Project 1

Question 1: Minimax Player

Play 100 games between Minimax and Random Player

```
Final Scoreboard:
Minimax Player wins 90/100 games
Random Player wins 0/100 games
Draws 10/100 games

Minimax Player average move duration: 0.19 seconds
Random Player average move duration: 0.00 seconds
Total evaluation time: 72.20 seconds
```

Play 100 games between Minimax and Minimax Player itself

```
Final Scoreboard:
Minimax Player wins 0/100 games
Minimax Player wins 0/100 games
Draws 100/100 games

Minimax Player average move duration: 0.15 seconds
Total evaluation time: 133.66 seconds
```

Play 10 games against human

```
Final Scoreboard:
Minimax Player wins 3/10 games
Human Player wins 0/10 games
Draws 7/10 games

Minimax Player average move duration: 0.16 seconds
Human Player average move duration: 3.35 seconds
Total evaluation time: 144.29 seconds
```

From three cases above, we can conclude that the minimax agent plays optimally.

Question 2: Alpha Beta Player

```
Final Scoreboard:
Alpha-Beta Player wins 0/20 games
Minimax Player wins 0/20 games
Draws 20/20 games

Alpha-Beta Player average move duration: 0.01 seconds
Minimax Player average move duration: 0.14 seconds
Total evaluation time: 14.35 seconds
```

Based on the observed final scoreboard, we can see that the alpha-beta agent is as good as the minimax player (draws 20/20 games).

