Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Лабораторная работа №5
по дисциплине «Методы машинного обучения»
«Обучение на основе временных различий»

ИСПОЛНИТЕЛЬ:

Алехин С.С.
Группа ИУ5-23М

_____

ПРОВЕРИЛ:

Балашов А.М.

_____

Москва, 2023

Задание

На основе рассмотренного на лекции примера реализуйте следующие алгоритмы:

- SARSA
- Q-обучение
- Двойное Q-обучение

Для любой среды обучения с подкреплением (кроме рассмотренной на лекции среды Toy Text / Frozen Lake) из библиотеки Gym (или аналогичной библиотеки).

Lab5

June 21, 2023

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import gymnasium as gym
from tqdm import tqdm
```

0.0.1

[ ]:
```
class BasicAgent:
    '''
          ,
    '''

    #
    ALGO_NAME = '---'

    def __init__(self, env, eps=0.1):
        #
        self.env = env
        #        Q-
        self.nA = env.action_space.n
        self.nS = env.observation_space.n
        #
        self.Q = np.zeros((self.nS, self.nA))
        #
        #
        self.eps=eps
        #
        self.episodes_reward = []

    def print_q(self):
        print('    Q-             ', self.ALGO_NAME)
        print(self.Q)

    def get_state(self, state):
        '''

        '''
        if type(state) is tuple:
```

```
            #                          ,
            return state[0]
        else:
            return state

    def greedy(self, state):
        '''
        <<   >>
                  ,                    Q-
              state
        '''
        return np.argmax(self.Q[state])

    def make_action(self, state):
        '''

        '''
        if np.random.uniform(0,1) < self.eps:

            #                eps
            #
            return self.env.action_space.sample()
        else:
            #          ,                    Q-
            return self.greedy(state)

    def draw_episodes_reward(self):
        #
        fig, ax = plt.subplots(figsize = (15,10))
        y = self.episodes_reward
        x = list(range(1, len(y)+1))
        plt.plot(x, y, '-', linewidth=1, color='green')
        plt.title('            ')
        plt.xlabel('        ')
        plt.ylabel('    ')
        plt.show()

    def learn():
        '''

        '''
        pass
```

0.0.2 SARSA

[ ]:
```
class SARSA_Agent(BasicAgent):
    '''
                SARSA
    '''
    #
    ALGO_NAME = 'SARSA'

    def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000):
        #
        super().__init__(env, eps)
        # Learning rate
        self.lr=lr
        #
        self.gamma = gamma
        #
        self.num_episodes=num_episodes
        #               eps
        self.eps_decay=0.00005
        self.eps_threshold=0.01

    def learn(self):
        '''
                        SARSA
        '''
        self.episodes_reward = []
        #
        for ep in tqdm(list(range(self.num_episodes))):
            #
            state = self.get_state(self.env.reset())
            #
            done = False
            #
            truncated = False
            #
            tot_rew = 0

            #             Q-
            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay

            #
            action = self.make_action(state)

            #
            while not (done or truncated):
                #
```

```
                next_state, rew, done, truncated, _ = self.env.step(action)

                #
                next_action = self.make_action(next_state)

                #            Q    SARSA
                self.Q[state][action] = self.Q[state][action] + self.lr * \
                    (rew + self.gamma * self.Q[next_state][next_action] - self.
↪Q[state][action])

                #
                state = next_state
                action = next_action
                #
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)
```

0.0.3 Q-

[ ]:
```
class QLearning_Agent(BasicAgent):
    '''
                Q-Learning
    '''
    #
    ALGO_NAME = 'Q-     '

    def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000):
        #
        super().__init__(env, eps)
        # Learning rate
        self.lr=lr
        #
        self.gamma = gamma
        #
        self.num_episodes=num_episodes
        #               eps
        self.eps_decay=0.00005
        self.eps_threshold=0.01

    def learn(self):
        '''
                        Q-Learning
        '''
        self.episodes_reward = []
        #
        for ep in tqdm(list(range(self.num_episodes))):
```

```
            #
            state = self.get_state(self.env.reset())
            #
            done = False
            #
            truncated = False
            #
            tot_rew = 0

            #             Q-
            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay

            #
            while not (done or truncated):
                #
                #   SARSA
                action = self.make_action(state)

                #
                next_state, rew, done, truncated, _ = self.env.step(action)

                #             Q    SARSA (        )
                # self.Q[state][action] = self.Q[state][action] + self.lr * \
                #     (rew + self.gamma * self.Q[next_state][next_action] -␣
↪self.Q[state][action])

                #               Q-
                self.Q[state][action] = self.Q[state][action] + self.lr * \
                    (rew + self.gamma * np.max(self.Q[next_state]) - self.
↪Q[state][action])

                #
                state = next_state
                #
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)
```

0.0.4 Q-

[ ]:
```
class DoubleQLearning_Agent(BasicAgent):
    '''
                Double Q-Learning
    '''
    #
    ALGO_NAME = '     Q-     '
```

```
def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000):
    #
    super().__init__(env, eps)
    #
    self.Q2 = np.zeros((self.nS, self.nA))
    # Learning rate
    self.lr=lr
    #
    self.gamma = gamma
    #
    self.num_episodes=num_episodes
    #               eps
    self.eps_decay=0.00005
    self.eps_threshold=0.01


def greedy(self, state):
    '''
    <<   >>
              ,                  Q-
          state
    '''
    temp_q = self.Q[state] + self.Q2[state]
    return np.argmax(temp_q)

def print_q(self):
    print(f"    Q-             {self.ALGO_NAME}")
    print('Q1')
    print(self.Q)
    print('Q2')
    print(self.Q2)

def learn(self):
    '''
                    Double Q-Learning
    '''
    self.episodes_reward = []
    #
    for ep in tqdm(list(range(self.num_episodes))):
        #
        state = self.get_state(self.env.reset())
        #
        done = False
        #
        truncated = False
        #
```

