe game.

o Which of the searching algorithms that you implemented is the best? Why?

There is only one algorithm, minimax. Reasons are given previously in this report. Another interesting algorithm that I could implement is "Alpha-Beta Pruning". This uses a heuristic value in the minimax algorithm to discard movements that are worse than the previously examined move. It is an algorithm to improve the minimax algorithm. https://stackabuse.com/minimax-and-alpha-beta-pruning-in-python/.

o Any references you might have used to complete the task.

Wikipedia, AI Lecture week 6, YouTube and Google. Links are provided previously in this report.

##########################END##########################