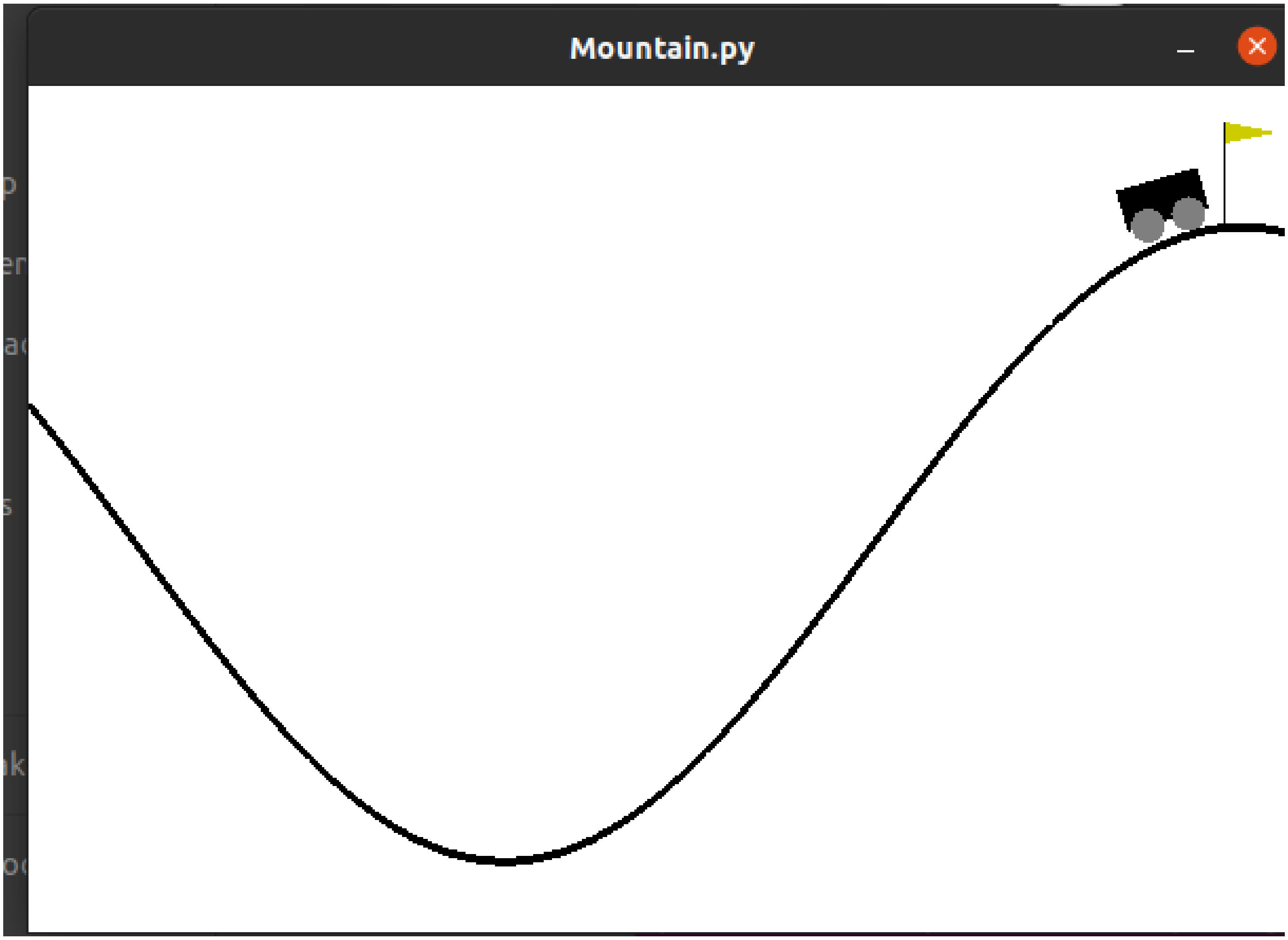
Machine Learning Lab Assignment 5

NAME – Rupayan Ghosh
ROLL NO. - 001811001015
Dept. - Information Technology

Mountain Climbing

Output:-
```
$ python3 Mountain.py
Mountain.py:32: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  return tuple(discrete_state.astype(np.int))
episode: 200, avg: -200.0, min: -200.0, max: -200.0
episode: 400, avg: -200.0, min: -200.0, max: -200.0
episode: 600, avg: -199.905, min: -200.0, max: -190.0
episode: 800, avg: -198.19, min: -200.0, max: -156.0
episode: 1000, avg: -194.595, min: -200.0, max: -157.0
episode: 1200, avg: -197.55, min: -200.0, max: -162.0
episode: 1400, avg: -193.87, min: -200.0, max: -151.0
episode: 1600, avg: -174.86, min: -200.0, max: -142.0
episode: 1800, avg: -191.94, min: -200.0, max: -141.0
episode: 2000, avg: -191.0, min: -200.0, max: -150.0
best reward: -141.0
best episode: 1756
```


−140
−150
−160
−170
−180
−190
−200
250
500
750
1000
1250
1500
1750
2000
avg
min
max



Code:-

```
import gym
import matplotlib.pyplot as plt
import numpy as np

env = gym.make("MountainCar-v0")
env.reset()

LEARNING_RATE = 0.1
DISCOUNT = 0.95
EPISODES = 2000

SHOW_EVERY = 200
RENDER_EVERY = 100

DISCRETE_OS_SIZE = [20] * len(env.observation_space.high)
```

```
discrete_os_win_size = (env.observation_space.high - env.observation_space.low) / DISCRETE_OS_SIZE

epsilon = 0.2
START_EPSILON_DECAYING = 1
END_EPSILON_DECAYING = EPISODES // 2

epsilon_decay_value = epsilon/(END_EPSILON_DECAYING - START_EPSILON_DECAYING)

q_table = np.random.uniform(low=-2, high=0, size=(DISCRETE_OS_SIZE + [env.action_space.n]))

ep_rewards = []
aggr_ep_rewards = {'ep': [], 'avg': [], 'min': [], 'max': []}


def get_discrete_state(state):
	discrete_state = (state - env.observation_space.low) / discrete_os_win_size
	return tuple(discrete_state.astype(np.int))


best_episode_reward = -200
best_episode = -1
best_q_table = q_table
for episode in range(1, EPISODES + 1):
	episode_success = False
	episode_reward = 0
	if not episode % SHOW_EVERY:
	try:
	average_reward = sum(ep_rewards[-SHOW_EVERY:])/len(ep_rewards[-SHOW_EVERY:])
	aggr_ep_rewards['ep'].append(episode)
	aggr_ep_rewards['avg'].append(average_reward)
	aggr_ep_rewards['min'].append(min(ep_rewards[-SHOW_EVERY:]))
	aggr_ep_rewards['max'].append(max(ep_rewards[-SHOW_EVERY:]))
	print(f"episode: {aggr_ep_rewards['ep'][-1]}, avg: {aggr_ep_rewards['avg'][-1]}, "
		+ f"min: {aggr_ep_rewards['min'][-1]}, max: {aggr_ep_rewards['max'][-1]}")
	except Exception as e:
	print(e)

	discrete_state = get_discrete_state(env.reset())
	done = False
	while not done:
	if np.random.random() > epsilon:
	action = np.argmax(q_table[discrete_state])
	else:
	action = np.random.randint(0, env.action_space.n)
```

```
        new_state, reward, done, _ = env.step(action)
        episode_reward += reward
        new_discrete_state = get_discrete_state(new_state)
        if RENDER_EVERY > 1:
        if not episode % RENDER_EVERY:
                env.render()
        if not done:
        max_future_q = np.max(q_table[new_discrete_state])
        current_q = q_table[discrete_state + (action, )]
        new_q = (1 - LEARNING_RATE) * current_q + LEARNING_RATE * (reward + DISCOUNT *
max_future_q)
        q_table[discrete_state + (action, )] = new_q
        elif new_state[0] >= env.goal_position:
        episode_success = True
        q_table[discrete_state + (action, )] = 0

        discrete_state = new_discrete_state

        if END_EPSILON_DECAYING >= episode >= START_EPSILON_DECAYING:
        epsilon -= epsilon_decay_value

        if episode_reward >= best_episode_reward:
        best_episode_reward = episode_reward
        best_episode = episode
        best_q_table = q_table
        ep_rewards.append(episode_reward)
        env.close()

print(f"best reward: {best_episode_reward}")
print(f"best episode: {best_episode}")

plt.plot(aggr_ep_rewards['ep'], aggr_ep_rewards['avg'], label="avg")
plt.plot(aggr_ep_rewards['ep'], aggr_ep_rewards['min'], label="min")
plt.plot(aggr_ep_rewards['ep'], aggr_ep_rewards['max'], label="max")
plt.legend(loc=4)
plt.show()
```