```
            tot_rew = 0

            #             Q-
            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay

            #
            while not (done or truncated):
                #
                #   SARSA
                action = self.make_action(state)

                #
                next_state, rew, done, truncated, _ = self.env.step(action)

                if np.random.rand() < 0.5:
                    #
                    self.Q[state][action] = self.Q[state][action] + self.lr * \
                        (rew + self.gamma * self.Q2[next_state][np.argmax(self.
↪Q[next_state])] - self.Q[state][action])
                else:
                    #
                    self.Q2[state][action] = self.Q2[state][action] + self.lr *␣
↪\
                        (rew + self.gamma * self.Q[next_state][np.argmax(self.
↪Q2[next_state])] - self.Q2[state][action])

                #
                state = next_state
                #
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)
```

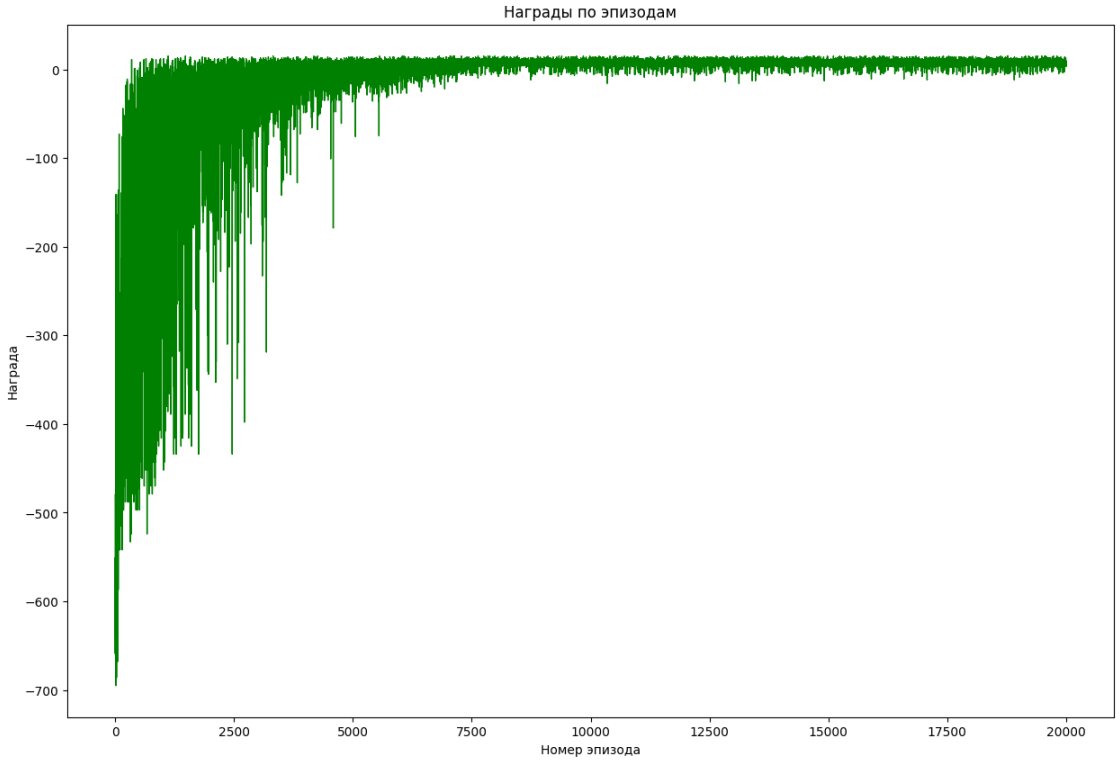
[ ]:
```
def play_agent(agent):
    '''

    '''
    env2 = gym.make('Taxi-v3', render_mode='human')
    state = env2.reset()[0]
    done = False
    while not done:
        action = agent.greedy(state)
        next_state, reward, terminated, truncated, _ = env2.step(action)
        env2.render()
        state = next_state
        if terminated or truncated:
```

```
            done = True

def run_sarsa():
    env = gym.make('Taxi-v3')
    agent = SARSA_Agent(env)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)

def run_q_learning():
    env = gym.make('Taxi-v3')
    agent = QLearning_Agent(env)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)

def run_double_q_learning():
    env = gym.make('Taxi-v3')
    agent = DoubleQLearning_Agent(env)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)
```

0.0.5 SARSA: eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000

[ ]:
```
run_sarsa()
```

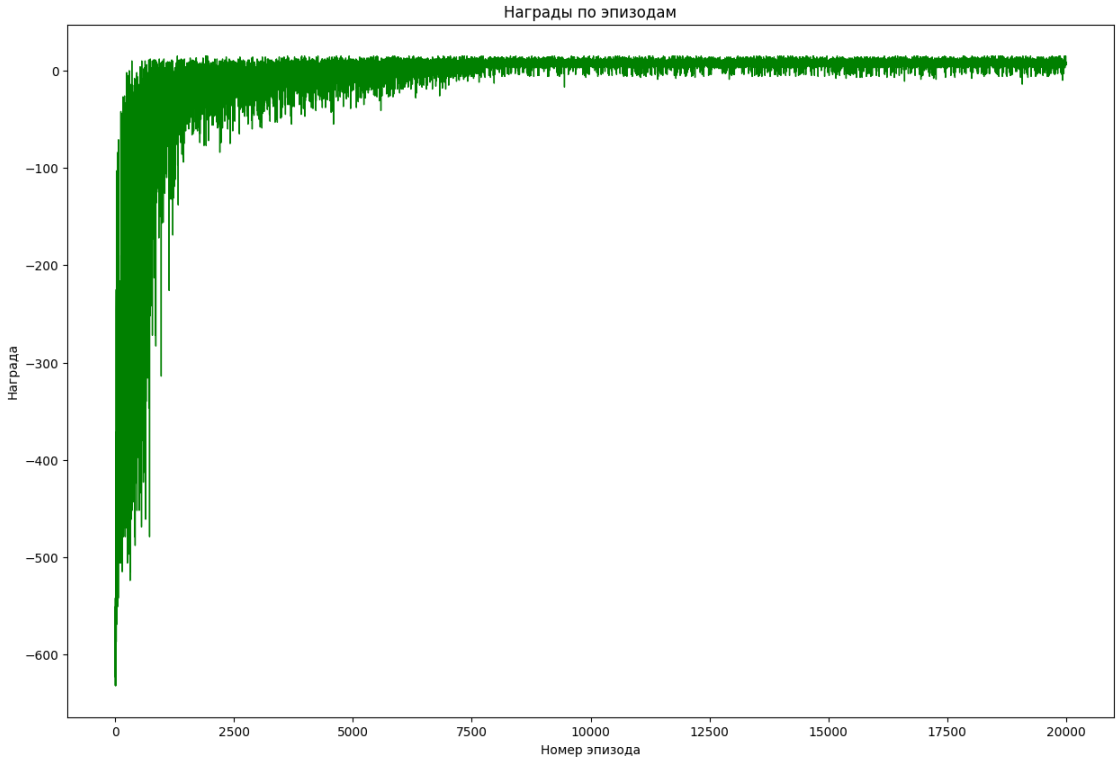
```
  0%|          | 0/20000 [00:00<?, ?it/s]100%|      | 20000/20000
[00:03<00:00, 6075.86it/s]

