Step 1: Import statements

In [ ]:

```
!pip install numpy
!pip install openai-gym
```

In [ ]:

```
import numpy as np
import gym
import random
```

In [ ]:

```
env = gym.make("FrozenLake-v0")
```

Step 2: Create the Q-table and initialize it

In [ ]:

```
action_size = env.action_space.n
state_size = env.observation_space.n

print(f"Action Space : {action_size} | State Space: {state_size}")
```

```
Action Space : 4 | State Space: 16
```

In [ ]:

```
qtable = np.zeros((state_size, action_size))
print(qtable.shape)
```

```
(16, 4)
```

Step 3: Create Required Hyperparameters

In [ ]:

```
total_episodes = 15000        # Total episodes
learning_rate = 0.8           # Learning rate
max_steps = 99                # Max steps per episode
gamma = 0.95                  # Discounting rate

# Exploration parameters
epsilon = 1.0                 # Exploration rate
max_epsilon = 1.0             # Exploration probability at start
min_epsilon = 0.01            # Minimum exploration probability
decay_rate = 0.005             # Exponential decay rate for exploration prob
```

Step 4 : Q-Learning Algorithm

In [ ]:

```
rewards = []

for episode in range(total_episodes):

    state = env.reset()
    step = 0
    done = False
```

```
    total_rewards = 0

    for step in range(max_steps):
        exp_exp_tradeoff = random.uniform(0, 1)

        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state,:])

        else:
            action = env.action_space.sample()

        new_state, reward, done, info = env.step(action)
        qtable[state, action] = qtable[state, action] + learning_rate * \
                                                        (reward + gamma * np.max(qtable
[new_state, :]) - qtable[state, action])

        total_rewards += reward
        state = new_state

        if done == True:
            break

    epsilon = min_epsilon + (max_epsilon - min_epsilon) * np.exp(-decay_rate * episode)
    rewards.append(total_rewards)

print ("Score over time: " +  str(sum(rewards)/total_episodes))
print(qtable)
```

```
Score over time: 0.49193333333333333
[[2.85642128e-01 5.63009679e-02 6.00197517e-02 5.84297141e-02]
 [1.24327386e-02 1.03987053e-03 2.07422474e-02 5.57637665e-02]
 [3.51826419e-03 3.34560407e-02 1.65847652e-02 4.14667415e-02]
 [7.82752313e-03 3.14799990e-03 1.90844369e-02 3.33905056e-02]
 [3.51875459e-01 5.04643642e-03 3.53273061e-02 2.18123407e-02]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [7.84896909e-05 6.60730328e-12 5.77437870e-03 2.30256352e-05]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [3.64913242e-02 4.54388862e-02 7.21510945e-02 1.42910860e-01]
 [1.98384110e-02 7.74155977e-02 1.35533203e-01 7.54856386e-02]
 [6.32227464e-02 7.07958860e-03 5.96900730e-04 2.70980272e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [8.83203784e-02 9.52954975e-02 6.88819326e-01 6.38565822e-02]
 [2.53616483e-01 9.91934449e-01 2.25794842e-01 2.47528926e-01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]
```

In [ ]:

```
for episode in range(5):
    state = env.reset()
    step = 0
    done = False
    print("****************************************************")
    print("EPISODE ", episode)

    for step in range(max_steps):
        action = np.argmax(qtable[state,:])
        new_state, reward, done, info = env.step(action)

        if done:
            env.render()
            print("Number of steps", step)
            break
        state = new_state
env.close()
```

```
****************************************************
EPISODE  0
  (Right)
SFFF
FHFH
FFFH
```

```
HFFG
Number of steps 54
**************************************************
EPISODE  1
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 30
**************************************************
EPISODE  2
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 11
**************************************************
EPISODE  3
  (Right)
SFFF
FHFH
FFFH
HFFG
Number of steps 4
**************************************************
EPISODE  4
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 65
```

2nd Trial

In [ ]:

```
rewards = []

for episode in range(10000):

    state = env.reset()
    step = 0
    done = False
    total_rewards = 0

    for step in range(max_steps):
        exp_exp_tradeoff = random.uniform(0, 1)

        if exp_exp_tradeoff > epsilon:
            action = np.argmax(qtable[state,:])

        else:
            action = env.action_space.sample()

        new_state, reward, done, info = env.step(action)
        qtable[state, action] = qtable[state, action] + learning_rate * \
                                                        (reward + gamma * np.max(qtable
[new_state, :]) - qtable[state, action])

        total_rewards += reward
        state = new_state

        if done == True:
            break

    epsilon = min_epsilon + (max_epsilon - min_epsilon) * np.exp(-decay_rate * episode)
    rewards.append(total_rewards)
```

```
print ("Score over time: " +  str(sum(rewards)/total_episodes))
print(qtable)

for episode in range(5):
    state = env.reset()
    step = 0
    done = False
    print("****************************************************")
    print("EPISODE ", episode)

    for step in range(max_steps):
        action = np.argmax(qtable[state,:])
        new_state, reward, done, info = env.step(action)

        if done:
            env.render()
            print("Number of steps", step)
            break
        state = new_state
env.close()
```

```
Score over time: 0.30406666666666665
[[1.20150016e-01 1.10296732e-01 4.18205756e-02 6.10854298e-02]
 [2.75803855e-03 8.75455604e-03 5.58200254e-03 5.96353838e-02]
 [2.32560777e-03 1.13637221e-02 6.42395352e-03 7.12626586e-02]
 [1.41572416e-03 6.48800942e-03 8.72054751e-04 4.96731323e-02]
 [9.64980219e-02 1.03890963e-02 1.98520496e-03 8.98424363e-02]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [2.86843030e-06 1.86694783e-03 8.69398769e-04 3.31246023e-08]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.22229924e-02 2.04584012e-02 2.54181614e-02 2.03572193e-01]
 [1.32770729e-02 1.86467398e-01 1.83254711e-02 1.14909216e-04]
 [4.18742309e-02 1.46471006e-02 1.56734790e-03 8.96309753e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [7.20441101e-02 3.74535420e-03 6.81300759e-01 9.35064916e-02]
 [8.35899114e-02 9.41974616e-01 2.17626046e-01 1.65590975e-01]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]
**************************************************
EPISODE  0
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 70
**************************************************
EPISODE  1
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 48
**************************************************
EPISODE  2
**************************************************
EPISODE  3
  (Down)
SFFF
FHFH
FFFH
HFFG
Number of steps 13
**************************************************
EPISODE  4
  (Down)
SFFF
FHFH
FFFH
HFFG
```

```
Number of steps 15
```

In [ ]: