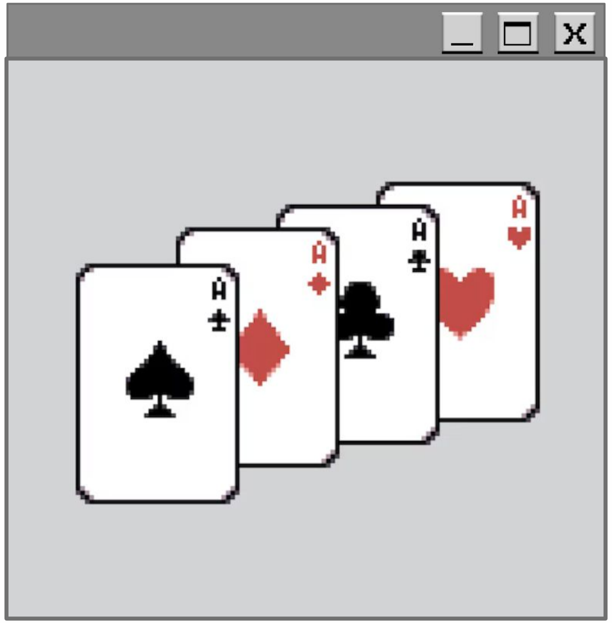
Final Project - CS30

Texas Hold'em Poker Simulator in Python



Project Overview

01 A simple but complete simulation of Texas Hold’em Poker

02 Two-player game: user input + random shuffling

03 Terminal-based interaction with card dealing, hand evaluation, and winner output

04 Designed using Python’s basic principles: functions, classes, control structures

Motivation

Why we chose this project:

- Fun, interactive, and engaging for audiences
- Great way to practice what we learned: loops, lists, dictionaries, classes
- Real-world application: simulating logic and fairness in card games
- Chance to explore structured thinking and probability

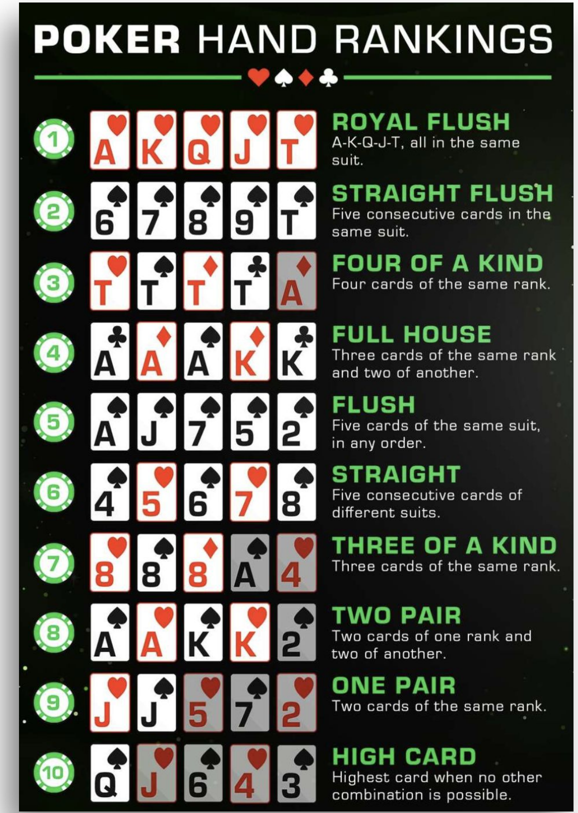
Core Features

1. Card, Deck, and Player classes
2. Shuffling and dealing cards
3. Community card logic (flop, turn, river)
4. Hand evaluation (e.g., Flush, Straight, Full House)
5. Winner comparison with tiebreaker logic

User Interaction Demo

How the game runs:

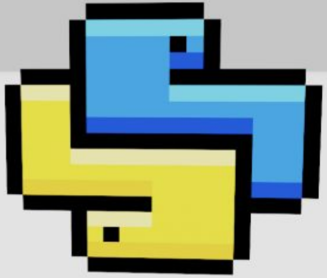
- Players input their names
- Cards dealt in real time (with user input prompts)
- Cards shown clearly in terminal
- Result printed with hand type and winner



POKER HAND RANKINGS

1	A♥ K♥ Q♥ J♥ T♥	ROYAL FLUSH	A-K-Q-J-T, all in the same suit.
2	6♠ 7♠ 8♠ 9♠ T♠	STRAIGHT FLUSH	Five consecutive cards in the same suit.
3	T♥ T♠ T♦ T♣ A♦	FOUR OF A KIND	Four cards of the same rank.
4	A♣ A♦ A♠ K♦ K♣	FULL HOUSE	Three cards of the same rank and two of another.
5	A♠ J♠ 7♠ 5♠ 2♠	FLUSH	Five cards of the same suit, in any order.
6	4♠ 5♥ 6♠ 7♥ 8♠	STRAIGHT	Five consecutive cards of different suits.
7	8♥ 8♠ 8♦ A♠ 4♥	THREE OF A KIND	Three cards of the same rank.
8	A♠ A♥ K♠ K♥ 2♠	TWO PAIR	Two cards of one rank and two of another.
9	J♥ J♠ 5♥ 7♠ 2♥	ONE PAIR	Two cards of the same rank.
10	Q♠ J♥ 6♠ 4♥ 3♠	HIGH CARD	Highest card when no other combination is possible.

