TIC TAC TOE GAME
Algorithm and Pseudocode

1. Define the board as per below position:

0	1	2
3	4	5
6	7	8

2. Find out if you are playing as "X" or "O" by checking the number of how many X's and O's on the board. If it's equal number, then you are playing as "X", if not then you are playing as "O". If you're playing as "X" then you will make the first move.

3. To win the game, you need to have three of the same symbols in a row, column or diagonal (i.e., 3 of X's or 3 of O's on the same row/column/diagonal). So, you need to place your agent in any of these 8 position combinations:
    a. Top row = 0, 1, 2
    b. Middle row = 3, 4, 5
    c. Bottom row = 6, 7, 8
    d. Left column = 0, 3, 6
    e. Middle column = 1, 4, 7
    f. Right column = 2, 5, 8
    g. Diagonal from top-left to bottom-right = 0, 4, 8
    h. Diagonal from top-right to bottom-left = 2, 4, 6

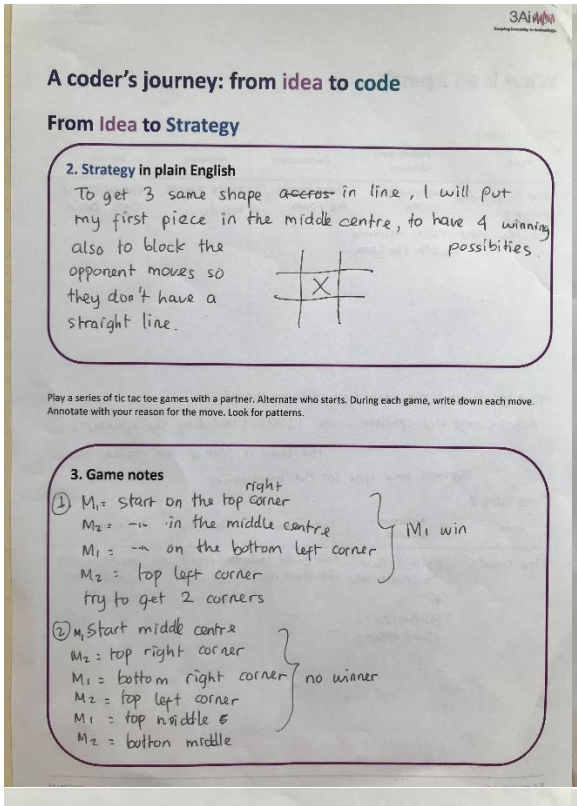
4. Also, the same strategy to block the opponent from winning the game if your opponent already has 2 out of 3, you need to place your agent on the 3rd position from any of these 8 possible combinations:
    a. Top row = 0, 1, 2
    b. Middle row = 3, 4, 5
    c. Bottom row = 6, 7, 8
    d. Left column = 0, 3, 6
    e. Middle column = 1, 4, 7
    f. Right column = 2, 5, 8
    g. Diagonal from top-left to bottom-right = 0, 4, 8
    h. Diagonal from top-right to bottom-left = 2, 4, 6

5. If the winning move if possible then take the winning move first then follow by blocking the opponent from winning as the next priority.

6. Then next priority is taking any of the corner and middle positions as per below:
    a. If position #0 (top-left corner) is empty, place your agent here.
    b. If position #2 (top-right corner) is empty, place your agent here.
    c. If position #4 (the middle square) is empty, place your agent here.
    d. If position #6 (bottom-left corner) is empty, place your agent here.
    e. If position #8 (bottom-right corner) is empty, place your agent here.

7. Otherwise, last priority is taking any available position on the board randomly.

Here are some of my notes from the session is class originally for idea, strategy, algorithm and pseudocode.



A coder's journey: from idea to code

From Idea to Strategy

2. Strategy in plain English

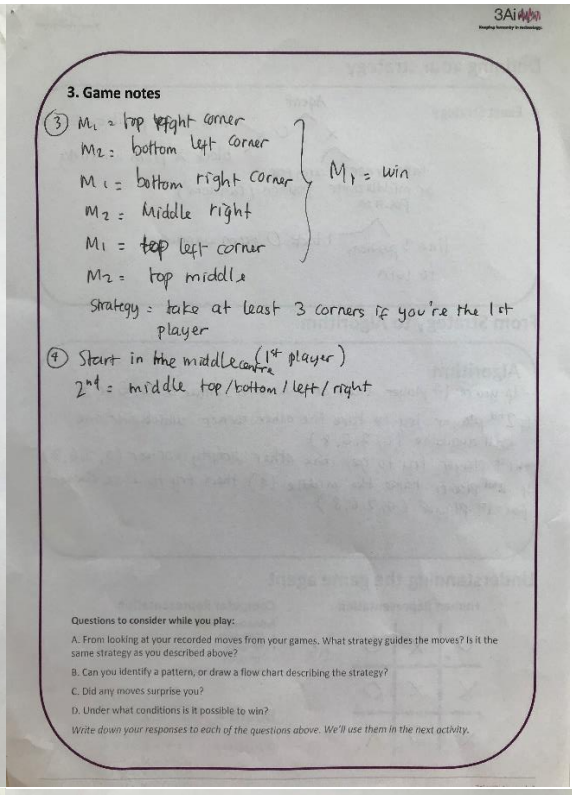
To get 3 same shape ~~across~~ in line, I will put my first piece in the middle centre, to have 4 winning possibilities.
also to block the opponent moves so they don't have a straight line.