    Q-              SARSA
[[  0.           0.           0.           0.           0.
    0.        ]
 [ -9.39511871  -7.82633886  -4.61973996  -7.91608244   7.52831391
  -14.58270727]
 [  0.72064374   2.33945919   2.24226059   1.10286948  13.0813314
   -6.7719644 ]
 …
 [ -3.08864603   5.58824812  -3.13475835  -2.35462461  -7.02756406
   -9.25639657]
 [ -7.77032983  -8.05657849  -8.49347554  -1.95002266 -14.61611739
  -14.22935898]
 [  5.45984704   4.18158543  10.07790051  18.007744     1.41430917
    1.34707593]]
```



0.0.6 Q- : eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000

[ ]:
```
run_q_learning()
```

```
100%|      | 20000/20000 [00:03<00:00, 5488.88it/s]

    Q-              Q-
[[ 0.          0.          0.          0.          0.          0.        ]
 [ 5.55943204  6.43481406  4.66190971  6.25449521  8.36234335 -2.09352707]
 [ 9.16004638 11.34118969  9.82937408 10.403992   13.27445578  2.29056275]
 …
 [ 1.48773717 13.44572688  1.69519677  0.142458   -3.59137074 -5.56482255]
 [-1.95348732  8.73563506 -2.38814501  0.59149548 -8.87902035 -7.38586058]
 [ 0.          6.46636365  9.54364847 18.59693097  0.87934239  3.81148721]]
```

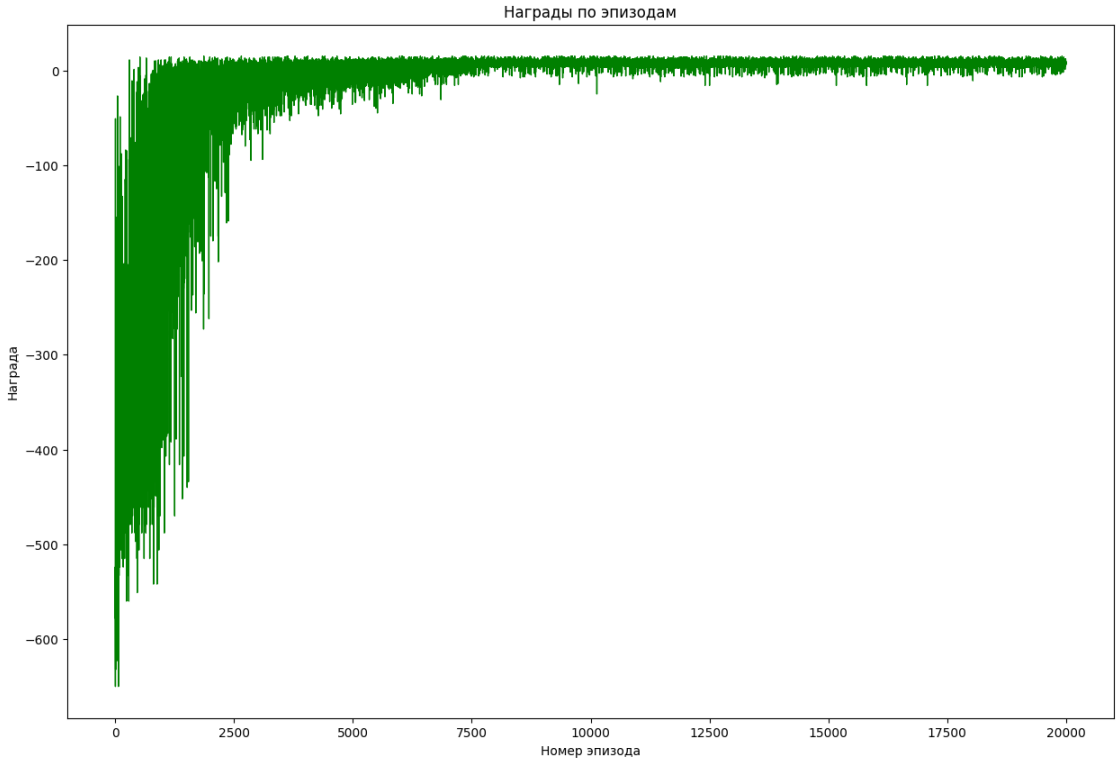


0.0.7 Q- : eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000

[ ]:
```
run_double_q_learning()
```

```
100%|      | 20000/20000 [00:04<00:00, 4883.75it/s]

