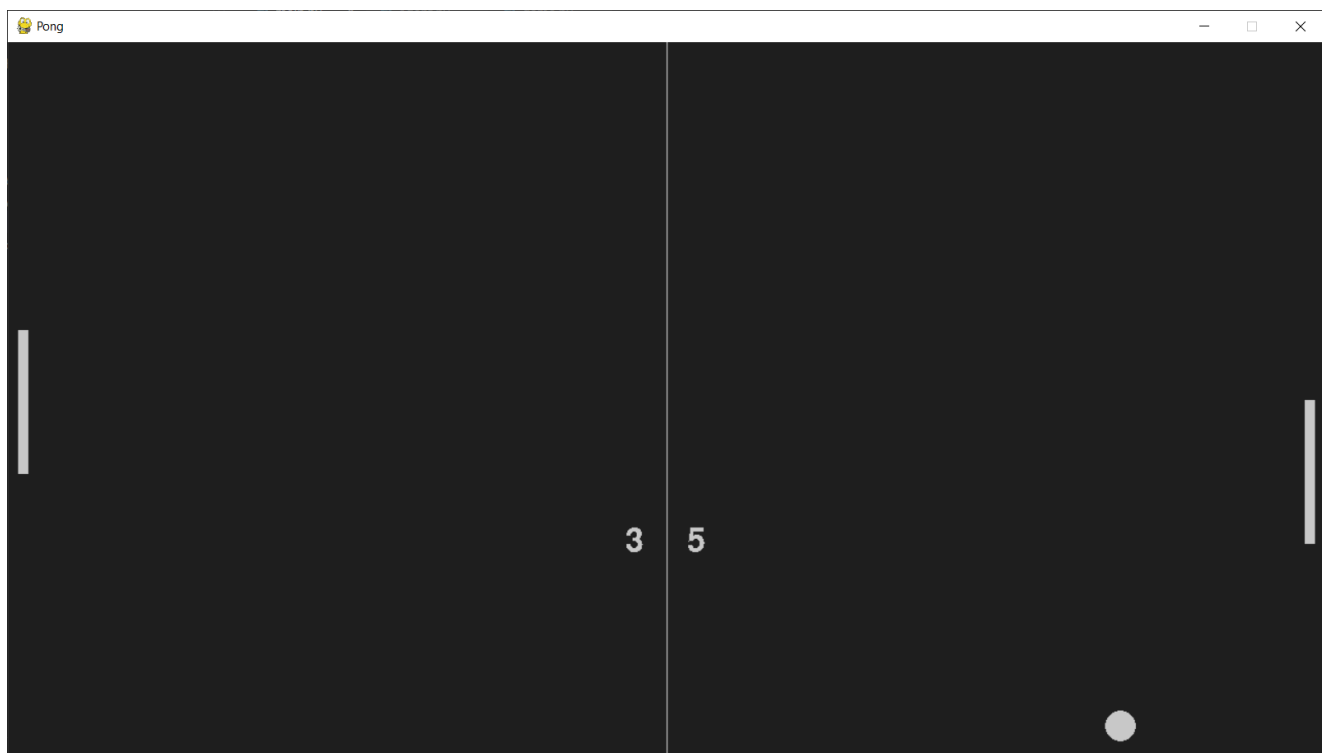


# Machine Learning

## Atelier 4: «Q Learning / PONG »



Réalisé par:  
DAAOUAN Mohammed

Encadré par:  
Pr . EL AACHAK LOTFI

## **Objective:**

l'objectif 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

## **Etape 1: Développement de la classe Agent**

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

def calculate_reward(rect, bar, ball):
    if bar.top <= ball.centery <= bar.bottom:
        return 1
    else:
        return -1

def centre_to_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[s, self.action])
    return s_ * bar.height

```

## **Etape 2:** L'intégration d'agent au niveau de jeu

```

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()

```

## **Etape 3:** Le graphe reward

L'agent commence sans connaissance de l'environnement. En explorant de nouveaux épisodes, son apprentissage progresse, conduisant à une augmentation des rewards. Ajustant ses actions en fonction de cette expérience, l'agent améliore ses performances au fil du temps.

Agent Cumulative Reward vs. Iteration