Roulette

Output:-
$ python3 Roulette.py
Number of actions available: 38
Number of states defined: 1
Therefore, Q-Table with 38 columns and 1 rows will be created
Q-Table shape: (1, 38)

****
Episode: 2280
Round: 1
Action: 33
Reward: 1.0
Total reward so far: 1.0

****
Episode: 2280
Round: 2
Action: 33
Reward: 1.0
Total reward so far: 2.0

****
Episode: 2280
Round: 3
Action: 33
Reward: 1.0
Total reward so far: 3.0

****
Episode: 2280
Round: 4
Action: 33
Reward: 1.0
Total reward so far: 4.0

****
Episode: 2280
Round: 5
Action: 33
Reward: 1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 6
Action: 33
Reward: -1.0
Total reward so far: 4.0

****
Episode: 2280
Round: 7
Action: 33
Reward: -1.0
Total reward so far: 3.0

****
Episode: 2280
Round: 8
Action: 33
Reward: 1.0
Total reward so far: 4.0

****
Episode: 2280
Round: 9
Action: 33
Reward: 1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 10
Action: 33
Reward: 1.0
Total reward so far: 6.0

****
Episode: 2280
Round: 11
Action: 33
Reward: -1.0
Total reward so far: 5.0

****
Episode: 2280

Round: 12
Action: 33
Reward: -1.0
Total reward so far: 4.0

****
Episode: 2280
Round: 13
Action: 16
Reward: 1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 14
Action: 33
Reward: -1.0
Total reward so far: 4.0

****
Episode: 2280
Round: 15
Action: 33
Reward: -1.0
Total reward so far: 3.0

****
Episode: 2280
Round: 16
Action: 33
Reward: -1.0
Total reward so far: 2.0

****
Episode: 2280
Round: 17
Action: 16
Reward: 1.0
Total reward so far: 3.0

****
Episode: 2280
Round: 18
Action: 16

Reward: 1.0
Total reward so far: 4.0

****
Episode: 2280
Round: 19
Action: 16
Reward: -1.0
Total reward so far: 3.0

****
Episode: 2280
Round: 20
Action: 16
Reward: -1.0
Total reward so far: 2.0

****
Episode: 2280
Round: 21
Action: 16
Reward: 1.0
Total reward so far: 3.0

****
Episode: 2280
Round: 22
Action: 16
Reward: 1.0
Total reward so far: 4.0

****
Episode: 2280
Round: 23
Action: 16
Reward: 1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 24
Action: 16
Reward: 1.0
Total reward so far: 6.0

****
Episode: 2280
Round: 25
Action: 16
Reward: -1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 26
Action: 16
Reward: 1.0
Total reward so far: 6.0