    Q-                  Q-
Q1
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 1.16163046e+00  6.45016960e-01  1.00769948e+00  3.72291477e+00
   8.36234335e+00 -5.01468586e+00]
 [ 6.09505909e+00  8.36560187e+00 -4.07224966e-01  7.05822996e+00
   1.32744558e+01  6.45841012e-01]
 …
 [-7.99180545e-01  1.07550337e+01 -1.44867522e+00 -2.35590222e+00
  -5.86881524e+00 -5.62712420e+00]
 [-3.98247635e+00  4.37814489e+00 -3.92624883e+00 -2.60475579e+00
  -5.10770158e+00 -7.54219253e+00]
 [ 2.78683960e+00  2.85534245e+00  0.00000000e+00  1.83727537e+01
   4.18884557e-01 -7.50927475e-03]]
Q2
[[ 0.          0.          0.          0.          0.          0.        ]
 [-0.2934294   3.26994885 -1.42217566  5.04477693  8.36234335 -5.90593791]
```

```
[ 5.90112017  5.46169946  2.42918746  3.80673123 13.27445578 -0.5747265 ]
…
[-2.10538343 11.9391959  -1.57538495 -1.53177373 -3.18288299 -4.73475405]
[-3.74525344  3.01047271 -3.83516418 -4.04781581 -4.80672105 -5.4513151 ]
[ 3.14380145  1.05446787  1.51034447 18.44538414 -3.68630097  0.42193263]]
```



Lab5

June 22, 2023

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import gymnasium as gym
from tqdm import tqdm
```

0.0.1

[ ]:
```
class BasicAgent:
    '''
          ,
    '''

    #
    ALGO_NAME = '---'

    def __init__(self, env, eps=0.1):
        #
        self.env = env
        #        Q-
        self.nA = env.action_space.n
        self.nS = env.observation_space.n
        #
        self.Q = np.zeros((self.nS, self.nA))
        #
        #
        self.eps=eps
        #
        self.episodes_reward = []

    def print_q(self):
        print('    Q-             ', self.ALGO_NAME)
        print(self.Q)

    def get_state(self, state):
        '''

        '''
        if type(state) is tuple:
```

```
            #                          ,
            return state[0]
        else:
            return state

    def greedy(self, state):
        '''
        <<   >>
                  ,                    Q-
              state
        '''
        return np.argmax(self.Q[state])

    def make_action(self, state):
        '''

        '''
        if np.random.uniform(0,1) < self.eps:

            #                eps
            #
            return self.env.action_space.sample()
        else:
            #          ,                    Q-
            return self.greedy(state)

    def draw_episodes_reward(self):
        #
        fig, ax = plt.subplots(figsize = (15,10))
        y = self.episodes_reward
        x = list(range(1, len(y)+1))
        plt.plot(x, y, '-', linewidth=1, color='green')
        plt.title('            ')
        plt.xlabel('        ')
        plt.ylabel('    ')
        plt.show()

    def learn():
        '''

        '''
        pass
```

0.0.2 SARSA

[ ]:
```
class SARSA_Agent(BasicAgent):
    '''
                SARSA
    '''
    #
    ALGO_NAME = 'SARSA'

    def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000):
        #
        super().__init__(env, eps)
        # Learning rate
        self.lr=lr
        #
        self.gamma = gamma
        #
        self.num_episodes=num_episodes
        #               eps
        self.eps_decay=0.00005
        self.eps_threshold=0.01

    def learn(self):
        '''
                        SARSA
        '''
        self.episodes_reward = []
        #
        for ep in tqdm(list(range(self.num_episodes))):
            #
            state = self.get_state(self.env.reset())
            #
            done = False
            #
            truncated = False
            #
            tot_rew = 0

            #             Q-
            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay

            #
            action = self.make_action(state)

            #
            while not (done or truncated):
                #
```

```
                next_state, rew, done, truncated, _ = self.env.step(action)

                #
                next_action = self.make_action(next_state)

                #            Q    SARSA
                self.Q[state][action] = self.Q[state][action] + self.lr * \
                    (rew + self.gamma * self.Q[next_state][next_action] - self.
↪Q[state][action])

                #
                state = next_state
                action = next_action
                #
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)
```

0.0.3 Q-

[ ]:
```
class QLearning_Agent(BasicAgent):
    '''
                Q-Learning
    '''
    #
    ALGO_NAME = 'Q-     '

    def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000):
        #
        super().__init__(env, eps)
        # Learning rate
        self.lr=lr
        #
        self.gamma = gamma
        #
        self.num_episodes=num_episodes
        #               eps
        self.eps_decay=0.00005
        self.eps_threshold=0.01

    def learn(self):
        '''
                        Q-Learning
        '''
        self.episodes_reward = []
        #
        for ep in tqdm(list(range(self.num_episodes))):
```

```
            #
            state = self.get_state(self.env.reset())
            #
            done = False
            #
            truncated = False
            #
            tot_rew = 0

            #             Q-
            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay

            #
            while not (done or truncated):
                #
                #   SARSA
                action = self.make_action(state)

                #
                next_state, rew, done, truncated, _ = self.env.step(action)

                #             Q    SARSA (        )
                # self.Q[state][action] = self.Q[state][action] + self.lr * \
                #     (rew + self.gamma * self.Q[next_state][next_action] -␣
↪self.Q[state][action])

                #               Q-
                self.Q[state][action] = self.Q[state][action] + self.lr * \
                    (rew + self.gamma * np.max(self.Q[next_state]) - self.
↪Q[state][action])

                #
                state = next_state
                #
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)
```

