

```

import random

from copy import deepcopy


# ----- Constants -----

BLACKJACK = 21

DEALER_STANDS = 17

STARTING_MONEY = 1000

MIN_BET = 1

USERS_FILE = "users.txt"


# ----- User persistence -----

def load_users(filename=USERS_FILE):
    """Load user bankrolls from file"""
    users = {}
    try:
        with open(filename, "r") as f:
            for line in f:
                name, money = line.strip().split(",")
                users[name] = int(money)
    except FileNotFoundError:
        pass
    return users


def save_users(users, filename=USERS_FILE):
    """Save user bankrolls to file"""
    with open(filename, "w") as f:

```

```

    for name, money in users.items():

        f.write(f"{name},{money}\n")


# ----- Blackjack logic -----

def create_deck():

    """Create a standard deck of cards"""

    return (

        ['A'] * 4 +

        [2, 2, 2, 2] +

        [3, 3, 3, 3] +

        [4, 4, 4, 4] +

        [5, 5, 5, 5] +

        [6, 6, 6, 6] +

        [7, 7, 7, 7] +

        [8, 8, 8, 8] +

        [9, 9, 9, 9] +

        ['J', 'J', 'J', 'J'] +

        ['Q', 'Q', 'Q', 'Q'] +

        ['K', 'K', 'K', 'K']

    )


def total(hand):

    """Calculate hand total with blackjack rules"""

    subhand = deepcopy(hand)

    subtotal = 0

    # Face cards = 10

```

```

for face in ['K', 'Q', 'J']:
    count = subhand.count(face)
    subtotal += 10 * count
    for _ in range(count):
        subhand.remove(face)
# Aces handled after
aces = subhand.count('A')
subhand = [card for card in subhand if card != 'A']
subtotal += sum(subhand) if subhand else 0
# Add aces optimally
while aces > 0 and subtotal <= 10:
    subtotal += 11
    aces -= 1
subtotal += aces # any remaining aces = 1 each
return subtotal

```

```

def is_blackjack(hand):
    """True if exactly two cards: an Ace + a 10-value card"""
    if len(hand) != 2:
        return False
    vals = set(hand)
    return 'A' in vals and any(c in vals for c in [10, 'J', 'Q', 'K'])

```

```

def deal_initial_hands(deck):
    """Deal 2 cards each"""
    player = [deck.pop(), deck.pop()]

```

```
dealer = [deck.pop(), deck.pop()]

return player, dealer
```

```
def play_dealer(deck, dealer_hand):

    """Dealer plays until reaching DEALER STANDS"""

    while total(dealer_hand) < DEALER STANDS:

        dealer_hand.append(deck.pop())

    return dealer_hand
```

```
def play_round(deck, player_hand):

    """Handle player actions"""

    while True:

        print(f"\nYour hand: {player_hand} (total: {total(player_hand)})")

        if total(player_hand) >= BLACKJACK:

            break

        move = input("Hit or stand? (h/s): ").lower()

        if move == "h":

            player_hand.append(deck.pop())

        elif move == "s":

            break

        else:

            print("Invalid input. Please enter h or s.")

    return player_hand
```

```
def check_winner(player_hand, dealer_hand, bet, bankroll):

    """Determine result and adjust bankroll, including 3:2 blackjack payout"""
```

```

player_total = total(player_hand)
dealer_total = total(dealer_hand)

print(f"\nYour hand: {player_hand} (total: {player_total})")
print(f"Dealer hand: {dealer_hand} (total: {dealer_total})")

# --- 3:2 payout for natural blackjack ---
# Natural blackjack means exactly 2 cards totaling 21
player_blackjack = len(player_hand) == 2 and player_total == BLACKJACK
dealer_blackjack = len(dealer_hand) == 2 and dealer_total == BLACKJACK

if player_blackjack and dealer_blackjack:
    print("Both you and the dealer have Blackjack – Push!")
    return bankroll
elif player_blackjack:
    payout = int(bet * 1.5)
    print(f"Blackjack! You win ${payout} (3:2 payout).")
    bankroll += payout
    return bankroll
elif dealer_blackjack:
    print("Dealer has Blackjack. You lose.")
    bankroll -= bet
    return bankroll

# --- Standard outcomes ---
if player_total > BLACKJACK:

```

```
    print("You bust! You lose.")
    bankroll -= bet
elif dealer_total > BLACKJACK:
    print("Dealer busts! You win.")
    bankroll += bet
elif player_total > dealer_total:
    print("You win!")
    bankroll += bet
elif dealer_total > player_total:
    print("You lose.")
    bankroll -= bet
else:
    print("Push! It's a tie.")
return bankroll
```

```
# ----- Game flow -----
```

```
def main():
    users = load_users()

    # Login
    name = input("Enter your player name: ").strip()
    bankroll = users.get(name, STARTING_MONEY)
    print(f"Welcome {name}! Your bankroll is ${bankroll}")

    while True:
```

```
# ----- Restart option when broke -----
```

```
if bankroll <= 0:
```

```
    print("\nYou're out of money!")
```

```
    choice = input("Do you want to restart with $1000? (y/n): ").lower()
```

```
    if choice == "y":
```

```
        bankroll = STARTING_MONEY
```

```
        print("Bankroll reset to $1000.")
```

```
    else:
```

```
        break
```

```
# ----- Place bet -----
```

```
while True:
```

```
    try:
```

```
        bet = int(input(f"\nPlace your bet (minimum ${MIN_BET}, you have ${bankroll}): "))
```

```
        if bet < MIN_BET:
```

```
            print(f"Minimum bet is ${MIN_BET}.")
```

```
        elif bet > bankroll:
```

```
            print("You don't have that much money.")
```

```
        else:
```

```
            break
```

```
    except ValueError:
```

```
        print("Please enter a valid number.")
```

```
# ----- Deal cards -----
```

```
deck = create_deck()
```

```
random.shuffle(deck)
```

```
player_hand, dealer_hand = deal_initial_hands(deck)

print(f"\nDealer shows: {dealer_hand[1]}")


# Player turn

player_hand = play_round(deck, player_hand)


# Dealer turn if player hasn't busted
if total(player_hand) <= BLACKJACK:
    dealer_hand = play_dealer(deck, dealer_hand)


# Determine result and update bankroll

bankroll = check_winner(player_hand, dealer_hand, bet, bankroll)

print(f"Your bankroll is now ${bankroll}")


# Save progress

users[name] = bankroll

save_users(users)


again = input("\nPlay again? (y/n): ").lower()

if again != "y":
    break


print(f"\nThanks for playing, {name}! Final bankroll: ${bankroll}")

users[name] = bankroll

save_users(users)
```

```
if __name__ == "__main__":  
    main()
```