When it comes to the runtime per move, I recorded the values as below:

```
The runtime per move 1 of agent Alpha-Beta Player is: 0.0009984970092773438
The runtime per move 2 of agent Alpha-Beta Player is: 0.01695418357849121
The runtime per move 3 of agent Alpha-Beta Player is: 0.003990650177001953
The runtime per move 4 of agent Alpha-Beta Player is: 0.0009968280792236328
The runtime per move 5 of agent Alpha-Beta Player is: 0.0
The runtime per move 6 of agent Alpha-Beta Player is: 0.0788264274597168
The runtime per move 7 of agent Alpha-Beta Player is: 0.008975505828857422
The runtime per move 8 of agent Alpha-Beta Player is: 0.001993417739868164
The runtime per move 9 of agent Alpha-Beta Player is: 0.0
The runtime per move 10 of agent Alpha-Beta Player is: 0.0
The runtime per move 11 of agent Alpha-Beta Player is: 0.03493928909301758
The runtime per move 12 of agent Alpha-Beta Player is: 0.001995563507080078
The runtime per move 13 of agent Alpha-Beta Player is: 0.0
The runtime per move 14 of agent Alpha-Beta Player is: 0.0
The runtime per move 15 of agent Alpha-Beta Player is: 0.055883407592773444
The runtime per move 16 of agent Alpha-Beta Player is: 0.002991914749145508
The runtime per move 17 of agent Alpha-Beta Player is: 0.0019910335540771484
The runtime per move 18 of agent Alpha-Beta Player is: 0.0
The runtime per move 19 of agent Alpha-Beta Player is: 0.0
The runtime per move 20 of agent Alpha-Beta Player is: 0.011966705322265625
The runtime per move 21 of agent Alpha-Beta Player is: 0.0009980201721191406
The runtime per move 22 of agent Alpha-Beta Player is: 0.0
The runtime per move 23 of agent Alpha-Beta Player is: 0.0
The runtime per move 24 of agent Alpha-Beta Player is: 0.06981229782104492
The runtime per move 25 of agent Alpha-Beta Player is: 0.007982254028320312
The runtime per move 26 of agent Alpha-Beta Player is: 0.0019958019256591797
The runtime per move 27 of agent Alpha-Beta Player is: 0.0
The runtime per move 28 of agent Alpha-Beta Player is: 0.0
The runtime per move 29 of agent Alpha-Beta Player is: 0.010972023010253906
The runtime per move 30 of agent Alpha-Beta Player is: 0.0029935836791992188
The runtime per move 31 of agent Alpha-Beta Player is: 0.0009975433349609375
The runtime per move 32 of agent Alpha-Beta Player is: 0.0
The runtime per move 33 of agent Alpha-Beta Player is: 0.058843374252319336
The runtime per move 34 of agent Alpha-Beta Player is: 0.00997161865234375
The runtime per move 35 of agent Alpha-Beta Player is: 0.0009980201721191406
The runtime per move 36 of agent Alpha-Beta Player is: 0.0
The runtime per move 37 of agent Alpha-Beta Player is: 0.0009973049163818359
The runtime per move 38 of agent Alpha-Beta Player is: 0.02991938591003418
The runtime per move 39 of agent Alpha-Beta Player is: 0.0029914379119873047
The runtime per move 40 of agent Alpha-Beta Player is: 0.0
The runtime per move 41 of agent Alpha-Beta Player is: 0.0009973049163818359
```

```
The runtime per move 42 of agent Alpha-Beta Player is: 0.074627876281738 28
The runtime per move 43 of agent Alpha-Beta Player is: 0.006981372833251953
The runtime per move 44 of agent Alpha-Beta Player is: 0.0019953250885009766
The runtime per move 45 of agent Alpha-Beta Player is: 0.0009984970092773438
The runtime per move 46 of agent Alpha-Beta Player is: 0.0009970664978027344
The runtime per move 47 of agent Alpha-Beta Player is: 0.034903049468994 14
The runtime per move 48 of agent Alpha-Beta Player is: 0.0020308494567871094
The runtime per move 49 of agent Alpha-Beta Player is: 0.0010001659393310547
The runtime per move 50 of agent Alpha-Beta Player is: 0.0
The runtime per move 51 of agent Alpha-Beta Player is: 0.049867868423461914
The runtime per move 52 of agent Alpha-Beta Player is: 0.0059833526611328125
The runtime per move 53 of agent Alpha-Beta Player is: 0.0009982585906982422
The runtime per move 54 of agent Alpha-Beta Player is: 0.0010013580322265625
The runtime per move 55 of agent Alpha-Beta Player is: 0.0019943714141845703
The runtime per move 56 of agent Alpha-Beta Player is: 0.0299222469329834
The runtime per move 57 of agent Alpha-Beta Player is: 0.0029897689819335938
The runtime per move 58 of agent Alpha-Beta Player is: 0.0
The runtime per move 59 of agent Alpha-Beta Player is: 0.0009927749633789062
The runtime per move 60 of agent Alpha-Beta Player is: 0.07383394241333008
The runtime per move 61 of agent Alpha-Beta Player is: 0.003988504409790039
The runtime per move 62 of agent Alpha-Beta Player is: 0.0009984970092773438
The runtime per move 63 of agent Alpha-Beta Player is: 0.0009968280792236328
The runtime per move 64 of agent Alpha-Beta Player is: 0.0
The runtime per move 65 of agent Alpha-Beta Player is: 0.020943880081176758
The runtime per move 66 of agent Alpha-Beta Player is: 0.0019943714141845703
The runtime per move 67 of agent Alpha-Beta Player is: 0.0
The runtime per move 68 of agent Alpha-Beta Player is: 0.0
The runtime per move 69 of agent Alpha-Beta Player is: 0.06287717819213867
The runtime per move 70 of agent Alpha-Beta Player is: 0.005984306335449219
The runtime per move 71 of agent Alpha-Beta Player is: 0.0009984970092773438
The runtime per move 72 of agent Alpha-Beta Player is: 0.0
The runtime per move 73 of agent Alpha-Beta Player is: 0.0009968280792236328
The runtime per move 74 of agent Alpha-Beta Player is: 0.026513099670410156
The runtime per move 75 of agent Alpha-Beta Player is: 0.0030319690704345703
The runtime per move 76 of agent Alpha-Beta Player is: 0.0
The runtime per move 77 of agent Alpha-Beta Player is: 0.0
The runtime per move 78 of agent Alpha-Beta Player is: 0.07779312133789062
The runtime per move 79 of agent Alpha-Beta Player is: 0.0049877166748046875
The runtime per move 80 of agent Alpha-Beta Player is: 0.0009961128234863281
The runtime per move 81 of agent Alpha-Beta Player is: 0.00099945068359375
The runtime per move 82 of agent Alpha-Beta Player is: 0.0
The runtime per move 83 of agent Alpha-Beta Player is: 0.044908761978149414
```

```
The runtime per move 84 of agent Alpha-Beta Player is: 0.004060268402099609
The runtime per move 85 of agent Alpha-Beta Player is: 0.0009980201721191406
The runtime per move 86 of agent Alpha-Beta Player is: 0.0
The runtime per move 87 of agent Alpha-Beta Player is: 0.08580660820007324