0.0.4 Q-

[ ]:
```
class DoubleQLearning_Agent(BasicAgent):
    '''
                Double Q-Learning
    '''
    #
    ALGO_NAME = '     Q-     '
```

```
def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000):
    #
    super().__init__(env, eps)
    #
    self.Q2 = np.zeros((self.nS, self.nA))
    # Learning rate
    self.lr=lr
    #
    self.gamma = gamma
    #
    self.num_episodes=num_episodes
    #               eps
    self.eps_decay=0.00005
    self.eps_threshold=0.01


def greedy(self, state):
    '''
    <<   >>
              ,                  Q-
          state
    '''
    temp_q = self.Q[state] + self.Q2[state]
    return np.argmax(temp_q)

def print_q(self):
    print(f"    Q-             {self.ALGO_NAME}")
    print('Q1')
    print(self.Q)
    print('Q2')
    print(self.Q2)

def learn(self):
    '''
                    Double Q-Learning
    '''
    self.episodes_reward = []
    #
    for ep in tqdm(list(range(self.num_episodes))):
        #
        state = self.get_state(self.env.reset())
        #
        done = False
        #
        truncated = False
        #
```

```
            tot_rew = 0

            #             Q-
            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay

            #
            while not (done or truncated):
                #
                #   SARSA
                action = self.make_action(state)

                #
                next_state, rew, done, truncated, _ = self.env.step(action)

                if np.random.rand() < 0.5:
                    #
                    self.Q[state][action] = self.Q[state][action] + self.lr * \
                        (rew + self.gamma * self.Q2[next_state][np.argmax(self.
↪Q[next_state])] - self.Q[state][action])
                else:
                    #
                    self.Q2[state][action] = self.Q2[state][action] + self.lr *␣
↪\
                        (rew + self.gamma * self.Q[next_state][np.argmax(self.
↪Q2[next_state])] - self.Q2[state][action])

                #
                state = next_state
                #
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)
```

[ ]:
```
def play_agent(agent):
    '''

    '''
    env2 = gym.make('CliffWalking-v0', render_mode='human')
    state = env2.reset()[0]
    done = False
    while not done:
        action = agent.greedy(state)
        next_state, reward, terminated, truncated, _ = env2.step(action)
        env2.render()
        state = next_state
        if terminated or truncated:
```

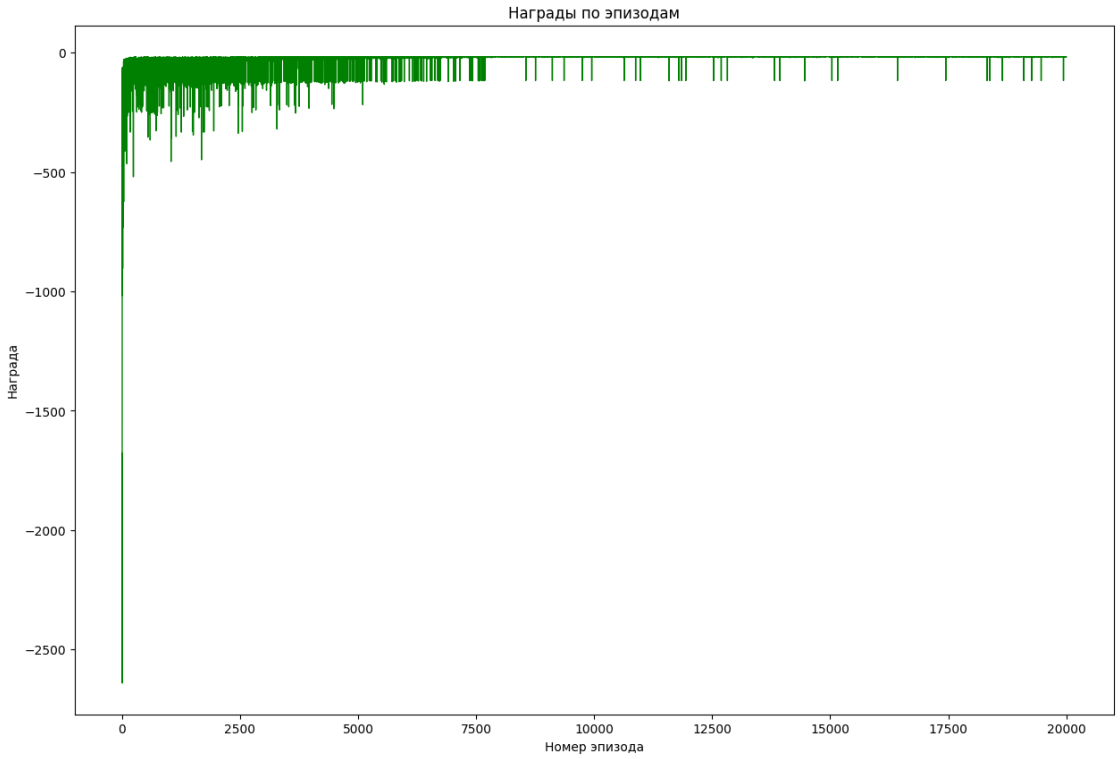
```
            done = True

def run_sarsa():
    env = gym.make('CliffWalking-v0')
    agent = SARSA_Agent(env)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)

def run_q_learning():
    env = gym.make('CliffWalking-v0')
    agent = QLearning_Agent(env)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)

def run_double_q_learning():
    env = gym.make('CliffWalking-v0')
    agent = DoubleQLearning_Agent(env)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)
```

