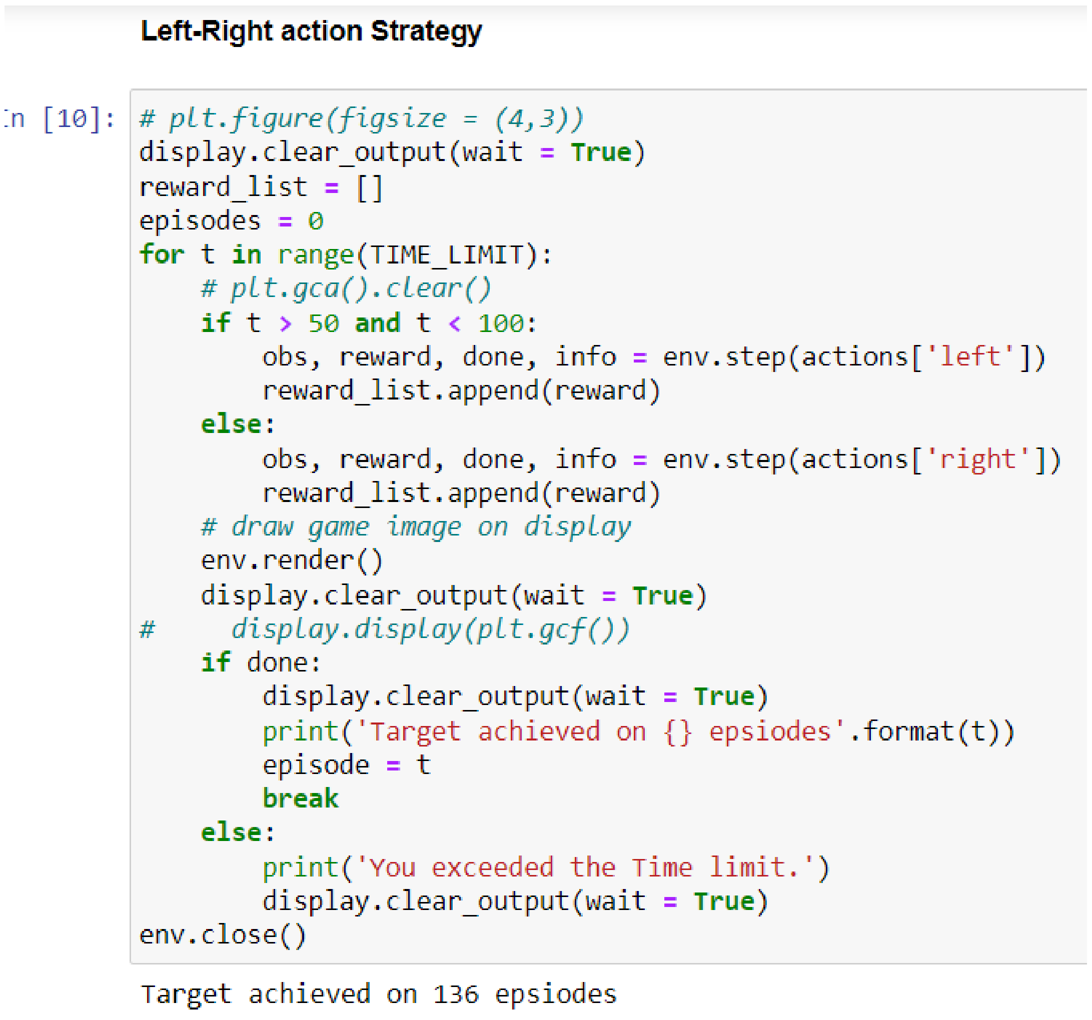
Exercise: 1

First Strategy: Left-Right Approach

The "Mountain Car" problem in OpenAI Gym is a classic problem in reinforcement learning where an underpowered car must drive up a steep hill. One approach to solving this problem is to use a "left-right" approach, where the car is trained to move left or right-based randomly.

I applied action for the first few iterations to move the car right and left. Applying such actions for many episodes, the car gained enough velocity to reach the flag.



Left-Right action Strategy

```
:n [10]: # plt.figure(figsize = (4,3))
         display.clear_output(wait = True)
         reward_list = []
         episodes = 0
         for t in range(TIME_LIMIT):
             # plt.gca().clear()
             if t > 50 and t < 100:
                 obs, reward, done, info = env.step(actions['left'])
                 reward_list.append(reward)
             else:
                 obs, reward, done, info = env.step(actions['right'])
                 reward_list.append(reward)
             # draw game image on display
             env.render()
             display.clear_output(wait = True)
         #     display.display(plt.gcf())
             if done:
                 display.clear_output(wait = True)
                 print('Target achieved on {} epsiodes'.format(t))
                 episode = t
                 break
             else:
                 print('You exceeded the Time limit.')
                 display.clear_output(wait = True)
         env.close()

         Target achieved on 136 epsiodes
```

Using this method, our car reached the flag on the 136 episodes.

