Texas Hold'em Poker Game Final Project Report

By Andy, Hongyi, and Jerry

Introduction

Our project is a simplified program of Texas Hold'em Poker implemented in Python. The program allows two players to compete in a single round of poker, during which each person is dealt two private cards and shares five community cards. We chose this topic for our project because it is technically rich and intriguing yet accessible to beginners. It gave us the opportunity to apply the core concepts we learned in CS30, such as loops and functions, as well as the use of external libraries. Additionally, the interactive and fun nature of this project made it more engaging to build and present.

Algorithm

The poker game is run from the command line. First, the user is prompted to enter the names of two players. Then, the program randomly deals two private cards to each player from a shuffled deck. As the game progresses, the user presses Enter to reveal each stage of community cards: the flop (three cards), the turn (one card), and the river (one card). Once all cards have been revealed, each player's hand is evaluated based on standard poker rules. The program compares the strength of both hands and prints the winner or declares a tie.

The interaction is fully text-based and intuitive. Users receive prompts and instructions before each major stage of the game. After each round, they are asked if they want to play again.

Implementation

The following are the major components of our implementation:

Card Class: Represents a single card with a suit, rank, and numeric value.

Deck Class: Initializes a full deck of 52 cards and contains methods to shuffle and deal cards.

Player Class: Stores a player's name and their private hand.

Evaluate Hand Function: Analyzes a player's seven-card hand (two private and five community cards) and determines the poker hand rank.

The compare_hands function: Compares the evaluated hands of two players and determines the winner.

DealCommunity Function: Handles the process of dealing and printing the community cards.

play_poker_round function: Orchestrates a single round of the game, tying together all other components.

Main Function: Handles user interaction and supports playing multiple rounds.

We also used two Python libraries:

random for shuffling the deck and selecting cards.

collections.Counter to count occurrences of card values when evaluating hand ranks.

Some key variables include:

SUITS and RANKS: The global constants defining all possible card types.

RANK_VALUES: A dictionary mapping each rank to a numeric value used for comparison.

HAND_RANKS: A dictionary assigning numerical strength to each poker hand type for easy comparison.

Debugging and Challenges

For the entire project schedule, we spread the whole work into different pieces. They are “Constants”, “Classes”, “Hand Evaluation”, “Compare two players”, “Main Game”, and “Main loop”. This strategy helps us save a lot of time. But we still meet some problems. Since this project was worked together by different people, it is hard for us to combine all the pieces together. For instance, we will use the same variable names to define different things. In this case, the second version silently replaces the first one, or code breaks due to unexpected behaviors. Under this condition, we spent a lot of time going over every function in this code and checking its values.

We encountered several bugs when comparing hands, particularly with regard to tie-breaking and identifying specific hand types, such as a Full House or a Straight Flush. We resolved these issues by reading the game rules online, making the workflow of the program clear to us, asking ChatGPT for coding advice, and testing with sample card combinations. We also used documentation to understand unfamiliar libraries, such as how the Counter works.

To check whether the code is working as we designed it to, we used many samples to run the code and check what this function returns. This method works well, especially in the part of “Hand evaluation” because this part requires a great deal of “if” statements, which makes the entire function complex. As we did previously, we separate this function into different parts, each subproblem represents a different card combination. Thus, what we need to do is to make sure the code is still working properly after the combination.

Conclusion

This project helped us understand how to build a complete application by combining small, manageable parts. It improved our understanding of loops, conditional statements, and functions in the python language. We also learned game logic and other syntaxes like classes by ourselves during the program. It also gave us a chance to work on a project through group collaboration, debugging, and technical communication.

If we had more time, we would add a betting system to simulate actual poker rounds more realistically and monitor the players' stakes. We're also interested in developing the game to a multiplayer one over a network or adding graphics using a library like PyGame. Moreover, we could leverage the Pygame library to create a game interface for each player on different devices, so that their private cards remain truly private. Currently, we are proud of what we accomplished and grateful for the learning experience.

Reference:

ChatGPT: for poker rules, library documentation, and optimization advice.

Python Official Documentation: for list, class, and Counter usage.

Stack Overflow: for examples of card shuffling and hand comparison logic.