0.0.5 SARSA: eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000

[ ]:
```
run_sarsa()
```

```
  0%|          | 0/20000 [00:00<?, ?it/s]100%|      | 20000/20000
[00:02<00:00, 7673.73it/s]

    Q-              SARSA
[[ -13.23268491  -12.44924206  -14.14831971  -13.21756225]
 [ -12.45494055  -11.67549393  -13.20950629  -13.4035355 ]
 [ -11.69529513  -10.84139566  -12.57271256  -12.64912424]
 [ -10.87218618   -9.9998437   -11.76705827  -11.90242225]
 [ -10.06010061   -9.1837413   -11.09200247  -11.06105557]
 [  -9.25122716   -8.34921633  -10.17854914  -10.23577852]
 [  -8.38617936   -7.4893919    -9.4211632    -9.53270723]
 [  -7.55922396   -6.60501981   -8.51359079   -8.58745887]
 [  -6.63714748   -5.70849155   -7.68342711   -7.75050899]
 [  -5.77551649   -4.80404568   -5.54444773   -6.83273798]
 [  -4.84843535   -3.88159414   -4.15313036   -5.99035878]
 [  -3.94294948   -3.91932543   -2.9404       -5.05266446]
 [ -13.17392746  -13.34637305  -14.89641159  -13.95140211]
 [ -12.42665153  -16.47657546  -29.01680815  -17.65084815]
```

```
[ -11.66567482  -16.68672111  -20.5102616   -17.8414342 ]
[ -10.86919026  -15.61148531  -32.42843662  -18.47134339]
[ -10.15820666  -16.98428296  -25.91981234  -17.14481683]
[  -9.25234558  -15.06992891  -26.94642106  -15.56501666]
[  -8.41619366  -12.26754446  -16.97101777  -16.54237379]
[  -7.53755978  -12.37958957  -29.74526021  -14.20447512]
[  -6.66052327   -9.36973255  -17.07372286  -12.09477326]
[  -7.35783232   -3.97032385  -15.92247357  -10.53564895]
[  -5.4088259    -3.02463449   -3.46600362   -5.67229515]
[  -4.08753469   -2.95444307   -1.98         -4.06757945]
[ -13.89391328  -14.71450139  -15.58969982  -14.75950388]
[ -13.34619252  -31.16288633 -126.36814692  -19.96574689]
[ -16.50520592  -34.06474865 -122.34482628  -25.91373421]
[ -16.90923146  -19.95245543 -110.86121513  -24.70884421]
[ -15.33529128  -29.425508   -108.8013712   -18.97302926]
[ -13.66564486  -17.65124734 -130.70426823  -22.69681827]
[ -13.53784948  -22.76811558 -117.95302017  -25.04003713]
[ -13.60826709  -24.78443046  -98.17996515  -21.38798081]
[ -10.39753932  -12.42324541 -121.66256427  -14.49835846]
[  -8.19363547  -14.65103168 -111.03711951  -20.27325864]
[  -6.59063812   -1.98406991 -125.98873843  -20.06085201]
[  -3.33434516   -1.98019898   -1.           -2.98974388]
[ -14.61157293 -114.40445387  -15.41952684  -16.28296638]
[   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.        ]
[   0.            0.            0.            0.        ]]
```