The runtime per move 88 of agent Alpha-Beta Player is: 0.011008024215698242
The runtime per move 89 of agent Alpha-Beta Player is: 0.0010211467742919922
The runtime per move 90 of agent Alpha-Beta Player is: 0.0
The runtime per move 1 of agent Minimax Player is: 1.1918187141418457
The runtime per move 2 of agent Minimax Player is: 0.03091883659362793
The runtime per move 3 of agent Minimax Player is: 0.0010473728179931641
The runtime per move 4 of agent Minimax Player is: 0.0009975433349609375
The runtime per move 5 of agent Minimax Player is: 0.0
The runtime per move 6 of agent Minimax Player is: 0.17058396339416504
The runtime per move 7 of agent Minimax Player is: 0.006981372833251953
The runtime per move 8 of agent Minimax Player is: 0.0010349750518798828
The runtime per move 9 of agent Minimax Player is: 0.0
The runtime per move 10 of agent Minimax Player is: 1.2303292751312256
The runtime per move 11 of agent Minimax Player is: 0.01695394515991211
The runtime per move 12 of agent Minimax Player is: 0.0009975433349609375
The runtime per move 13 of agent Minimax Player is: 0.0009975433349609375
The runtime per move 14 of agent Minimax Player is: 0.0
The runtime per move 15 of agent Minimax Player is: 0.15558791160583496
The runtime per move 16 of agent Minimax Player is: 0.008975505828857422
The runtime per move 17 of agent Minimax Player is: 0.0009868144989013672
The runtime per move 18 of agent Minimax Player is: 0.0
The runtime per move 19 of agent Minimax Player is: 1.5049109458923334
The runtime per move 20 of agent Minimax Player is: 0.020943641662597656
The runtime per move 21 of agent Minimax Player is: 0.0009965896606445312
The runtime per move 22 of agent Minimax Player is: 0.0
The runtime per move 23 of agent Minimax Player is: 0.0
The runtime per move 24 of agent Minimax Player is: 0.16459035873413086
The runtime per move 25 of agent Minimax Player is: 0.0029916763305664062
The runtime per move 26 of agent Minimax Player is: 0.0009987354278564453
The runtime per move 27 of agent Minimax Player is: 0.0
The runtime per move 28 of agent Minimax Player is: 1.2609548568725586
The runtime per move 29 of agent Minimax Player is: 0.026971817016601562
The runtime per move 30 of agent Minimax Player is: 0.001995086669921875
The runtime per move 31 of agent Minimax Player is: 0.0
The runtime per move 32 of agent Minimax Player is: 0.0
The runtime per move 33 of agent Minimax Player is: 0.18354272842407227
The runtime per move 34 of agent Minimax Player is: 0.006000995635986328
The runtime per move 35 of agent Minimax Player is: 0.0009975433349609375
The runtime per move 36 of agent Minimax Player is: 0.0
```

```
The runtime per move 37 of agent Minimax Player is: 1.4030001163482666
The runtime per move 38 of agent Minimax Player is: 0.02991938591003418
The runtime per move 39 of agent Minimax Player is: 0.0019960403442382812
The runtime per move 40 of agent Minimax Player is: 0.0
The runtime per move 41 of agent Minimax Player is: 0.0
The runtime per move 42 of agent Minimax Player is: 0.15262436866760254
The runtime per move 43 of agent Minimax Player is: 0.005012035369873047
The runtime per move 44 of agent Minimax Player is: 0.0009965896606445312
The runtime per move 45 of agent Minimax Player is: 0.0
The runtime per move 46 of agent Minimax Player is: 1.2472164630889893
The runtime per move 47 of agent Minimax Player is: 0.019948482513427734
The runtime per move 48 of agent Minimax Player is: 0.002026796340942383
The runtime per move 49 of agent Minimax Player is: 0.0
The runtime per move 50 of agent Minimax Player is: 0.0
The runtime per move 51 of agent Minimax Player is: 0.15561771392822266
The runtime per move 52 of agent Minimax Player is: 0.0060079097747802734
The runtime per move 53 of agent Minimax Player is: 0.0
The runtime per move 54 of agent Minimax Player is: 0.0
The runtime per move 55 of agent Minimax Player is: 1.293088674545288
The runtime per move 56 of agent Minimax Player is: 0.033904314041137695
The runtime per move 57 of agent Minimax Player is: 0.0019919872283935547
The runtime per move 58 of agent Minimax Player is: 0.0009999275207519531
The runtime per move 59 of agent Minimax Player is: 0.0009982585906982422
The runtime per move 60 of agent Minimax Player is: 0.13164734840393066
The runtime per move 61 of agent Minimax Player is: 0.0039882659912109375
The runtime per move 62 of agent Minimax Player is: 0.000997304916381836
The runtime per move 63 of agent Minimax Player is: 0.0009868144989013672
The runtime per move 64 of agent Minimax Player is: 1.4295177459716797
The runtime per move 65 of agent Minimax Player is: 0.02097320556640625
The runtime per move 66 of agent Minimax Player is: 0.002006053924560547
The runtime per move 67 of agent Minimax Player is: 0.0009996891021728516
The runtime per move 68 of agent Minimax Player is: 0.0
The runtime per move 69 of agent Minimax Player is: 0.15358829498291016
The runtime per move 70 of agent Minimax Player is: 0.005984067916870117
The runtime per move 71 of agent Minimax Player is: 0.0009970664978027344
The runtime per move 72 of agent Minimax Player is: 0.0
The runtime per move 73 of agent Minimax Player is: 1.3053555488586426
The runtime per move 74 of agent Minimax Player is: 0.016952991485595703
The runtime per move 75 of agent Minimax Player is: 0.0009961128234863281
The runtime per move 76 of agent Minimax Player is: 0.0
The runtime per move 77 of agent Minimax Player is: 0.0009963512420654297
The runtime per move 78 of agent Minimax Player is: 0.12668871879577637
The runtime per move 79 of agent Minimax Player is: 0.009002923965454102
```

```
The runtime per move 80 of agent Minimax Player is: 0.0
The runtime per move 81 of agent Minimax Player is: 0.0
The runtime per move 82 of agent Minimax Player is: 1.150510549545288
The runtime per move 83 of agent Minimax Player is: 0.029948949813842773
The runtime per move 84 of agent Minimax Player is: 0.0009965896606445312
The runtime per move 85 of agent Minimax Player is: 0.0
The runtime per move 86 of agent Minimax Player is: 0.0
The runtime per move 87 of agent Minimax Player is: 0.21043753623962402
The runtime per move 88 of agent Minimax Player is: 0.006014823913574219
The runtime per move 89 of agent Minimax Player is: 0.000997781753540039
The runtime per move 90 of agent Minimax Player is: 0.0
```

