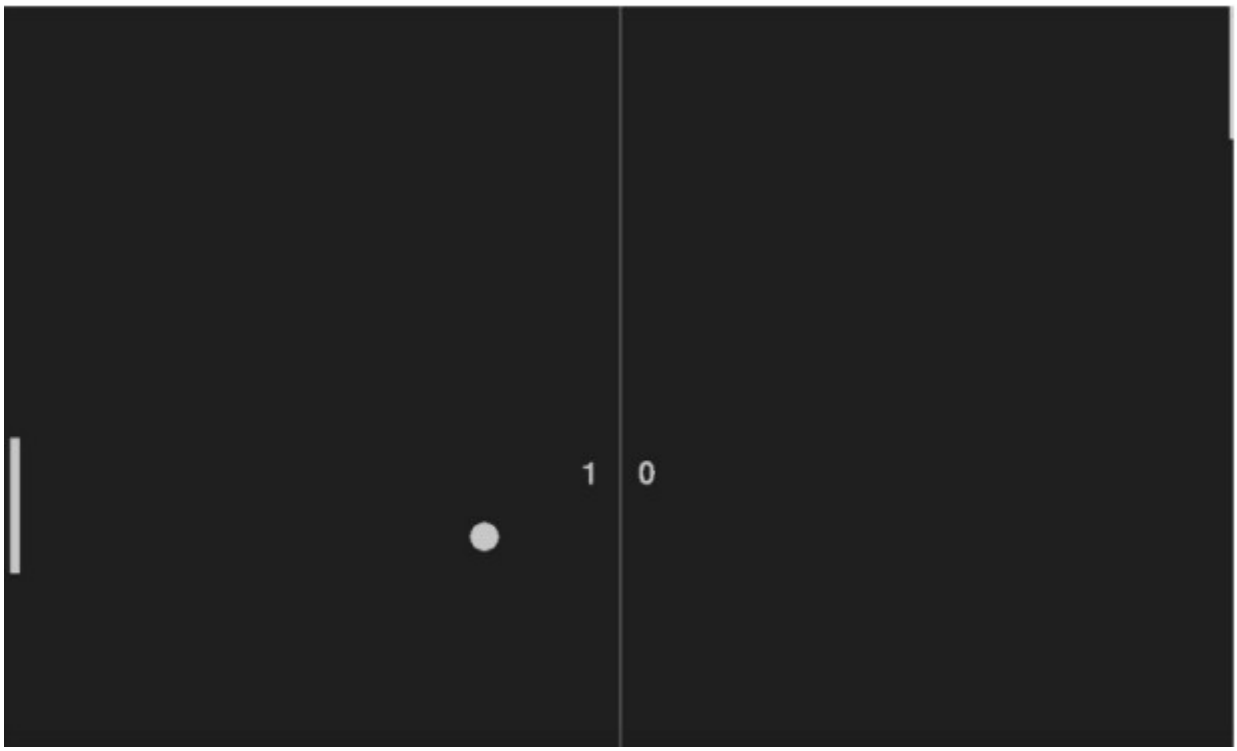
Université Abdelmalek Essaad Faculté ses Sciences et techniques de Tanger
Département Génie Informatique

Atelier 4
«Qlearning / PONG »



Réalisée par :
KAISSI Houda

Objective :

l’objective principal de cet atelier est d’implémenter l’algorithme Q-learning dans un agent qui va
contrôler la barre player du jeu PONG.

Outils :

Python ,Pygame, matplotlib, Qlearning, Numpy, Matplotlib.

1. Développez la classe Agent qui contient tous les méthodes nécessaires pour assurer les
fonctionnalités de algorithme Qlearning.