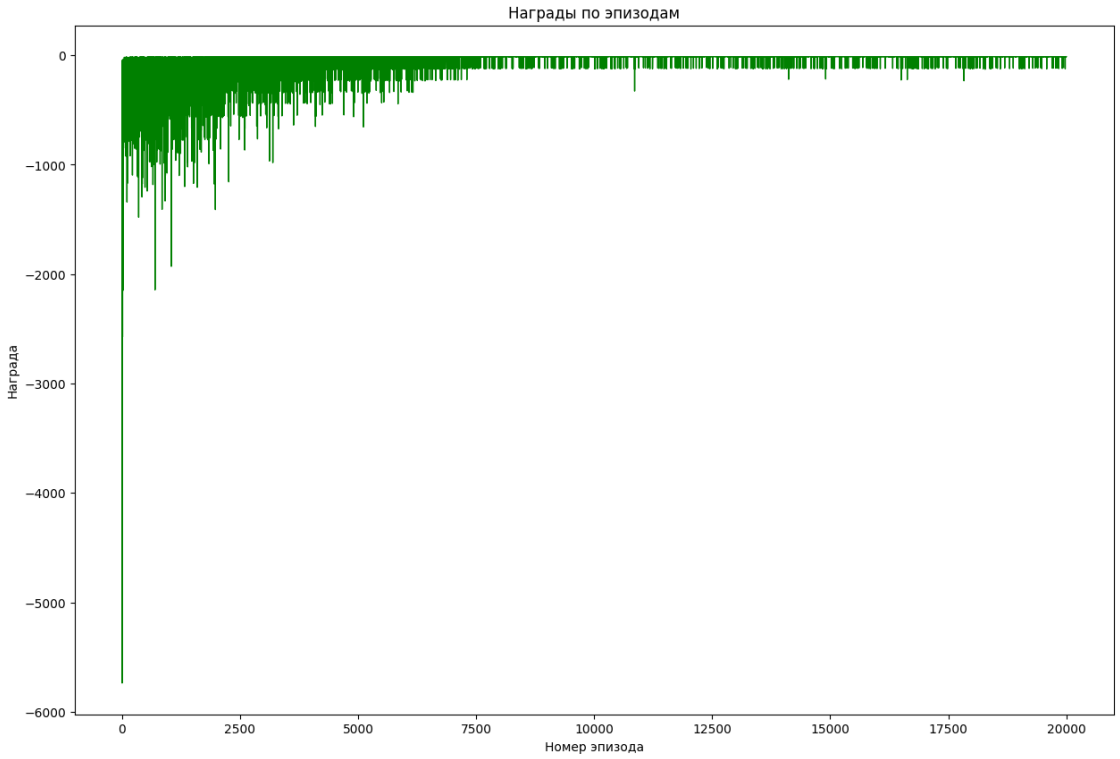


0.0.6 Q- : eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000

[ ]:
```
run_q_learning()
```

```
100%|      | 20000/20000 [00:02<00:00, 7060.42it/s]

    Q-              Q-
[[ -12.62716012  -12.30066353  -12.30187123  -12.53106295]
 [ -12.07218236  -11.54852805  -11.54851636  -12.11875414]
 [ -11.42779808  -10.76413781  -10.76413919  -12.16847347]
 [ -10.57702507   -9.96342977   -9.96342972  -11.42478967]
 [  -9.9027344    -9.14635907   -9.14635904  -10.7323342 ]
 [  -9.10277569   -8.31261174   -8.31261174   -9.80236621]
 [  -8.29075731   -7.46184883   -7.46184883   -9.13145237]
 [  -7.4603197    -6.59372333   -6.59372333   -8.30611621]
 [  -6.58453374   -5.70788095   -5.70788095   -7.43266689]
 [  -5.69808106   -4.80396016   -4.80396016   -6.54830251]
 [  -4.77048085   -3.881592     -3.881592     -5.67970024]
 [  -3.85517669   -3.74152451   -2.9404       -4.70627397]
 [ -13.03839926  -11.54888054  -11.54888054  -12.31783472]
 [ -12.31686436  -10.76416381  -10.76416381  -12.31789838]
 [ -11.54878676   -9.96343246   -9.96343246  -11.54888044]
 [ -10.76415382   -9.14635966   -9.14635966  -10.76416365]
 [  -9.96343031   -8.31261189   -8.31261189   -9.96343245]
```

```
[  -9.14635841   -7.46184887   -7.46184887   -9.14635965]
[  -8.3126105    -6.59372334   -6.59372334   -8.31261183]
[  -7.46184883   -5.70788096   -5.70788096   -7.46184884]
[  -6.59372304   -4.80396016   -4.80396016   -6.59372329]
[  -5.70788088   -3.881592     -3.881592     -5.70788091]
[  -4.80396012   -2.9404       -2.9404       -4.80396016]
[  -3.881592     -2.9404       -1.98         -3.881592  ]
[ -12.31790293  -10.76416381  -12.31790293  -11.54888054]
[ -11.54888054   -9.96343246 -111.31790293  -11.54888054]
[ -10.76416381   -9.14635966 -111.31790293  -10.76416381]
[  -9.96343246   -8.31261189 -111.31790293   -9.96343246]
[  -9.14635966   -7.46184887 -111.31790293   -9.14635966]
[  -8.31261189   -6.59372334 -111.31790293   -8.31261189]
[  -7.46184887   -5.70788096 -111.31790293   -7.46184887]
[  -6.59372334   -4.80396016 -111.31790293   -6.59372334]
[  -5.70788096   -3.881592   -111.31790293   -5.70788096]
[  -4.80396016   -2.9404     -111.31790293   -4.80396016]
[  -3.881592     -1.98       -111.31790293   -3.881592  ]
[  -2.9404       -1.98         -1.           -2.9404    ]
[ -11.54888054 -111.31790293  -12.31790293  -12.31790293]
[   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.        ]
[   0.            0.            0.            0.        ]]
```



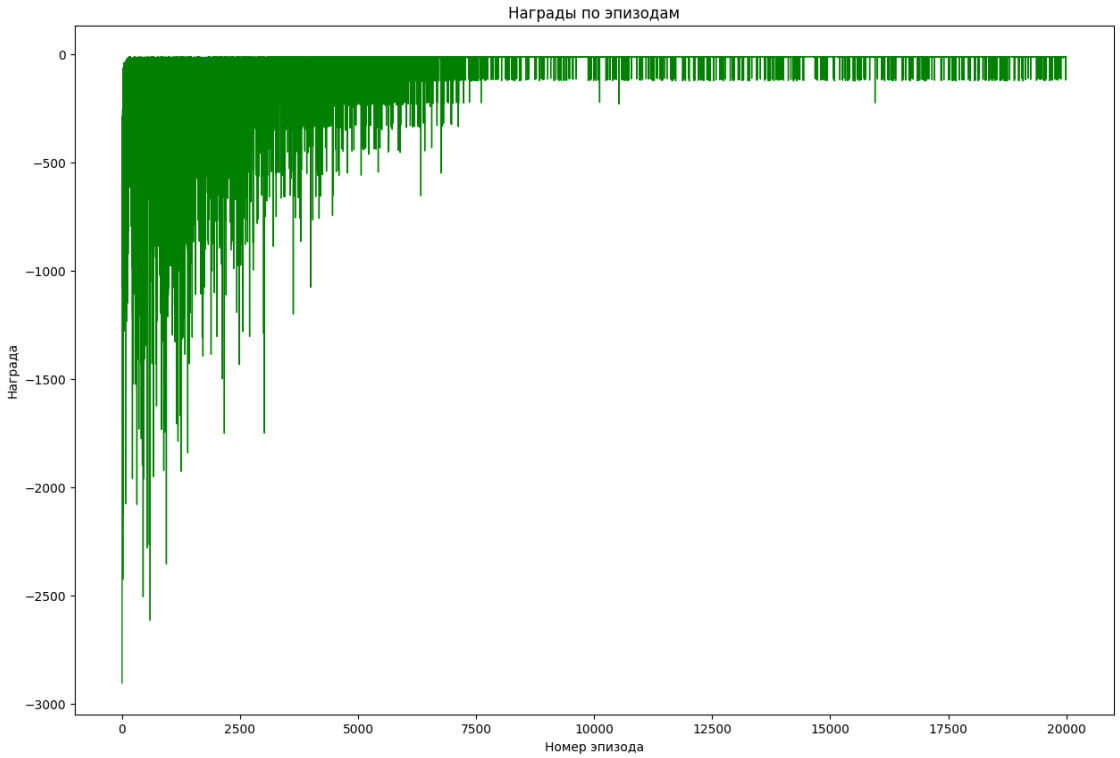
0.0.7 Q- : eps=0.4, lr=0.1, gamma=0.98, num_episodes=20000

