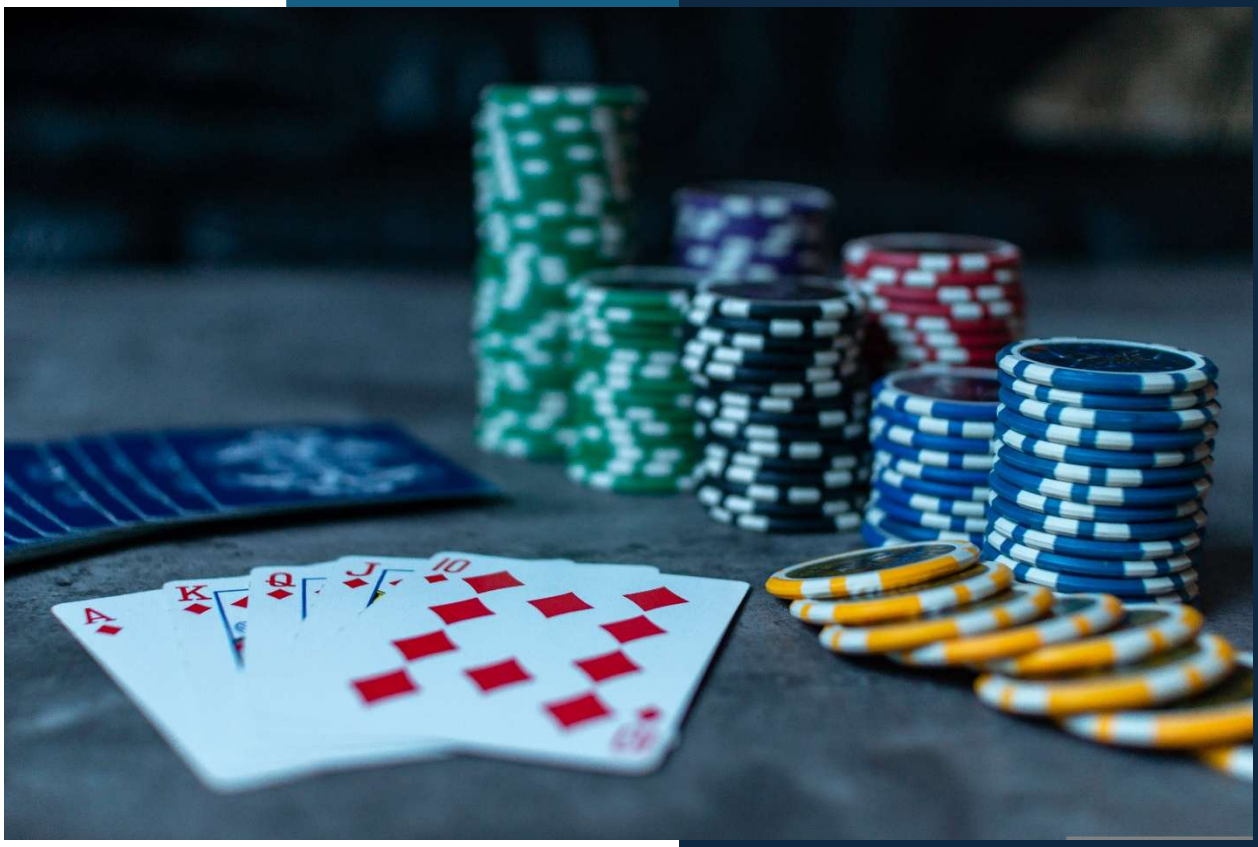
Summer 2026

Texas Hold'em Betting Program

Aurelia Miller



Aurelia Miller

Mentored by Oliver Andrew & William Gasarch

Summer 2026

Table of Contents

Introduction 2
The Assignment 3
Key Elements of the Program and Implementation Lessons Learned 4
    Addressing the Pre-flop Scenario 4
    After the Flop 4
Future Enhancements 7
Conclusion/ Summary 7
Appendix 8

Introduction

Poker is a popular card game around the world. The global poker economy, which includes live casino games, physical tournaments, and online apps, is estimated to generate over $150 billion annually.[1] Texas Hold'em is the most widely played version of Poker, its rules almost synonymous with the game itself. "Solving poker" is a challenge placed before many a programmer, with Pokerbot competitions hosted to encourage young programmers to try their hands at the classic problem.

