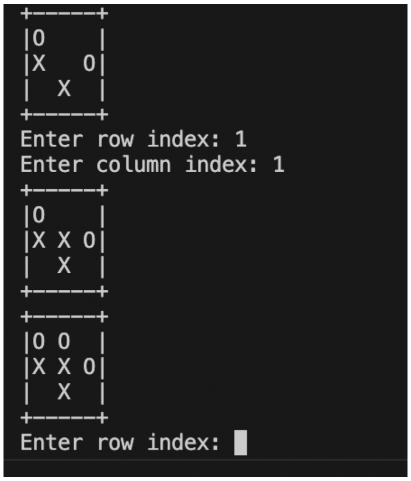
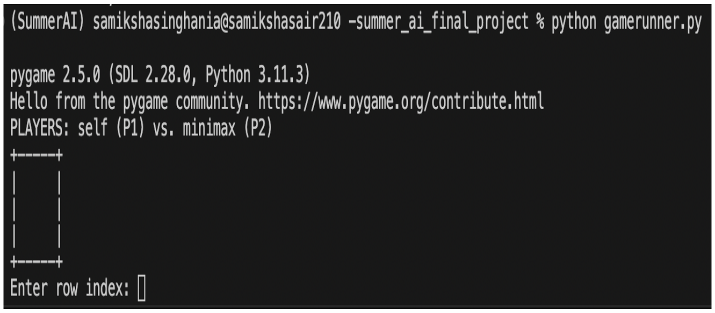
Reflection For Final

Emma and Samiksha

We worked on the Adversarial Search Problem. In this task we were supposed to implement a minimax or alpha-beta algorithm to evaluate a function for the game Tic-Tac-Toe and Connect4. We decided to use minimax as we thought that using minimax explored all paths unlike alpha-beta. We used 3 different functions in our code. We were a little stuck when we started as we were not familiar with what we had to do, but then we made a pseudocode which really helped us. Next we translated the pseudocode to a python code and we ran the projects a few times to see if there were any errors. We found a few so we debugged it. Lastly we played the game a few times to make sure the code worked perfectly. Our project works very well on Samiksha's laptop but since Emma has an old version of python it does not work very well. We figured it out and it runs in her terminal now. We think our methods were quite good. The code works perfectly. If we had more time we would make an alpha-beta algorithm for the connect 4 project as when you run it through minimax it is extremely slow.

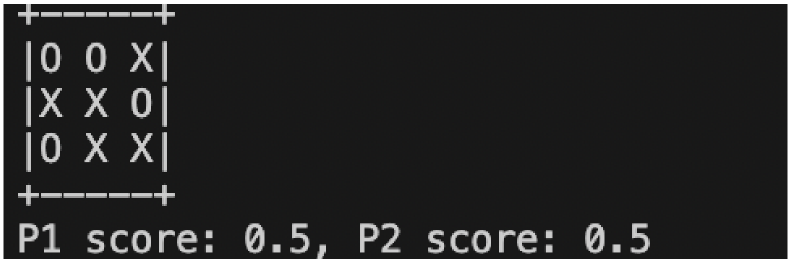


```
+-----+
|O    |
|X   O|
|  X  |
+-----+
Enter row index: 1
Enter column index: 1
+-----+
|O    |
|X X O|
|  X  |
+-----+
+-----+
|O O  |
|X X O|
|  X  |
+-----+
Enter row index: ▌
```



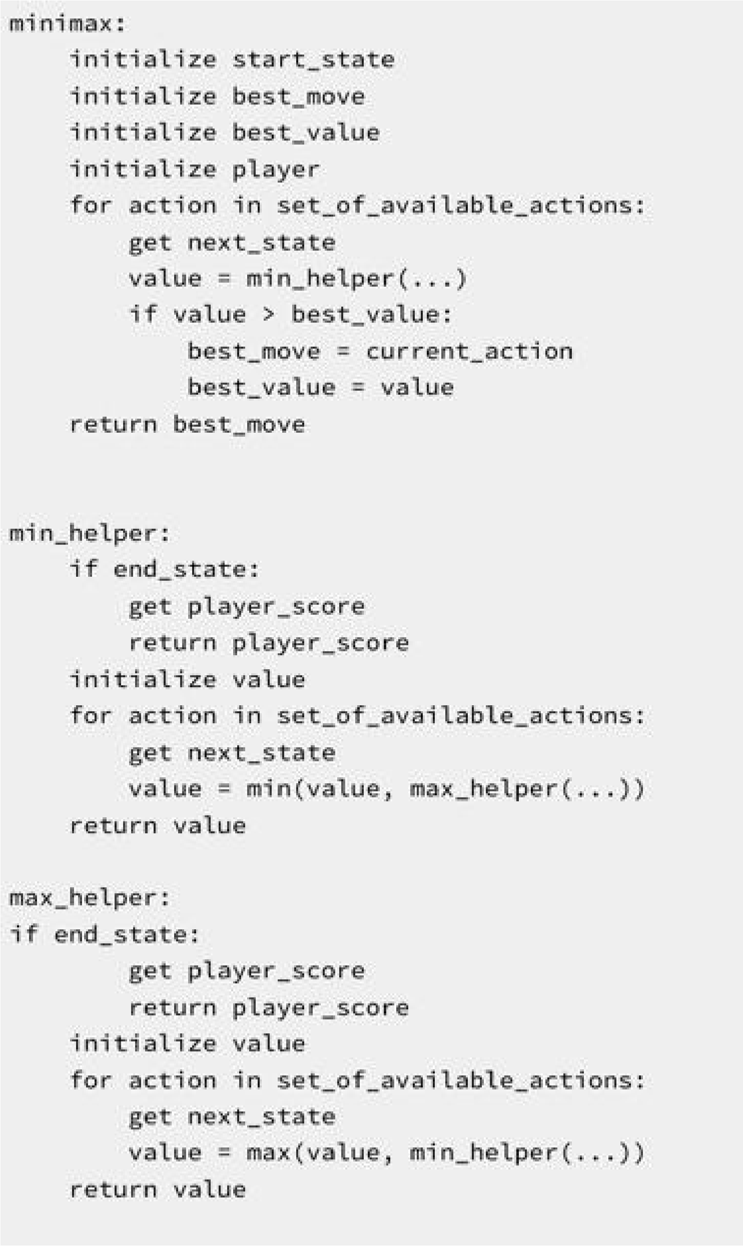
```
(SummerAI) samikshasinghania@samikshasair210 -summer_ai_final_project % python gamerunner.py

pygame 2.5.0 (SDL 2.28.0, Python 3.11.3)
Hello from the pygame community. https://www.pygame.org/contribute.html
PLAYERS: self (P1) vs. minimax (P2)
+-----+
|     |
|     |
|     |
+-----+
Enter row index: ▯
```



```
+-----+
|O O X|
|X X O|
|O X X|
+-----+
P1 score: 0.5, P2 score: 0.5
```

Pseudocode we used



```
minimax:
    initialize start_state
    initialize best_move
    initialize best_value
    initialize player
    for action in set_of_available_actions:
        get next_state
        value = min_helper(...)
        if value > best_value:
            best_move = current_action
            best_value = value
    return best_move


min_helper:
    if end_state:
        get player_score
        return player_score
    initialize value
    for action in set_of_available_actions:
        get next_state
        value = min(value, max_helper(...))
    return value

max_helper:
if end_state:
        get player_score
        return player_score
    initialize value
    for action in set_of_available_actions:
        get next_state
        value = max(value, min_helper(...))
    return value
```