By comparing the average time per move of each agent, we can easily conclude that the alpha-beta player results in an improvement in time (0.01s compared to 0.14s of minimax player).

```
Alpha-Beta Player average move duration: 0.01 seconds
Minimax Player average move duration: 0.14 seconds
```

Question 3: MCTS

Rolls out 5000 times for the problem that have small branching factor (~ 2 per node) like 3x3 tic tac toe seems not a good idea because it leads to large runtime. Therefore, in this problem, I will reset the number of roll out to 50 times. This will slightly affect the result but we will have better runtime.

For example, the record below is the result when I run 100 games of MCTS player and Random player with 5000 simulations (~ 5000 rolls out). We have a quite good performance, but ~ 6 minutes of runtime

```
Final Scoreboard:
MCTS Player wins 96/100 games
Random Player wins 0/100 games
Draws 4/100 games

MCTS Player average move duration: 0.97 seconds
Random Player average move duration: 0.00 seconds
Total evaluation time: 312.81 seconds
```

In the record below, I tried to reduce the number of roll out from 5000 to 50 and have an amazing runtime with the performance not too bad.

```
Final Scoreboard:
MCTS Player wins 88/100 games
Random Player wins 1/100 games
Draws 11/100 games

MCTS Player average move duration: 0.01 seconds
Random Player average move duration: 0.00 seconds
Total evaluation time: 5.67 seconds
```

MCTS vs Alphabeta ~ 100 games

```
Final Scoreboard:
MCTS Player wins 0/100 games
Alpha-Beta Player wins 24/100 games
Draws 76/100 games

MCTS Player average move duration: 0.01 seconds
Alpha-Beta Player average move duration: 0.01 seconds
Total evaluation time: 10.66 seconds
```

MCTS vs Alphabeta ~ 10000 games

```
Final Scoreboard:
MCTS Player wins 0/10000 games
Alpha-Beta Player wins 3064/10000 games
Draws 6936/10000 games

MCTS Player average move duration: 0.01 seconds
Alpha-Beta Player average move duration: 0.01 seconds
Total evaluation time: 1154.93 seconds
```

The number of simulations leads to the more accurate result (according to the law of large number). In this case, the mcts player has 69.36% to draw against Alpha-beta player compared to 76% in the case of 100 games.