Under the guidance of Mr. Andrew and Dr. Gasarch, I spent the summer of 2026 attempting to "solve (Texas Hold'em) poker" through the lens of probability and combinatorics. Over the course of three months, six high school students and I independently created our own poker-playing programs to be tested against each other as the conclusion of the internship program.

This report documents the process and output of the work I conducted over the summer, starting with a description of the assignment, and the output produced followed by a discussion of key elements of my approach to programming my poker bot. I conclude with a discussion of areas for refinement of the models I created to "solve poker."

Texas Hold'em Breakdown

Texas Hold'em's ruleset can be intimidating, but here is a brief overview of what was considered for this project:

a) Cards

Each player is dealt 2 cards each private to them and unknown and inaccessible to other players; this gamestate is referred to as "pre-flop". After this, 3 communal cards, known as "the flop", are revealed to the table. This is followed by "the turn", in which another communal card is added, and afterwards "the river", the final communal card. Ultimately, each player has access to 7 cards: 2 private to themselves, and 5 communal cards that are revealed gradually.

b) Actions

After each distribution of cards, including the initial, private one, players are given the option to fold, call, raise, or sometimes check. Folding is bowing out of the game, thereby surrendering any previous wagered money to the pot but avoiding wasting more money on a poor hand. Calling is simply paying enough to stay in the game without raising the stakes, but can be a tell, especially before the flop. Raising is increasing the amount each player must pay to stay in the game and in the process increasing the total winnings for the game.

c) Showdown

[1] How Much Revenue Does the Poker Market Generate Worldwide? MercoPress, June 2025
.

After the river and a final round of player actions, the private cards the remaining engaged players hold are revealed, and whichever player is able to make the highest-ranked 5-card hand with the 7 cards at their disposal wins the pot. Ties of the rank of hand type (for example, two players both with a pair) are resolved through the ranks of the cards in the hand, and after that, by the ranks of the cards included in the final hand played but not the hand type (the other 3 cards a player with a pair chose to include). These secondary tiebreakers are called "kicker(s)".

The Assignment

Here was our assignment:

Programming Tasks
1. Given two cards, rate how good they are by returning any of the options above.
2. Given your two cards, and three cards on the table, rate how good they are by returning any of the options above.
3. Given your two cards, and four cards on the table, rate how good they are by returning any of the options above.
4. Given your two cards, and five cards on the table, rate how good they are by returning any of the options above.

Programming was done in Python.

After this initial assignment was completed, its scope was expanded, moving beyond the consideration of one's cards (pre-flop) as the key factor in the bot's decision on how to play. This more advanced step asked students to enhance their models to consider player positioning at the poker table along with the actions of other players when refining their pokerbot.

Over the weeks of summer, I worked to develop and refine my pokerbot, meeting with the other students on the project and occasionally with Dr. Gasarch to report on progress, challenges and solutions. My final program can be found here:
https://github.com/alansai/pokerproblems/blob/main/pokerproblemsaurelia.py

In addition to the programming tasks described, I was also tasked with an independent assignment to explore the mathematics of Texas Hold'em and practice simulations that test mathematical hypotheses. I documented the output of this work in a separate report submitted to Dr. Gasarch. This report also appears in the appendix.

Key Elements of the Program and Implementation Lessons Learned

Throughout the development of my program, I created decision rules that determined how my pokerbot operates. In this section, I discuss key decisions I made when creating my program, along with the rationale for them.

