import numpy as np

def calculate\_payoff(player1\_action, player2\_action):

payoff\_matrix = np.array([[4, 1], [3, 2]])

return payoff\_matrix[player1\_action][player2\_action]

def find\_nash\_equilibrium(payoff\_matrix):

num\_actions = payoff\_matrix.shape[0]

# Oyuncu 1 için Nash dengesi stratejisi

player1\_strategy = np.zeros(num\_actions)

max\_payoff = float('-inf')

for action in range(num\_actions):

expected\_payoff = np.max(payoff\_matrix[action, :])

if expected\_payoff > max\_payoff:

max\_payoff = expected\_payoff

player1\_strategy = np.zeros(num\_actions)

player1\_strategy[action] = 1

elif expected\_payoff == max\_payoff:

player1\_strategy[action] = 1

# Oyuncu 2 için Nash dengesi stratejisi

player2\_strategy = np.zeros(num\_actions)

max\_payoff = float('-inf')

for action in range(num\_actions):

expected\_payoff = np.max(payoff\_matrix[:, action])

if expected\_payoff > max\_payoff:

max\_payoff = expected\_payoff

player2\_strategy = np.zeros(num\_actions)

player2\_strategy[action] = 1

elif expected\_payoff == max\_payoff:

player2\_strategy[action] = 1

return player1\_strategy, player2\_strategy

def check\_pareto\_efficiency(payoff\_matrix, player1\_strategy, player2\_strategy):

num\_actions = payoff\_matrix.shape[0]

for action1 in range(num\_actions):

payoff1 = np.dot(payoff\_matrix[action1, :], player2\_strategy)

for action2 in range(num\_actions):

payoff2 = np.dot(payoff\_matrix[:, action2], player1\_strategy)

if payoff1 < payoff\_matrix[action1, action2] and payoff2 < payoff\_matrix[action1, action2]:

return False

return True

# Ödeme matrisi

payoff\_matrix = np.array([[4, 1], [3, 2]])

# Nash denge stratejilerini bulma

player1\_strategy, player2\_strategy = find\_nash\_equilibrium(payoff\_matrix)

print("Player 1 Nash Strategy:", player1\_strategy)

print("Player 2 Nash Strategy:", player2\_strategy)

# Pareto etkinliğini kontrol etme

pareto\_efficient = check\_pareto\_efficiency(payoff\_matrix, player1\_strategy, player2\_strategy)

if pareto\_efficient:

print("The outcome is Pareto efficient.")

else:

print("The outcome is not Pareto efficient.")

# Ödeme matrisini kontrol etme

print("Payoff Matrix:")

print(payoff\_matrix)

# Stratejileri kontrol etme

print("Player 1 Strategy Payoffs:")

for action1 in range(payoff\_matrix.shape[0]):

payoff1 = np.dot(payoff\_matrix[action1, :], player2\_strategy)

print("Action", action1, "Payoff:", payoff1)

print("Player 2 Strategy Payoffs:")

for action2 in range(payoff\_matrix.shape[0]):

payoff2 = np.dot(payoff\_matrix[:, action2], player1\_strategy)

print("Action", action2, "Payoff:", payoff2)