[4]:
```
def get_action(self, s):
    return np.argmax(self.Q[s, :])
```

[5]:
```
def reward(rect, bar, ball):
    if bar.top <= ball.centery <= bar.bottom:
        return 1
    else:
        return -1
```

```
def state(self, centre, screen_height, bar_height):
    a = 0
    b = bar_height
    s = 0

    for i in range(int(screen_height / bar_height)):
        if a < centre < b:
            s = (b / bar_height) - 1
        else:
            a += bar_height
            b += bar_height

    return int(s)
```

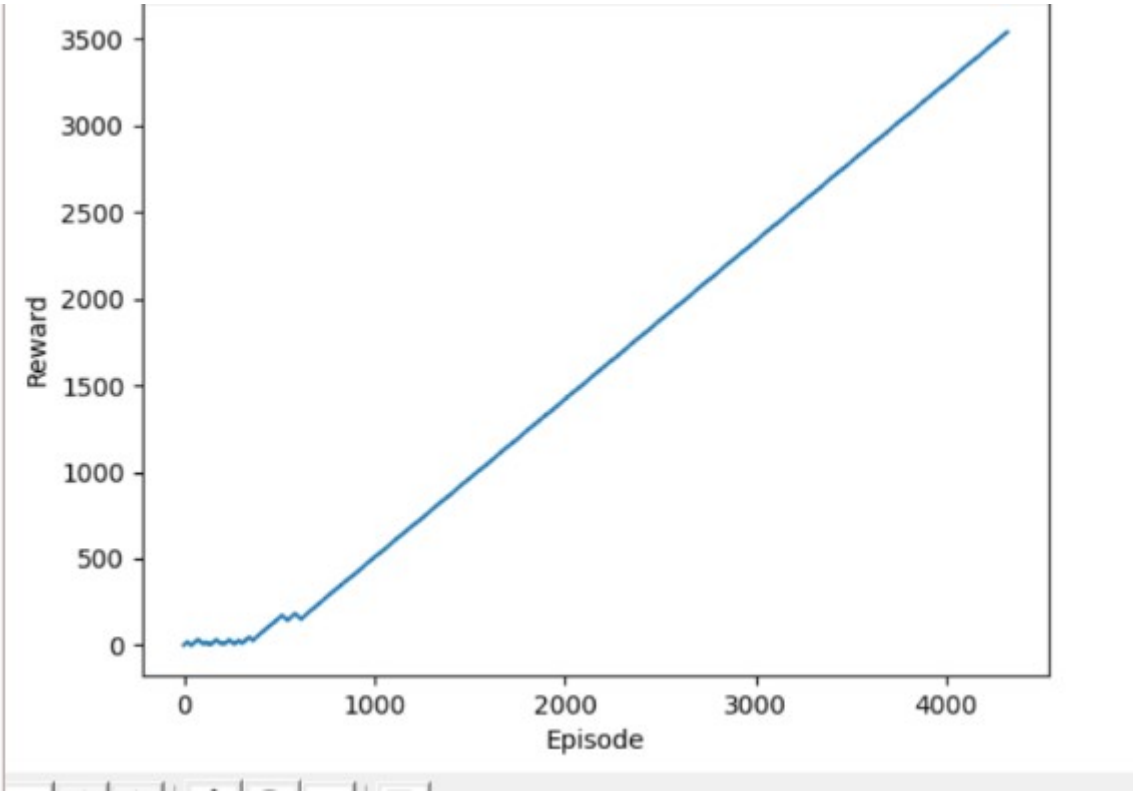
```
def update(self, s, bar, ball, screen_height, ball_speed_x, is_permanent):
    s_ = s
    position_cal = bar.right + 10
    speed = ball_speed_x * (-1)
    ballX = ball.x

    if not is_permanent:
        position_cal = bar.left - 10 - ball.width
        speed = ball_speed_x
        ballX = position_cal

    position_cal = ball.x

    if position_cal <= ballX and speed > 0:
        reward = self.calculate_reward(bar, ball)
        self.rewards.append(reward)
        self.action = self.get_action(s)

        if self.action != 0:
            s_ = self.centre_to_state(ball.centery, screen_height, bar.height)
        else:
            s_ = s

        if s_ < 0:
            s_ = 0
        elif s_ > int(screen_height / bar.height) - 1:
            s_ = int(screen_height / bar.height) - 1

        self.state = s
        self.Q[s, self.action] += self.alpha * (reward + self.gamma * np.max(self.Q[s_, :]) - self.Q[

    return s_ * bar.height
```

