```
from frozen_lake import FrozenLakeEnv
#from frozen_lake_slippery import FrozenLakeEnv
import numpy as np
import random

env = FrozenLakeEnv()

Q = np.zeros([env.nS,env.nA])



def epsilon_greedy_action(env,Q,state,epsilon=0.3):
    n = random.uniform(0,1)
    if n<= epsilon:
        return np.random.randint(env.action_space.n)
    else:
        return np.argmax(Q[state])


def Q_learing(env, episodes=1000, gamma=0.9, alpha=0.3):

    Q = np.zeros([env.nS,env.nA])
    for i in range(episodes):

        env.reset()
        finished = False

        S = env.s


        while not finished:

            A = epsilon_greedy_action(env,Q,S)
            next_S, R, finished, _ = env.step(A)


            Q[S][A] = Q[S][A] + alpha * (R + gamma * Q[next_S, np.argmax(Q[next_S])] - Q[S

            S = next_S

    return Q


Q = Q_learing(env,2000)
print(np.round(Q,2))
```

```
[[0.53 0.59 0.48 0.53]
 [0.53 0.   0.39 0.48]
 [0.48 0.   0.   0.1 ]
 [0.   0.   0.   0.  ]
 [0.59 0.66 0.   0.53]
 [0.   0.   0.   0.  ]
 [0.   0.81 0.   0.33]
 [0.   0.   0.   0.  ]
 [0.66 0.   0.73 0.59]
 [0.66 0.81 0.81 0.  ]
 [0.73 0.9  0.   0.73]
 [0.   0.   0.   0.  ]
 [0.   0.   0.   0.  ]
 [0.   0.78 0.9  0.73]
 [0.81 0.9  1.   0.81]
 [0.   0.   0.   0.  ]]
```