Play a series of tic tac toe games with a partner. Alternate who starts. During each game, write down each move. Annotate with your reason for the move. Look for patterns.

3. Game notes

① M_1: start on the top right corner
M_2: " in the middle centre
M_1: " on the bottom left corner
M_2: top left corner
try to get 2 corners
} M_1 win

② M_1 start middle centre
M_2: top right corner
M_1: bottom right corner
M_2: top left corner
M_1: top middle
M_2: bottom middle
} no winner



3. Game notes

③ M_1 = top left corner
M_2: bottom left corner
M_1: bottom right corner
M_2: Middle right
M_1: top left corner
M_2: top middle
} M_1 = win
Strategy: take at least 3 corners if you're the 1st player

④ Start in the middle centre (1st player)
2nd: middle top / bottom / left / right

Questions to consider while you play:

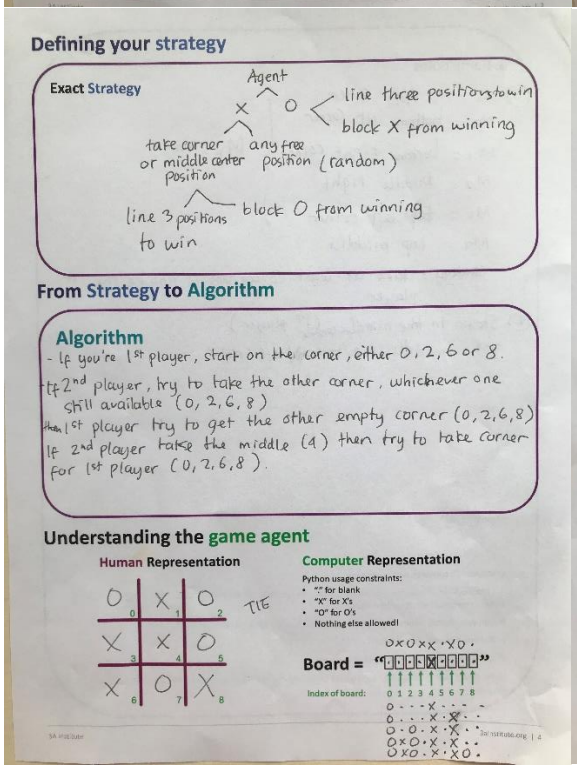
A. From looking at your recorded moves from your games. What strategy guides the moves? Is it the same strategy as you described above?

B. Can you identify a pattern, or draw a flow chart describing the strategy?

C. Did any moves surprise you?

D. Under what conditions is it possible to win?

Write down your responses to each of the questions above. We'll use them in the next activity.



Defining your strategy

Exact Strategy

Agent
X O
line three positions to win
block X from winning
take corner or middle center position
any free position (random)
line 3 positions to win
block O from winning

From Strategy to Algorithm

Algorithm

- If you're 1st player, start on the corner, either 0, 2, 6 or 8.
If 2nd player, try to take the other corner, whichever one still available (0, 2, 6, 8)
then 1st player try to get the other empty corner (0, 2, 6, 8)
If 2nd player take the middle (4) then try to take corner for 1st player (0, 2, 6, 8).

Understanding the game agent

Human Representation

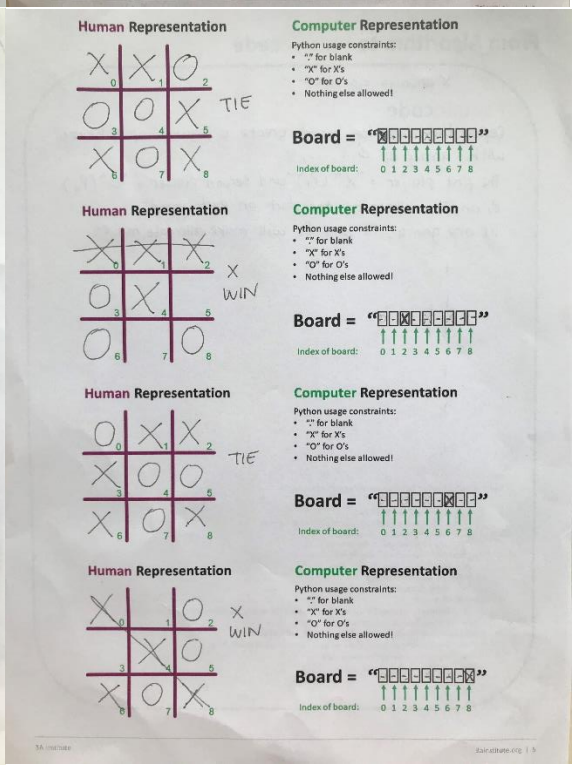
TIE

Computer Representation

Python usage constraints:
- "." for blank
- "X" for X's
- "O" for O's
- Nothing else allowed!