How does c value affect the result:

c = 1

```
Final Scoreboard:
MCTS Player wins 0/100 games
Alpha-Beta Player wins 28/100 games
Draws 72/100 games

MCTS Player average move duration: 0.01 seconds
Alpha-Beta Player average move duration: 0.01 seconds
Total evaluation time: 10.88 seconds
```

c = 2

```
Final Scoreboard:
MCTS Player wins 0/100 games
Alpha-Beta Player wins 26/100 games
Draws 74/100 games

MCTS Player average move duration: 0.01 seconds
Alpha-Beta Player average move duration: 0.01 seconds
Total evaluation time: 11.04 seconds
```

c = 9

```
Final Scoreboard:
MCTS Player wins 0/100 games
Alpha-Beta Player wins 53/100 games
Draws 47/100 games

MCTS Player average move duration: 0.01 seconds
Alpha-Beta Player average move duration: 0.01 seconds
Total evaluation time: 9.90 seconds
```

- In the case $c = 9$, the performance of MCTS is worse than the case $c = 1$, and $c = 2$. The reason for that is because when we increase the value of c, it lead to a broader search but at the risk of less focus on promising areas. A lower value of c emphasizes exploitation, focusing more on moves that have shown promise so far, which can speed up convergence but risks missing potentially better moves.
- And since the branching factor of this problem is not large, we can adjust the value of c to be $small$ enough to optimize the exploitation rather than the exploration.

Is MCTS a good policy for Gomoku?

The default rollout policy in many MCTS implementations is to play moves randomly until the game ends. While this is computationally simple and often surprisingly effective, it can be suboptimal, especially in complex games like Gomoku, where strategic depth is significant.

Alternative Rollout Strategies:

1. **Heuristic-Based Simulations**:

- Use simple heuristics to guide the selection of moves during the rollout phase. For instance, in Gomoku, you might prioritize moves that immediately block an opponent's four-in-a-row or extend your own rows.
- Heuristics can be designed to capture basic strategic elements of the game without requiring the depth of full evaluation.

2. **Domain-Specific Policies**:

- Develop a lightweight, domain-specific policy for making decisions during rollouts. This could be based on simpler versions of the full evaluation function used for strategic play.

3. **Machine Learning Models**:

- Train a lightweight machine learning model to predict the next best move or to evaluate board states during the rollout. This can be particularly effective if you can use data from expert games to train the model.

4. **Limited Depth Search**:

- Instead of playing completely randomly, use a limited depth minimax search with a basic evaluation function to decide moves during rollouts. This approach bridges the gap between naive random play and computationally expensive full game tree searches.

5. **Capture and Threat-Based Moves**:

- Prioritize moves that involve capturing stones or setting up/tackling immediate threats. This strategy aligns with key tactics in Gomoku and can be more effective than random play.

Question 4 - Tabular Q-Learning

MDP for the problem

- **States** S: All combinations of the 'X' and 'O' in the board.
- **Actions** A: Put 'X' or 'O' in an empty cell.
- **Transition** P: contains no uncertainty ~ $= 1$ because each player's action can only lead to 1 possible resulted state.
- **Reward** R: +1 for the win, -1 for the loses, 0 for the draw game.

Record 100 games with other agents ~ Num of episode = 200000

QLearning vs Alphabeta/Minimax

```
Final Scoreboard:
Q-Learning Player wins 0/100 games
Alpha-Beta Player wins 29/100 games
Draws 71/100 games

Q-Learning Player average move duration: 0.00 seconds
Alpha-Beta Player average move duration: 0.01 seconds
Total evaluation time: 7.98 seconds
```

QLearning vs MCTS

```
Final Scoreboard:
Q-Learning Player wins 29/100 games
MCTS Player wins 26/100 games
Draws 45/100 games

Q-Learning Player average move duration: 0.00 seconds
MCTS Player average move duration: 0.01 seconds
Total evaluation time: 6.13 seconds
```

QLearning vs QLearning

```
Final Scoreboard:
Q-Learning Player wins 0/100 games
Q-Learning Player wins 0/100 games
Draws 100/100 games

Q-Learning Player average move duration: 0.00 seconds
Total evaluation time: 3.63 seconds
```