2 . Intégrez l’agent au niveau de jeu pour qu’il prend le rôle de joueur, en mode Agent RL vs Agent AI.

```
if __name__ == '__main__':
    gl = GameLearning()
    episodes = 20
    gl.beginPlaying(episodes)
```

3. Dessiner le graphe reward « plot_agent_reward» , puis faire une synthèse globale sur la solution
proposée.

```
def plot_agent_reward(rewards):
    """ Function to plot agent's accumulated reward vs. iteration """
    plt.plot(np.cumsum(rewards))
    plt.title('Agent Cumulative Reward vs. Iteration')
    plt.ylabel('Reward')
    plt.xlabel('Episode')
    plt.show()
```



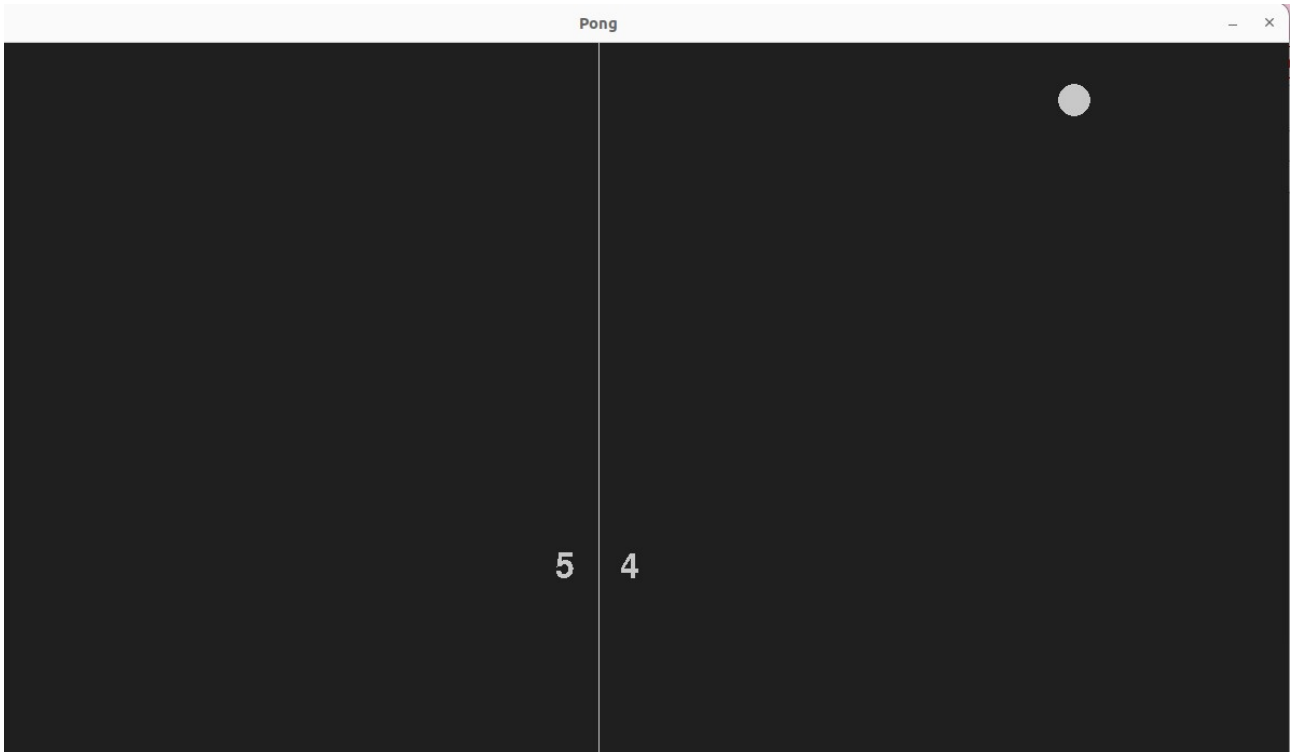
synthèse globale :

Initialement ignorant de son environnement, l'agent accroît sa connaissance au fil de nouveaux épisodes, ce qui se traduit par une augmentation progressive de la récompense.

4. il faut Refaire la même chose en mode Agent RL vs Humain .

