```python
import random
import numpy as np
from scipy.optimize import linprog


your_penny=0
person_penny=0


games2play = int(input('How many games would you like to play?\n'))
possible_actions = ["heads", "tails"]
while True:

    if games2play == 0:
        print("\nRANDOM SELECTION")
        print(f"Your penny: {your_penny}")
        print(f"Person's penny': {person_penny}")
        break

    person_action_random = random.choice(possible_actions)
    print("\n --------------------------\n")
    user_action = input("Enter a choice (heads, tails): \n")
    print(f"You chose {user_action}, person chose {person_action_random}.")
    if user_action == "heads" and person_action_random == "heads":
        print("You took a penny.")
        your_penny+=1
        person_penny-=1

    elif user_action == "heads" and person_action_random == "tails":
        print("You lost a penny.")
        your_penny-=1
        person_penny+=1

    elif user_action == "tails" and person_action_random == "heads":
        print("You lost a penny")
        your_penny-=1
        person_penny+=1

    elif user_action == "tails" and person_action_random == "tails":
        print("You took a penny.")
        your_penny+=1
        person_penny-=1

    else:
        print("Unexpected input.")

    games2play-=1
```

```
How many games would you like to play?
2

 --------------------------

Enter a choice (heads, tails): 
heads
You chose heads, person chose heads.
You took a penny.

 --------------------------

Enter a choice (heads, tails): 
tails
You chose tails, person chose tails.
You took a penny.

RANDOM SELECTION
Your penny: 2
Person's penny': -2
```

```python
def find_mixed_nash_equilibrium(payoff_matrix):
    # Create the objective function for player 1
    c1 = [-1 for _ in range(len(payoff_matrix))]

    # Create the constraints for player 1
    A1 = np.transpose(payoff_matrix)
    b1 = [1 for _ in range(len(payoff_matrix[0]))]

    # Solve the linear program for player 1
    result1 = linprog(c1, A_ub=A1, b_ub=b1)

    # Player 1's mixed strategy
    player1_strategy = result1.x
    # Create the objective function for player 2
    c2 = [1 for _ in range(len(payoff_matrix[0]))]

    # Create the constraints for player 2
    A2 = payoff_matrix
    b2 = [-1 for _ in range(len(payoff_matrix))]

    # Solve the linear program for player 2
    result2 = linprog(c2, A_ub=A2, b_ub=b2)

    # Player 2's mixed strategy
    player2_strategy = result2.x

    return player1_strategy, player2_strategy

# Main program
if __name__ == "__main__":
    print("Welcome to the Mixed-Strategy Nash Equilibrium Finder for Zero-Sum Games!")

    # Define the payoff matrix (replace this with your own matrix)
    payoff_matrix = np.array([[1, -1],
                              [-1, 1]])

    print("Payoff Matrix:")
    print(payoff_matrix)

    player1_strategy, player2_strategy = find_mixed_nash_equilibrium(payoff_matrix)

    print("Mixed-Strategy Nash Equilibrium:")
    print(f"Player 1's mixed strategy: {player1_strategy}")
    print(f"Player 2's mixed strategy: {player2_strategy}")
```

```
Welcome to the Mixed-Strategy Nash Equilibrium Finder for Zero-Sum Games!
Payoff Matrix:
[[ 1 -1]
 [-1  1]]
Mixed-Strategy Nash Equilibrium:
Player 1's mixed strategy: None
Player 2's mixed strategy: None
```