Board = "O-XX-O-X-O"
Index of board: 0 1 2 3 4 5 6 7 8

OXOXX·XO·
O··X···X·
O···X·X··
O·O·X·X··
OXO·X·XO·
OXO·X·XO·



Human Representation

TIE

Computer Representation

Python usage constraints:
- "." for blank
- "X" for X's
- "O" for O's
- Nothing else allowed!

Board = "X-------"
Index of board: 0 1 2 3 4 5 6 7 8

Human Representation

X WIN

Computer Representation

Python usage constraints:
- "." for blank
- "X" for X's
- "O" for O's
- Nothing else allowed!

Board = "--X------"
Index of board: 0 1 2 3 4 5 6 7 8

Human Representation

TIE

Computer Representation

Python usage constraints:
- "." for blank
- "X" for X's
- "O" for O's
- Nothing else allowed!

Board = "-------X-"
Index of board: 0 1 2 3 4 5 6 7 8

Human Representation

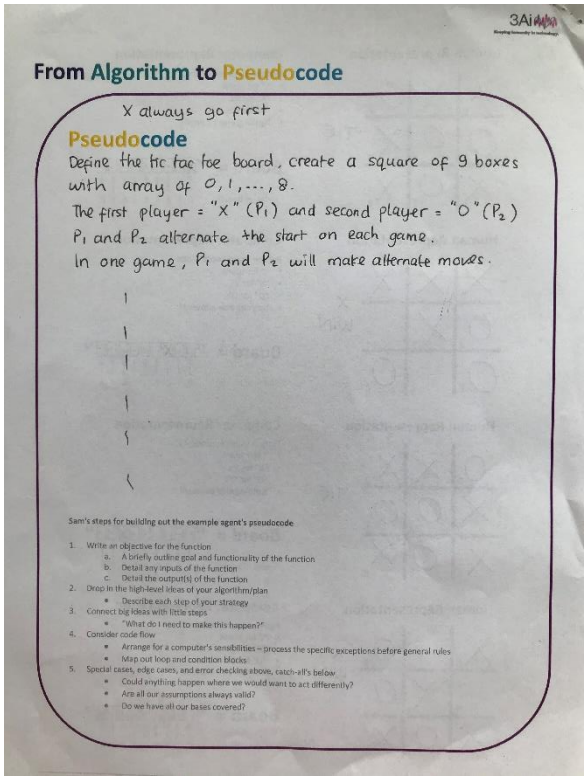
X WIN

Computer Representation

Python usage constraints:
- "." for blank
- "X" for X's
- "O" for O's
- Nothing else allowed!

Board = "--------X"
Index of board: 0 1 2 3 4 5 6 7 8

I didn't have a chance to complete this pseudocode draft in the studio during our session. For more details of my pseudocode, please see at the beginning of this document.



From Algorithm to Pseudocode

X always go first

Pseudocode

Define the tic tac toe board, create a square of 9 boxes with array of 0, 1, ..., 8.

The first player = "X" (P_1) and second player = "O" (P_2)

P_1 and P_2 alternate the start on each game.

In one game, P_1 and P_2 will make alternate moves.

Sam's steps for building out the example agent's pseudocode

1. Write an objective for the function
   a. A briefly outline goal and functionality of the function
   b. Detail any inputs of the function
   c. Detail the output(s) of the function
2. Drop in the high-level ideas of your algorithm/plan
   - Describe each step of your strategy
3. Connect big ideas with little steps
   - "What do I need to make this happen?"
4. Consider code flow
   - Arrange for a computer's sensibilities – process the specific exceptions before general rules
   - Map out loop and condition blocks
5. Special cases, edge cases, and error checking above, catch-all's below
   - Could anything happen where we would want to act differently?
   - Are all our assumptions always valid?
   - Do we have all our bases covered?

Through playing the games multiple times, I tested my pseudocode then codes, I changed the order of priority (#6) of the corner positions first and then middle centre after as shown below:

a. If position #0 (top-left corner) is empty, place your agent here.
b. If position #2 (top-right corner) is empty, place your agent here.
c. If position #6 (bottom-left corner) is empty, place your agent here.
d. If position #8 (bottom-right corner) is empty, place your agent here.
e. If position #4 (the middle square) is empty, place your agent here.

The order of the position priority has increased the probability of wins ~61%, losses ~33% and draws ~6% as shown below, so I'm pretty happy with my agent 😊

```
Playing winning_agent against all_possible_moves:
    losses: 69, draws: 14, wins: 129
    (total games played: 212)

    first game where you lost:
    (you were playing as X)
        X.O   X.O   X.O
        .X.   .X.   .X.
        ...   O..   O.X
```