Addressing the Pre-flop Scenario

- **Use of widely available pre-flop charts guided program development.** In the pre-flop scenario, every player has been dealt two cards but no cards are on the table where they would be visible to all players. I spent some time attempting to use probabilities to create rules for the bot to operate under this scenario. After reading up on the topic, I opted to use pre-flop charts to guide bot decisions. Pre-flop charts are resources for human poker players that efficiently sort every possible hand by whether the player holding it should fold or raise. The specific pre-flop charts used are attached in the appendices.

- **My program only ever raises or folds in the pre-flop scenario but never calls.** This is in accordance with the pre-flop charts' recommendations. To simply call the previous players' bet in the pre-flop scenario is considered to be a pretty low-level poker play because it provides the table with additional information about the quality of one's hand, a significant disadvantage in poker. As a result, I avoided this play ("call") entirely in my program.

- **Only cards and not table position guided programming decisions**. While pre-flop charts differ based on the position held at the table, we had been instructed to not account for anything but the cards visible to the player/the program in the initial program (we added these features later), so I picked a flop chart for a middling position (specifically the hijack position in 6-player games) to create the bot.

After the Flop

In the flop, three cards are placed on the table for everyone to see. The following elements of the program's development created rules based on having the additional information provided to the table by these cards.

- **A function was created to identify the best possible poker hand with the information a player can see dynamically throughout the game, starting with 5 cards after the flop.** Between the two cards held in the player's hand and the three cards on the table, this function checks for every possible poker hand in descending order to ensure that a full house is not passed up for a pair. This function was then adapted for after the turn, when there are four cards on the table, and after the river, when all five cards are out. Special attention was given to this final gamestate after the river is out to consider the fact that there is no more future probability to analyze. Eventually, this function was refined so that it could not only identify the best hand type available (full house, straight, two pair, etc), but also the best specific hand within that category (for example, a flush including an ace and a king being better than a flush including a 2 and a 7, since in this gamestate there are seven cards available and thus up to two could be "extras" for a flush).

- **Combination formulas were pivotal in analyzing the future potential of a hand.** The combination or choose function, where $\binom{n}{k}$ is represented as nCr(n, k) in python's math library, was used extensively to determine what better hands might be possible in the future of a game. For example, achieving a flush when two spades are currently held and another two turned up in the flop is likely, especially compared to gamestates where it is impossible to do so. The combination functions in this program were used to analyze the odds of the player achieving a certain ideal hand as well as the probability of other players getting a good hand.

- **Another function was created for the program to analyze the influence of other players' hands.** Using the same framework and adapting it to only consider the pooled table cards as cards currently held allowed an analysis of the opponents' odds. For example, if the turn comes out and reveals a fourth club in a row, the odds that some player has a flush are relatively high. Nesting the probabilities that an opponent player had cards that the program's player needed or some similar scenario rapidly increased complexity. As a result, the program does not account for these changes in probability.

- **Using the previously established tools, the user interface and analysis advisory were constructed.** The main body of the program takes and considers inputs from the player, feeding it into the program's functions. It also takes outputs from the functions and weighs and analyzes them to offer the user its advice. For example, this program weights the user's odds of hitting a certain hand higher than the opponents, to avoid overly cautious playing. After rating the entered positioning with a numerical score, the program instructs the user on what action to take: to raise by some amount, to call, or to fold. Attached below are some screenshotted examples of the user interface.

```
How many players (including yourself) are in the game?
6
AFTER THE FLOP, what is your position in terms of turn order?   (ex. small blind =
1, under the gun = 3, etc)
4
What stage are you in?   (P = pre-flop, F = flop, T = turn, R = river)
P

Guide: ten of hearts = Th, ace of spades = As, 2 of diamonds = 2d
What is the first card in your hand?
4c
What is the second card in your hand?
6s
Fold!

=== Code Execution Successful ===
```

```
How many players (including yourself) are in the game?
6
AFTER THE FLOP, what is your position in terms of turn order?   (ex. small blind =
1, under the gun = 3, etc)
5
What stage are you in?   (P = pre-flop, F = flop, T = turn, R = river)
F

Guide: ten of hearts = Th, ace of spades = As, 2 of diamonds = 2d
What is the first card in your hand?
As
What is the second card in your hand?
Ts
What is the first card on the table?
4h
What is the next card on the table?
9s
What is the last card on the table?
Jd
Has anyone raised this turn?    (y/n)
n
Check!
```

