Maze: Q-Learning

Install Dependencies:

```
!pip install gym
!pip install pygame
```

In [106…

```
# Importing Libraries and dependencies
import gym
import MazeEnv
import random
import time
import numpy as np
```

In [107…

```
# Initializing Gym-env
env=gym.make('maze-v0')
```

In [108…

```
# Initializing hyperparameters
alpha = 0.5
gamma = 0.75
reward_final=0
random_state = np.random.RandomState(100)
eps = 0.06
curr_state=0
env.reset()
flag=False
```

In [109…

```
# Initializing Q-Table
q_table = np.zeros((6, 6))
```

In [110…

```
# Initializing the rewards matrix
reward_table = np.array([[-1, -1, -1, -1, 0, -1],
                         [-1, -1, -1, 0, -1, 100],
                         [-1, -1, -1, 0, -1, -1],
                         [-1, 0, 0, -1, 0, -1],
                         [0, -1, -1, 0, -1, 100],
                         [-1, 0, -1, -1, 0, 100]])
```

In [111…

```
print("Initial Q-Table")
print(q_table)

print("Rewards Matrix")
print(reward_table)
```

```
Initial Q-Table
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
Rewards Matrix
[[ -1  -1  -1  -1   0  -1]
```

```
[ -1  -1  -1   0  -1 100]
[ -1  -1  -1   0  -1  -1]
[ -1   0   0  -1   0  -1]
[  0  -1  -1   0  -1 100]
[ -1   0  -1  -1   0 100]]
```

In [112…

```python
for i in range(100):
    states = list(range(6))
    random_state.shuffle(states)
    if flag == False:
            #action = np.argmax(q_table[state])
            #env.render()
            legal = reward_table[env.state] >= 0
            actions = np.array(list(range(6)))
            legal_actions = actions[legal == True]
            print('In', env.state)
            print("Next possible states:", legal_actions)
            if random_state.rand() < eps:
                action = int(legal_actions[0])
            else:
                if np.sum(q_table[env.state]) > 0:
                    action = np.argmax(q_table[env.state])
                else:
                    action = env.actions[int(random.random() * len(env.actions))]
            next_state, r, is_terminal, info = env.step(action)
            if r >0:
                reward_final += r
            print("In " + str(env.state), ", Next Action:" + str(action) + " to " + str(ne
            print("Reward:", reward_final)

            reward = reward_table[env.state, next_state]
            compute = reward + gamma * max(q_table[next_state, :])
            q_table[env.state, next_state] = compute

            if is_terminal == True:
                curr_state=next_state
                flag=True
```

```
In 2
Next possible states: [3]
In 3 , Next Action:3 to 3
Reward: 0
In 3
Next possible states: [1 2 4]
In 3 , Next Action:0 to 3
Reward: 0
In 3
Next possible states: [1 2 4]
In 2 , Next Action:2 to 2
Reward: 0
In 2
Next possible states: [3]
In 3 , Next Action:3 to 3
Reward: 0
In 3
Next possible states: [1 2 4]
In 3 , Next Action:3 to 3
Reward: 0
In 3
Next possible states: [1 2 4]
In 3 , Next Action:3 to 3
Reward: 0
In 3
Next possible states: [1 2 4]
```

```
In 3 , Next Action:5 to 3
Reward: 0
In 3
Next possible states: [1 2 4]
In 2 , Next Action:2 to 2
Reward: 0
In 2
Next possible states: [3]
In 2 , Next Action:1 to 2
Reward: 0
In 2
Next possible states: [3]
In 2 , Next Action:1 to 2
Reward: 0
In 2
Next possible states: [3]
In 2 , Next Action:0 to 2
Reward: 0
In 2
Next possible states: [3]
In 3 , Next Action:3 to 3
Reward: 0
In 3
Next possible states: [1 2 4]
In 1 , Next Action:1 to 1
Reward: 0
In 1
Next possible states: [3 5]
In 1 , Next Action:1 to 1
Reward: 0
In 1
Next possible states: [3 5]
In 1 , Next Action:1 to 1
Reward: 0
In 1
Next possible states: [3 5]
In 3 , Next Action:3 to 3
Reward: 0
In 3
Next possible states: [1 2 4]
In 1 , Next Action:1 to 1
Reward: 0
In 1
Next possible states: [3 5]
In 1 , Next Action:4 to 1
Reward: 0
In 1
Next possible states: [3 5]
In 1 , Next Action:0 to 1
Reward: 0
In 1
Next possible states: [3 5]
In 5 , Next Action:5 to 5
Reward: 100
```

In [113…

```
print('Final state:',curr_state)
print()
print("Final reward:", reward_final)
env.close()
```

```
Final state: 5

Final reward: 100
```