[ ]:
```
run_double_q_learning()
```

```
  0%|          | 0/20000 [00:00<?, ?it/s]100%|     | 20000/20000
[00:02<00:00, 6718.46it/s]

    Q-                Q-
Q1
[[ -15.03480352  -13.96217453  -12.32230003  -14.82268257]
 [ -13.7147303   -13.2499411   -11.54906998  -14.33643105]
 [ -12.81266898  -14.47698313  -10.78932154  -14.66383856]
 [ -13.42803396  -11.47301851  -10.0188274   -12.13172171]
 [ -11.31325571  -10.77256639   -9.14882818  -12.7290895 ]
 [ -11.56242234  -10.65416174   -8.33248595  -11.30154597]
 [ -10.66153285  -10.61448073   -7.59024673  -11.20785512]
 [ -10.2525893    -9.65393915   -6.83089794  -10.31010697]
 [  -9.55535575  -11.34013494   -6.2468964   -10.4420707 ]
 [  -8.88901743   -5.15805226   -5.55624811   -7.49357761]
 [  -4.93387568   -3.90417767   -8.26339178   -8.97559953]
 [  -4.32036073   -4.28170187   -2.94061165   -4.91503919]
 [ -13.13117418  -11.54888054  -11.56698742  -12.34185644]
 [ -12.3192548   -10.76422298  -10.76416381  -12.31810624]
 [ -11.57107435   -9.98597752   -9.96343246  -11.55419573]
```

```
 [ -11.17201602   -9.14635966   -9.19267055  -10.79078294]
 [  -9.97576733   -8.31572057   -8.31261189   -9.96655246]
 [  -9.4434069    -7.51769995   -7.46184887   -9.25573982]
 [  -8.5863678    -6.82564741   -6.59372334   -8.34738322]
 [  -8.31345366   -6.0192416    -5.70788096   -7.54135764]
 [  -7.20641636   -4.89457495   -4.80396016   -7.10136596]
 [  -9.76495001   -6.5084585    -3.881592     -6.50636395]
 [  -4.90001143   -2.9404       -2.82206552   -4.77136939]
 [  -3.89792973   -2.93915145   -1.98         -5.62332046]
 [ -12.31790293  -10.76416381  -12.31790293  -11.54888054]
 [ -11.54888054   -9.96343246 -111.31790293  -11.54888054]
 [ -10.76416381   -9.14635966 -111.31790293  -10.76416381]
 [  -9.96343246   -8.31261189 -111.31790293   -9.96343246]
 [  -9.14635966   -7.46184887 -111.31790293   -9.14635966]
 [  -8.31261189   -6.59372334 -111.31790293   -8.31261189]
 [  -7.46184887   -5.70788096 -111.31790293   -7.46184887]
 [  -6.59372334   -4.80396016 -111.31790293   -6.59372334]
 [  -5.70788096   -3.881592   -111.31790293   -5.70788096]
 [  -4.80396016   -2.9404     -111.31790293   -4.80396016]
 [  -5.20634293   -1.98       -111.31790293   -3.881592  ]
 [  -2.9404       -1.98         -1.           -2.9404    ]
 [ -11.54888054 -111.31790293  -12.31790293  -12.31790293]
 [   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.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]]
Q2
[[ -15.13898105  -14.32381453  -12.33766728  -15.42049931]
 [ -13.30661702  -13.22022211  -11.54896719  -14.7253205 ]
 [ -13.1569111   -12.12323786  -10.76501363  -12.65767022]
 [ -12.3223195   -12.85927305  -10.02728743  -13.53417177]
 [ -11.93460369  -10.49832192   -9.14701041  -12.43739036]
 [ -11.18535804  -11.52614607   -8.36375519  -10.92710904]
 [ -10.30942737   -9.9026903    -7.52550216  -10.10652428]
 [ -10.24995713  -10.91493566   -6.73108938  -10.38467829]
 [  -9.66191857   -8.47767518   -5.82286404   -9.18476604]
 [  -7.98984369   -9.06477142  -10.36917949   -8.39480054]
 [  -7.14394848   -3.88935579   -3.0064895    -7.80777961]
 [  -4.10619778   -4.14614242   -2.94043699   -7.63097265]
 [ -13.16545749  -11.54888054  -11.55525827  -12.33208842]
 [ -12.31920849  -10.76419416  -10.76416381  -12.31794082]
```

```
[ -11.71263134   -9.97211536   -9.96343246  -11.55285812]
[ -11.03648746   -9.14635966   -9.15281165  -10.77331658]
[  -9.97479528   -8.32735686   -8.31261189   -9.9663456 ]
[  -9.30734431   -7.54761851   -7.46184887   -9.24280442]
[  -8.76268076   -6.71879852   -6.59372334   -8.41714579]
[  -8.24483529   -5.80260824   -5.70788096   -7.58570949]
[  -8.25061721   -5.69000502   -4.80396016   -6.86921384]
[  -9.25068217   -3.90290821   -3.881592     -5.97879333]
[  -8.93289972   -2.9404       -4.29191055   -5.14470667]
[  -3.88546839   -2.94250158   -1.98         -3.87540829]
[ -12.31790293  -10.76416381  -12.31790293  -11.54888054]
[ -11.54888054   -9.96343246 -111.31790293  -11.54888054]
[ -10.76416381   -9.14635966 -111.31790293  -10.76416381]
[  -9.96343246   -8.31261189 -111.31790293   -9.96343246]
[  -9.14635966   -7.46184887 -111.31790293   -9.14635966]
[  -8.31261189   -6.59372334 -111.31790293   -8.31261189]
[  -7.46184887   -5.70788096 -111.31790293   -7.46184887]
[  -6.59372334   -4.80396016 -111.31790293   -6.59372334]
[  -5.70788096   -3.881592   -111.31790293   -5.70788096]
[  -4.80396016   -2.9404     -111.31790293   -4.80396016]
[  -3.881592     -1.98       -111.31790293   -3.881592  ]
[  -2.9404       -1.98         -1.           -2.9404    ]
[ -11.54888054 -111.31790293  -12.31790293  -12.31790293]
[   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.        ]
[   0.            0.            0.            0.        ]]
```




```
The Kernel crashed while executing code in the the current cell or a previous␣
↪cell. Please review the code in the cell(s) to identify a possible cause of␣
↪the failure. Click <a href='https://aka.ms/vscodeJupyterKernelCrash'>here</a>␣
↪for more info. View Jupyter <a href='command:jupyter.viewOutput'>log</a> for␣
↪further details.
```