After an initial presentation of functionality, the program was updated following additional guidelines from Dr. Gasarch. After a meeting demonstrating the program's capabilities in an actual poker game, Dr. Gasarch expanded the project's parameters. The program was updated to consider the user's position at the table, which primarily manifested in the pre-flop scenario, and the actions of other players. However, less time was allotted for this part of the project, so these mechanics ended up less fleshed out than other parts of the program.

Conclusions

While there were certainly limitations to this project, there are conclusions to be drawn from it. For one, it is possible to make a Texas Hold'em-playing program of decent quality. While this program likely would not find much success in a casino, neither do most human players. When aiming high, this program seems to leave much to be desired, but, ultimately, it was a successful project.

When critically examining this project, it is important to consider the data collected about its performance. When the different, independently developed programs of our intern group were tested against one another, some were consistently more successful than others. While the data gathered were limited, causing luck to play a greater role in the success or failure of any given program, programs that accounted for more of the context of the game, as a human player might, were more successful. One program in particular sticks out in memory for its ability to play and

win even with a poor hand; by taking other players' actions and its own position as inputs and considering them thoroughly, in tandem with its hand, this program was even able to bluff, betting even when it had poor hands if it concluded that other players did, too. The reactive elements of these programs are clearly worth further investigation and development. It would be interesting, as well, to see how a program with well-programmed reactions but poorly-programmed assessment of its own specific cards performs in comparison to a program good at analyzing its own cards but weaker at reacting to other players.

Future Enhancements

In terms of the specific program I produced, further development of additional mechanics is a clear and desirable extension of the project. Among possible enhancements, I highlight the following:

- **Player position and other players' cards were incorporated into the program but were not as robust.** At the tail end of the project, we considered how other players and table positioning would influence the bot rules. Due to limited time available, the analysis of these data was not as thoroughly implemented and tested. Additional testing and refinements of the program to account for these key dynamic elements of the game should be a priority.

- **Additional testing would be ideal to improve the quality of the program's advice.** Testing of this program's quality was limited to sessions with human input that were both time-consuming and infrequent due to scheduling limitations. Using computer simulations of the bots created by each student would expedite this process and accelerate refinement. Additional, automated testing would allow for human analysis of the program's flaws and its own play tendencies and corresponding adjustments. For example, it could lead to changing the weighting of other players' odds of hitting high-ranking hands to account for an observed tendency of the program to play recklessly.

- **The user interface should be improved.** With the focus of the development process on the functionality of the program and the quality of its analysis and advice, the user interface received less attention. As such, the program has to be repeatedly re-run and each time the data must be re-entered in order for it to get through a single game. Small issues with following input format guidelines on the user end can compound the difficulties of this process. As a result, another clear next step for this project is adjusting the interface to be better designed and less choppy, which would also, in turn, contribute to the ability to analyze other players' actions.