QLearning vs Random

```
Final Scoreboard:
Q-Learning Player wins 88/100 games
Random Player wins 4/100 games
Draws 8/100 games

Q-Learning Player average move duration: 0.00 seconds
Random Player average move duration: 0.00 seconds
Total evaluation time: 2.72 seconds
```

Change transfer player to Random player

We just adjust the class TT_QPlayer(Player) to TT_QPlayer(RandomPlayer)

QLearning vs Alphabeta/Minimax

```
Final Scoreboard:
Q-Learning Player wins 0/100 games
Alpha-Beta Player wins 21/100 games
Draws 79/100 games

Q-Learning Player average move duration: 0.00 seconds
Alpha-Beta Player average move duration: 0.01 seconds
Total evaluation time: 6.40 seconds
```

QLearning vs MCTS

```
Final Scoreboard:
Q-Learning Player wins 16/100 games
MCTS Player wins 25/100 games
Draws 59/100 games

Q-Learning Player average move duration: 0.00 seconds
MCTS Player average move duration: 0.01 seconds
Total evaluation time: 6.71 seconds
```

QLearning vs QLearning

```
Final Scoreboard:
Q-Learning Player wins 0/100 games
Q-Learning Player wins 0/100 games
Draws 100/100 games

Q-Learning Player average move duration: 0.00 seconds
Total evaluation time: 3.63 seconds
```

QLearning vs Random

```
Final Scoreboard:
Q-Learning Player wins 84/100 games
Random Player wins 4/100 games
Draws 12/100 games

Q-Learning Player average move duration: 0.00 seconds
Random Player average move duration: 0.00 seconds
Total evaluation time: 2.04 seconds
```

From those cases above, we can see that if we adjust the transfer player to random player, there is only slight change in the result, which is not significant. Therefore, we can confidently conclude

that by changing the transfer player to random player, we still cannot generalize for other players.

Hyper parameter tuning:

Adjust the number of episodes

```
NUM_EPISODES = 1000000
LEARNING_RATE = 0.5
DISCOUNT_FACTOR = 0.9
EXPLORATION_RATE = 0.1
```

By adjusting the number of episodes, we increase a large number of runtime

```
Training Q Player [X] for 1000000 episodes...
  0%|                                                                                | 4527/1000000 [00:01<04:09, 3983.73it/s]
```

Therefore, it will probably lead to the "smarter" agent. Here, I will only compare it to the minimax agent, which is the smartest agent that has a dominant performance compared to other agents, to show that the performance has been increased when we increase the number of episodes.

```
Final Scoreboard:
Q-Learning Player wins 0/100 games
Alpha-Beta Player wins 0/100 games
Draws 100/100 games

Q-Learning Player average move duration: 0.00 seconds
Alpha-Beta Player average move duration: 0.02 seconds
Total evaluation time: 12.57 seconds
```

Default settings

```
NUM_EPISODES = 200000
LEARNING_RATE = 0.5
DISCOUNT_FACTOR = 0.9
EXPLORATION_RATE = 0.1
```

```
Training Q Player [X] for 200000 episodes...
  2%|▍                                                                              | 3748/200000 [00:00<00:50, 3886.91it/s]
```

Adjust the learning rate

```
NUM_EPISODES = 200000
LEARNING_RATE = 0.9
DISCOUNT_FACTOR = 0.9
EXPLORATION_RATE = 0.1
```

```
Training Q Player [X] for 200000 episodes...
  7%|▉                                                                              | 13449/200000 [00:03<00:51, 3654.35it/
```

Adjusting the learning rate doesn't have a significant change in the training time

Adjust the discount factor

```
NUM_EPISODES = 200000
LEARNING_RATE = 0.5
DISCOUNT_FACTOR = 0.2
EXPLORATION_RATE = 0.1
```

```
Training Q Player [X] for 200000 episodes...
  3%|                                                                                | 5574/200000 [00:01<00:47, 4079.24it/s]
```

Adjusting the discount factor doesn't have a significant change in the training time

Adjust the exploration rate

```
NUM_EPISODES = 200000
LEARNING_RATE = 0.5
DISCOUNT_FACTOR = 0.9
EXPLORATION_RATE = 0.5
```

```
Training Q Player [X] for 200000 episodes...
  3%|                                                                                | 5060/200000 [00:01<00:46, 4186.14it/s]
```

Adjusting the exploration rate doesn't have a significant change in the training time

And since there is no significant change in runtime, it doesn't cause any significant change in the performance.