****
Episode: 2280
Round: 27
Action: 16
Reward: -1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 28
Action: 16
Reward: 1.0
Total reward so far: 6.0

****
Episode: 2280
Round: 29
Action: 16
Reward: 1.0
Total reward so far: 7.0

****
Episode: 2280
Round: 30
Action: 16
Reward: 1.0
Total reward so far: 8.0

****

Episode: 2280
Round: 31
Action: 16
Reward: -1.0
Total reward so far: 7.0

****
Episode: 2280
Round: 32
Action: 16
Reward: -1.0
Total reward so far: 6.0

****
Episode: 2280
Round: 33
Action: 16
Reward: 1.0
Total reward so far: 7.0

****
Episode: 2280
Round: 34
Action: 16
Reward: -1.0
Total reward so far: 6.0

****
Episode: 2280
Round: 35
Action: 16
Reward: 1.0
Total reward so far: 7.0

****
Episode: 2280
Round: 36
Action: 16
Reward: 1.0
Total reward so far: 8.0

****
Episode: 2280
Round: 37

Action: 16
Reward: -1.0
Total reward so far: 7.0

****
Episode: 2280
Round: 38
Action: 16
Reward: -1.0
Total reward so far: 6.0

****
Episode: 2280
Round: 39
Action: 16
Reward: -1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 40
Action: 16
Reward: 1.0
Total reward so far: 6.0

****
Episode: 2280
Round: 41
Action: 16
Reward: -1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 42
Action: 16
Reward: 1.0
Total reward so far: 6.0

****
Episode: 2280
Round: 43
Action: 16
Reward: -1.0

Total reward so far: 5.0

****
Episode: 2280
Round: 44
Action: 16
Reward: -1.0
Total reward so far: 4.0

****
Episode: 2280
Round: 45
Action: 16
Reward: 1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 46
Action: 16
Reward: -1.0
Total reward so far: 4.0

****
Episode: 2280
Round: 47
Action: 16
Reward: 1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 48
Action: 31
Reward: -1.0
Total reward so far: 4.0