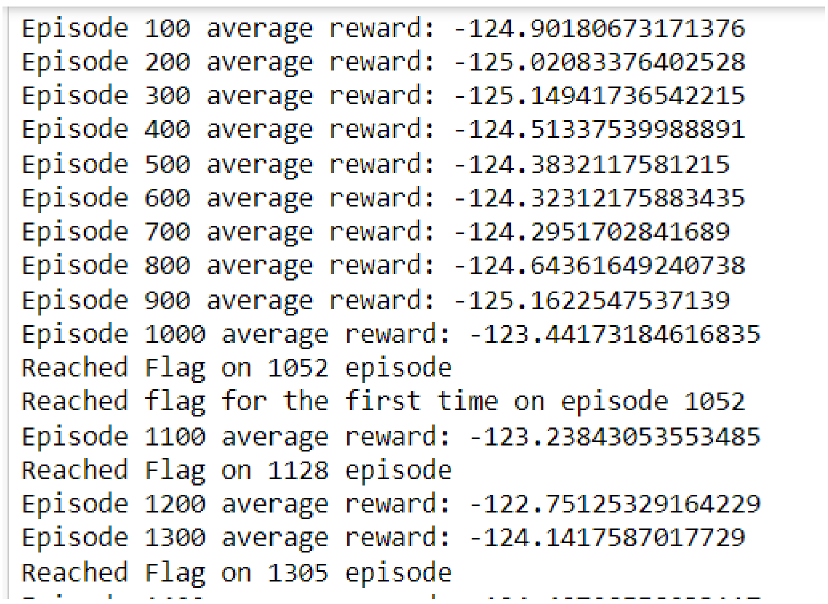
Second Strategy: Q-Learning

I implemented a Q-Learning function that learns to take actions that maximize the expected cumulative reward over time. It estimates the expected cumulative reward for taking a specific action in a specific state and then chooses the action that maximizes this Q-value. Each action, such as moving left or right, results in a new state and a corresponding reward based on the car's position and velocity.

The Q-function is represented by a table, which is updated based on the agent's experiences. The agent continues to explore the environment and update its Q-function until it reaches a satisfactory level of performance. Q-learning has been shown to be effective in solving the Mountain Car problem, but it can be computationally expensive and requires a lot of exploration data.

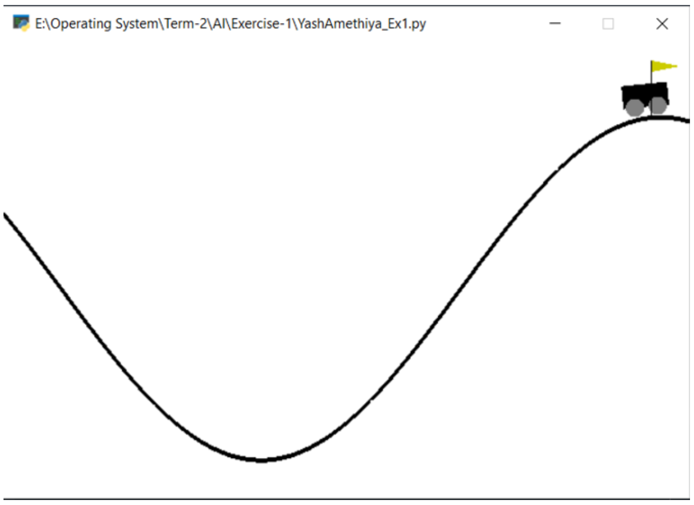
Also, here, I have used different rewarding methods. This new reward function which I have used instead of default values -1 for poor performance and +1 for correct action. This reward function rewards the machine for going right and the value varies between -1 and 0. When it reaches the flag it rewards 2.