So how does this code work?



Let's go through our game code!

Define Card Information and Card Class

```
SUITS = ['♠', '♥', '♦', '♣']
RANKS = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K', 'A']
RANK_VALUES = {rank: i for i, rank in enumerate(RANKS, 2)}
HAND_RANKS = {
    "High Card": 1, "One Pair": 2, "Two Pair": 3, "Three of a Kind": 4,
    "Straight": 5, "Flush": 6, "Full House": 7, "Four of a Kind": 8, "Straight Flush": 9
```

1. Defines the suits and ranks of a standard poker deck.
2. Assigns numeric values to each card rank (for sorting and comparing).
3. HAND_RANKS gives numeric values to each poker hand (used for comparison).

```
class Card:
    def __init__(self, suit, rank):
        self.suit = suit
        self.rank = rank
        self.value = RANK_VALUES[rank]

    def __str__(self):
        return f"{self.rank}{self.suit}"
```

1. Represents a single card (like A♠ or 10♦).
2. Stores suit, rank, and numeric value (e.g., A = 14).
3. __str__ lets us display the card nicely.

Deck Class and Player Class

```python
class Deck:
    def __init__(self):
        self.cards = [Card(s, r) for s in SUITS for r in RANKS]
        random.shuffle(self.cards)

    def deal(self, num):
        return [self.cards.pop() for _ in range(num)]

class Player:
    def __init__(self, name):
        self.name = name
        self.hand = []

    def __str__(self):
        return f"{self.name} Hand: {' '.join(str(card) for card in self.hand)}"
```

1. Builds a full 52-card deck.
2. Shuffles the deck using random.shuffle.
3. deal(num) gives out num cards and removes them from the deck.
4. Each player has a name and a hand of 2 cards.
5. Easy to print their name and current cards using __str__.

evaluate_hand() and compare_hands()

```
def evaluate_hand(cards):
    …
(Because the code is too long,
it is not shown in full.)
```

1. Analyzes 7 cards (2 hand + 5 community) to find the best hand.
2. Uses Counter to detect patterns like pairs, threes, etc.
3. Checks for: Straight Flush → Four of a Kind → Full House → Flush → …
4. Returns: hand name, main value, and kickers (tie-breakers).

```
def compare_hands(player1, player2, community):
    ...
(Because the code is too long, it is not shown in full.)
```

1. Combines each player’s hand with the community cards.
2. Uses evaluate_hand() to get each hand’s strength.
3. Compares
4. Returns winner's name or “Draw”.

Dealing Community Cards and One Round of Poker

```python
def deal_community(deck):
    cards = deck.deal(3)
    print("Flop:", ' '.join(str(c) for c in cards))
    cards += deck.deal(1)
    print("Turn:", str(cards[3]))
    cards += deck.deal(1)
    print("River:", str(cards[4]))
    return cards

def play_poker_round(name1, name2):
    deck = Deck()
    p1, p2 = Player(name1), Player(name2)
    p1.hand, p2.hand = deck.deal(2), deck.deal(2)

    print(p1)
    print(p2)

    community = deal_community(deck)
    print("Community Cards:", ' '.join(str(c) for c in community))

    winner = compare_hands(p1, p2, community)
    print(f"\nResult: {'It\'s a draw!' if winner == 'Draw' else winner + ' wins!'}")
```

1. Deals 5 community cards in 3 steps:
2. Flop (3 cards)
3. Turn (1 card)
4. River (1 card)
5. Prints each step so players can follow.
6. Creates a new deck and two players.
7. Deals hands and community cards.
8. Displays everything and announces the winner.

Dealing Community Cards and One Round of Poker

```python
def main():
    print("Welcome to Texas Hold'em!")
    n1 = input("Enter Player 1 name: ")
    n2 = input("Enter Player 2 name: ")
    while True:
        play_poker_round(n1, n2)
        if input("Play again? (y/n): ").lower() != 'y':
            break

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

Starts the game.-Gets player names.-Runs rounds in a loop.-Asks if players want to continue.

Thank you