****
Episode: 2280
Round: 49
Action: 25
Reward: 1.0
Total reward so far: 5.0

****
Episode: 2280
Round: 50
Action: 16
Reward: 1.0
Total reward so far: 6.0

****
Episode: 2280
Round: 51
Action: 16
Reward: 1.0
Total reward so far: 7.0

****
Episode: 2280
Round: 52
Action: 16
Reward: 1.0
Total reward so far: 8.0

****
Episode: 2280
Round: 53
Action: 16
Reward: -1.0
Total reward so far: 7.0

****
Episode: 2280
Round: 54
Action: 0
Reward: 36.0
Total reward so far: 43.0

****
Episode: 2280
Round: 55
Action: 0
Reward: -1.0
Total reward so far: 42.0

****
Episode: 2280

Round: 56
Action: 20
Reward: -1.0
Total reward so far: 41.0

****
Episode: 2280
Round: 57
Action: 0
Reward: -1.0
Total reward so far: 40.0

****
Episode: 2280
Round: 58
Action: 0
Reward: -1.0
Total reward so far: 39.0

****
Episode: 2280
Round: 59
Action: 0
Reward: -1.0
Total reward so far: 38.0

****
Episode: 2280
Round: 60
Action: 0
Reward: -1.0
Total reward so far: 37.0

****
Episode: 2280
Round: 61
Action: 0
Reward: -1.0
Total reward so far: 36.0

****
Episode: 2280
Round: 62
Action: 0

Reward: -1.0
Total reward so far: 35.0

****
Episode: 2280
Round: 63
Action: 0
Reward: -1.0
Total reward so far: 34.0

****
Episode: 2280
Round: 64
Action: 0
Reward: -1.0
Total reward so far: 33.0

****
Episode: 2280
Round: 65
Action: 0
Reward: -1.0
Total reward so far: 32.0

****
Episode: 2280
Round: 66
Action: 0
Reward: -1.0
Total reward so far: 31.0

****
Episode: 2280
Round: 67
Action: 0
Reward: -1.0
Total reward so far: 30.0

****
Episode: 2280
Round: 68
Action: 0
Reward: -1.0
Total reward so far: 29.0

****
Episode: 2280
Round: 69
Action: 0
Reward: -1.0
Total reward so far: 28.0

****
Episode: 2280
Round: 70
Action: 0
Reward: -1.0
Total reward so far: 27.0

****
Episode: 2280
Round: 71
Action: 0
Reward: -1.0
Total reward so far: 26.0

****
Episode: 2280
Round: 72
Action: 0
Reward: -1.0
Total reward so far: 25.0

****
Episode: 2280
Round: 73
Action: 0
Reward: 36.0
Total reward so far: 61.0

****
Episode: 2280
Round: 74
Action: 0
Reward: -1.0
Total reward so far: 60.0

****

Episode: 2280
Round: 75
Action: 0
Reward: -1.0
Total reward so far: 59.0

****

Episode: 2280
Round: 76
Action: 0
Reward: -1.0
Total reward so far: 58.0

****

Episode: 2280
Round: 77
Action: 0
Reward: -1.0
Total reward so far: 57.0

****

Episode: 2280
Round: 78
Action: 0
Reward: -1.0
Total reward so far: 56.0

****

Episode: 2280
Round: 79
Action: 0
Reward: 36.0