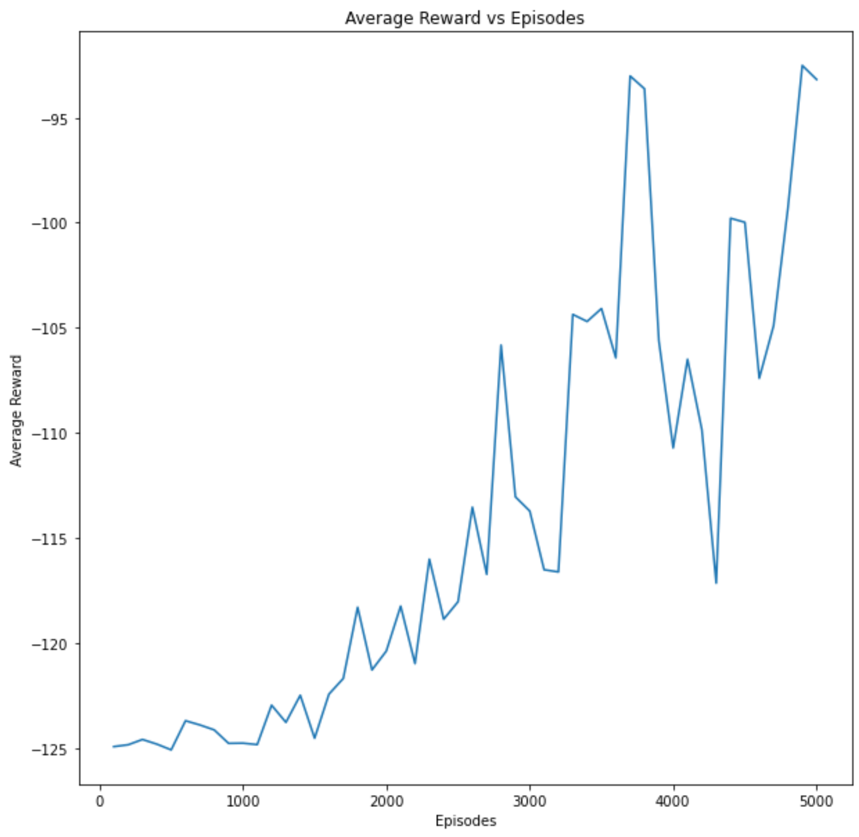
Output:



```
Episode 100 average reward: -124.90180673171376
Episode 200 average reward: -125.02083376402528
Episode 300 average reward: -125.14941736542215
Episode 400 average reward: -124.51337539988891
Episode 500 average reward: -124.3832117581215
Episode 600 average reward: -124.32312175883435
Episode 700 average reward: -124.2951702841689
Episode 800 average reward: -124.64361649240738
Episode 900 average reward: -125.1622547537139
Episode 1000 average reward: -123.44173184616835
Reached Flag on 1052 episode
Reached flag for the first time on episode 1052
Episode 1100 average reward: -123.23843053553485
Reached Flag on 1128 episode
Episode 1200 average reward: -122.75125329164229
Episode 1300 average reward: -124.1417587017729
Reached Flag on 1305 episode
```

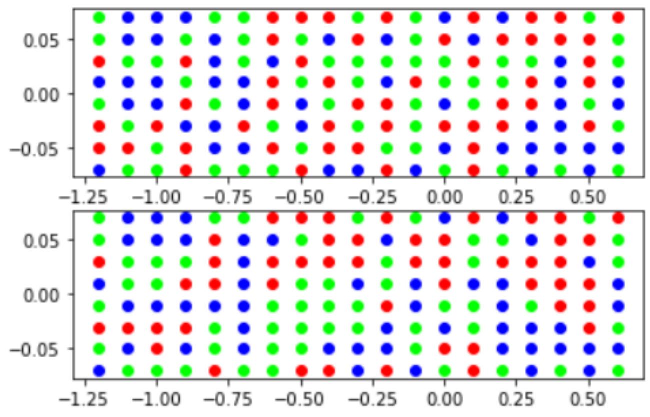
As we can see, in the above image the average is decreasing slowly and our car reached the flag for the first time on the 1052 episode.


E:\Operating System\Term-2\AI\Exercise-1\YashAmethiya_Ex1.py



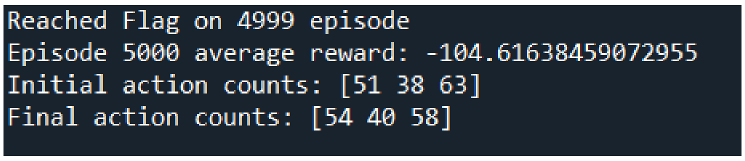
Average Reward vs Episode. At last, our average reward was around   -93 which is really good.

```
Initial action counts: [52 51 49]
Final action counts: [53 48 51]
```



- lime is for left action
- blue is for doing nothing
- red is for right action

This is the Q-table. The initial actions are randomly assigned and were almost the same and then after the model learns and updates the table and actions.



```
Reached Flag on 4999 episode
Episode 5000 average reward: -104.61638459072955
Initial action counts: [51 38 63]
Final action counts: [54 40 58]
```

No doubt, Q-learning is better than the left-right approach because Q-learning has the ability to make decisions for the next action by learning.