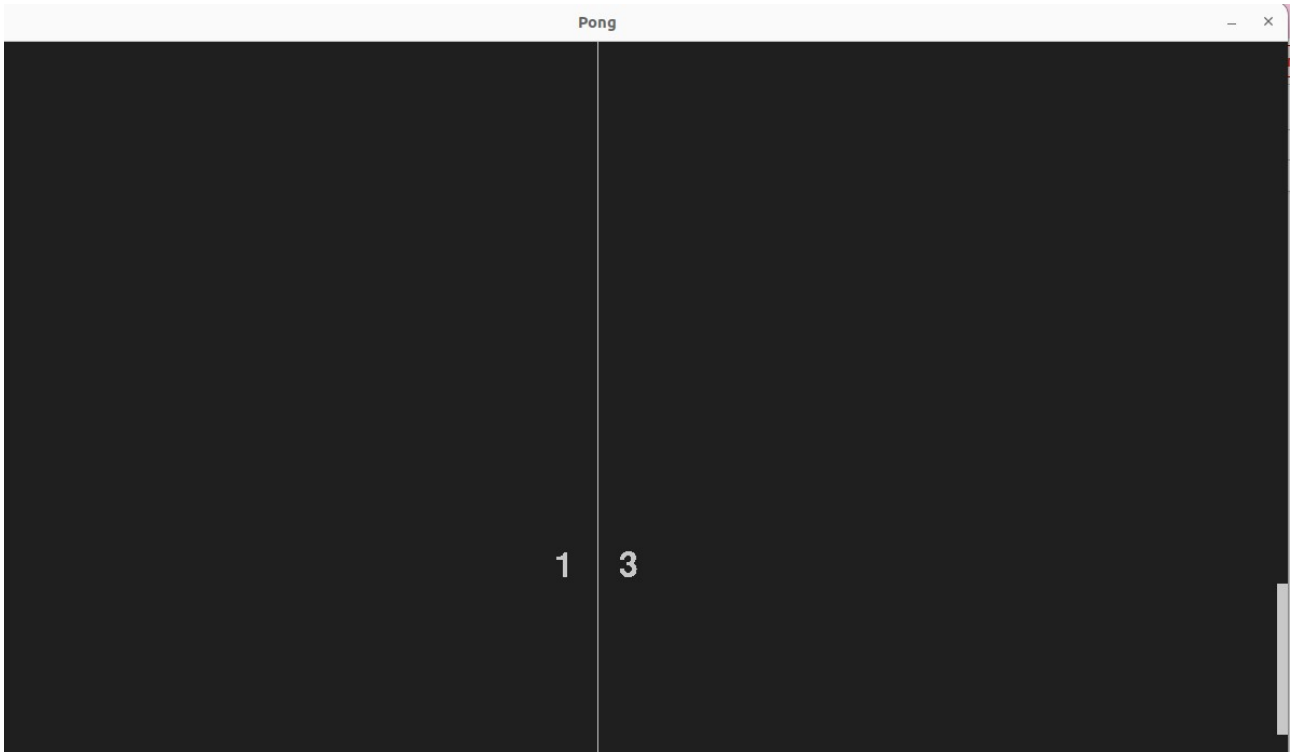
```
class GameLearning:

    def __init__(self):
        while True:
            print('\n--- Choose a mode: --- ')
            type = input('1. AgentRL vs AgentAI \n2. AgentRL vs Human \n3. AgentRL vs AgentRL\nMode nº = ')
            if type == '1' or type == '2' or type == '3':
                break
        if type == '1':
            self.game = g.Game('agentAI')
        elif type == '2':
            self.game = g.Game('human')
        else:
            self.game = g.Game('agentRL')

    def beginPlaying(self):
        self.game.play()

if __name__ == '__main__':
    gl = GameLearning()
    gl.beginPlaying()
```

```
--- Choose a mode: ---
1. AgentRL vs AgentAI
2. AgentRL vs Human
3. AgentRL vs AgentRL
Mode nº = 2
```



5. il faut Refaire la même chose en mode Agent RL vs Agent RL

```
--- Choose a mode: ---
1. AgentRL vs AgentAI
2. AgentRL vs Human
3. AgentRL vs AgentRL
Mode nº =  3
```


Pong
1
3