Total reward so far: 92.0

****

Episode: 2280
Round: 80
Action: 0
Reward: -1.0
Total reward so far: 91.0

****

Episode: 2280
Round: 81

Action: 0
Reward: -1.0
Total reward so far: 90.0

****
Episode: 2280
Round: 82
Action: 0
Reward: -1.0
Total reward so far: 89.0

****
Episode: 2280
Round: 83
Action: 0
Reward: -1.0
Total reward so far: 88.0

****
Episode: 2280
Round: 84
Action: 0
Reward: -1.0
Total reward so far: 87.0

****
Episode: 2280
Round: 85
Action: 0
Reward: -1.0
Total reward so far: 86.0

****
Episode: 2280
Round: 86
Action: 0
Reward: -1.0
Total reward so far: 85.0

****
Episode: 2280
Round: 87
Action: 0
Reward: -1.0

Total reward so far: 84.0

****
Episode: 2280
Round: 88
Action: 5
Reward: 1.0
Total reward so far: 85.0

****
Episode: 2280
Round: 89
Action: 0
Reward: -1.0
Total reward so far: 84.0

****
Episode: 2280
Round: 90
Action: 0
Reward: -1.0
Total reward so far: 83.0

****
Episode: 2280
Round: 91
Action: 0
Reward: -1.0
Total reward so far: 82.0

****
Episode: 2280
Round: 92
Action: 0
Reward: -1.0
Total reward so far: 81.0

****
Episode: 2280
Round: 93
Action: 0
Reward: -1.0
Total reward so far: 80.0

****
Episode: 2280
Round: 94
Action: 0
Reward: -1.0
Total reward so far: 79.0

****
Episode: 2280
Round: 95
Action: 0
Reward: -1.0
Total reward so far: 78.0

****
Episode: 2280
Round: 96
Action: 0
Reward: -1.0
Total reward so far: 77.0

****
Episode: 2280
Round: 97
Action: 25
Reward: -1.0
Total reward so far: 76.0

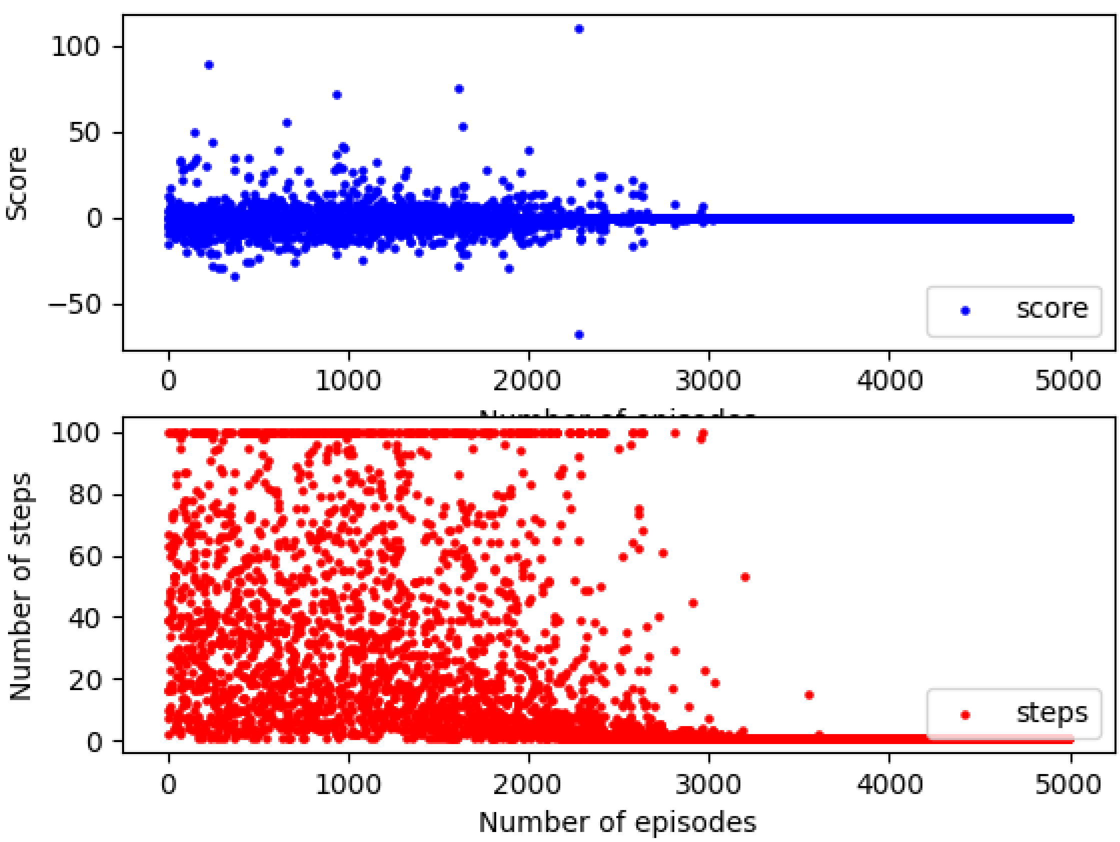
****
Episode: 2280
Round: 98
Action: 0
Reward: 36.0
Total reward so far: 112.0

****
Episode: 2280
Round: 99
Action: 0
Reward: -1.0
Total reward so far: 111.0

****
Episode: 2280

Round: 100
Action: 0
Reward: -1.0
Total reward so far: 110.0

Q-Table after 5000 episodes
[[-3.07583358e-02 -4.19334032e-02 -5.99201427e-02 -3.76336086e-02
  -9.83211895e-02 -1.35208913e-02 -1.48916007e-02 -7.34094117e-02
  -2.78890312e-02 -2.38599619e-02 -8.43982923e-02 -2.09545907e-02
  -3.38959585e-02 -7.01535786e-02 -5.08198245e-02 -8.84553123e-03
  -7.96039430e-02 -7.43652260e-02 -2.86639392e-02 -6.85138461e-02
  -7.48503926e-02 -3.73022417e-02 -1.20240760e-02 -2.79969394e-02
  -1.20484291e-02 -6.39220388e-02 -4.02999360e-02 -3.19100504e-03
  -6.82564864e-02 -7.04195457e-02 -2.58737313e-04 -1.96667366e-02
  -4.47963458e-02 -4.82113761e-02 -6.84705007e-02 -3.36668727e-02
  -2.11633852e-02 1.57973762e-06]]
The best score after running 5000 episodes: 110.0



Code:-

```
import gym
import gym_toytext
import numpy as np
from IPython.display import clear_output
from time import sleep
import matplotlib.pyplot as plt

env = gym.make('Roulette-v0')

action_space_size = env.action_space.n
state_space_size = env.observation_space.n
print(f"Number of actions available: {action_space_size}")
print(f"Number of states defined: {state_space_size}")
print(f"Therefore, Q-Table with {action_space_size} columns and {state_space_size} rows will be created")

q_table = np.random.random([state_space_size, action_space_size])
#OR
# q_table = np.zeros([state_space_size, action_space_size])
print(f"Q-Table shape: {q_table.shape}")

sleep(5)

# Hyperparameters
TOTAL_EPISODES = 5_000 #Number of epsiodes to train the algorithm
MAX_STEPS = 150 #Max steps an agent can take during an episode

LEARNING_RATE = 0.1
GAMMA = 0.95 # Discount (close to 0 makes it greedy, close to 1 considers long term)

# Exploration Parameters
epsilon = 1
START_EPSILON_DECAYING = 1
END_EPSILON_DECAYING = TOTAL_EPISODES // 2
DECAY_RATE = epsilon/(END_EPSILON_DECAYING - START_EPSILON_DECAYING)

def print_frames(frames):
    total_reward = 0
    for i, frame in enumerate(frames):
    clear_output(wait=True)
    print('\n*********************')
    print(f"Episode: {frame['episode']}")
    print(f'Round: {i + 1}')
    print(f"Action: {frame['action']}")
```

```
        print(f"Reward: {frame['reward']}")
        total_reward += frame['reward']
        print(f"Total reward so far: {total_reward}")
        sleep(0.1)


def edit_reward(info):
        print(info)

env.reset()
history = {'steps': [], 'total_score': [], 'episode_number': []}
best_frames = []
high_score = 0

for episode in range(TOTAL_EPISODES):

        state = env.reset()
        step = 0
        frames = []
        current_score = 0
        done = False

        for step in range(MAX_STEPS):

        if np.random.random() > epsilon: #This is exploitation
        action = np.argmax(q_table[state, :]) #Current state, max value
        else: #This is exploration
        action = np.random.randint(0, action_space_size)

        new_state, reward, done, info = env.step(action)

        current_score += reward

        frames.append({
        'episode': episode,
        'action': action,
        'reward': reward
        })

        q_table[state, action] = (1 - LEARNING_RATE) * q_table[state, action] + LEARNING_RATE * (reward
+ GAMMA * np.max(q_table[new_state, :]))

        if done:
```

```
        history['steps'].append(step + 1)
        history['total_score'].append(current_score)
        history['episode_number'].append(episode)

        if current_score >= high_score:
            high_score = current_score
            best_frames = frames.copy()
        break

        state = new_state

        if END_EPSILON_DECAYING >= episode >= START_EPSILON_DECAYING:
        epsilon -= DECAY_RATE

env.close()
print_frames(best_frames)
print(f"\nQ-Table after {TOTAL_EPISODES} episodes")
print(q_table)
print(f'The best score after running {TOTAL_EPISODES} episodes: {high_score}')

# Total score
plt.subplot(2, 1, 1)
plt.scatter(history['episode_number'], history['total_score'], s = 5, label = 'score', color = 'blue')
plt.xlabel('Number of episodes')
plt.ylabel('Score')
plt.legend(loc = 4)

# Number of steps
plt.subplot(2, 1, 2)
plt.scatter(history['episode_number'], history['steps'], s = 5, label = 'steps', color = 'red')
plt.xlabel('Number of episodes')
plt.ylabel('Number of steps')
plt.legend(loc = 4)
plt.show()
```