Conclusion

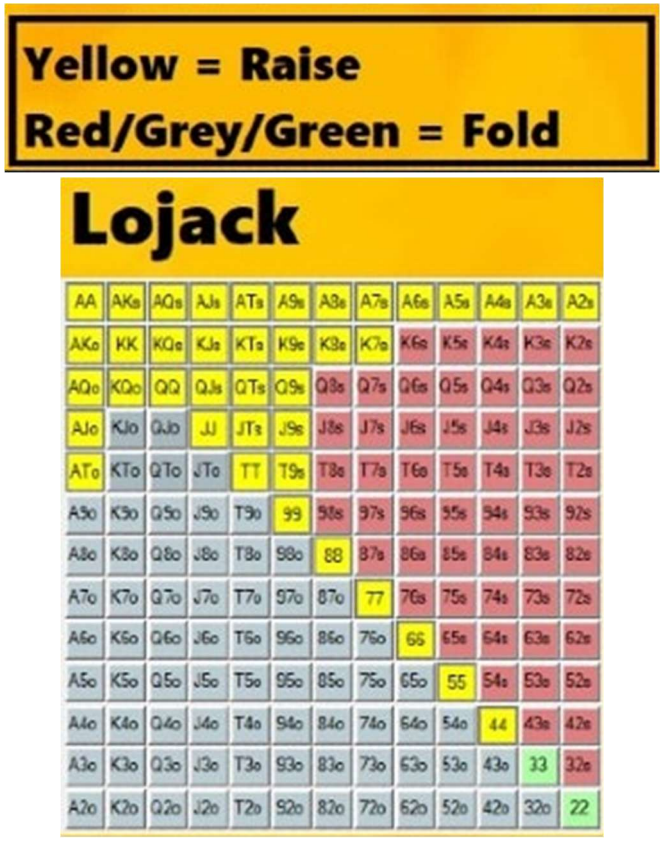
This report has detailed the process of developing the advisory poker program assigned by Dr. Gasarch. This work was completed throughout the summer of 2026 as part of the Montgomery Blair STEM Magnet Senior Research Program. To summarize:

- The first obstacle was addressing the pre-flop scenario, which taught me an important lesson about having a more flexible mindset.

- Before working on the post-flop calculations, I wrote three functions to provide a stable framework.
- After I added final touches in the user interface, Dr. Gasarch asked us to account for the amount and actions of other players.
- Due to limited time, some features were not implemented as, so clear future steps include better reaction to other players and improving the interface.

Appendices

Appendix A: Pre-Flop Charts



Hijack

AA	AKs	AQs	AJs	ATs	A9s	A8s	A7s	A6s	A5s	A4s	A3s	A2s
AKo	KK	KQs	KJs	KTs	K9s	K8s	K7s	K6s	K5s	K4s	K3s	K2s
AQo	KQo	QQ	QJs	QTs	Q9s	Q8s	Q7s	Q6s	Q5s	Q4s	Q3s	Q2s
AJo	KJo	QJo	JJ	JTs	J9s	J8s	J7s	J6s	J5s	J4s	J3s	J2s
ATo	KTo	QTo	JTo	TT	T9s	T8s	T7s	T6s	T5s	T4s	T3s	T2s
A9o	K9o	Q9o	J9o	T9o	99	98s	97s	96s	95s	94s	93s	92s
A8o	K8o	Q8o	J8o	T8o	98o	88	87s	86s	85s	84s	83s	82s
A7o	K7o	Q7o	J7o	T7o	97o	87o	77	76s	75s	74s	73s	72s
A6o	K6o	Q6o	J6o	T6o	96o	86o	76o	66	65s	64s	63s	62s
A5o	K5o	Q5o	J5o	T5o	95o	85o	75o	65o	55	54s	53s	52s
A4o	K4o	Q4o	J4o	T4o	94o	84o	74o	64o	54o	44	43s	42s
A3o	K3o	Q3o	J3o	T3o	93o	83o	73o	63o	53o	43o	33	32s
A2o	K2o	Q2o	J2o	T2o	92o	82o	72o	62o	52o	42o	32o	22

Cutoff

AA	AKs	AQs	AJs	ATs	A9s	A8s	A7s	A6s	A5s	A4s	A3s	A2s
AKo	KK	KQs	KJs	KTs	K9s	K8s	K7s	K6s	K5s	K4s	K3s	K2s
AQo	KQo	QQ	QJs	QTs	Q9s	Q8s	Q7s	Q6s	Q5s	Q4s	Q3s	Q2s
AJo	KJo	QJo	JJ	JTs	J9s	J8s	J7s	J6s	J5s	J4s	J3s	J2s
ATo	KTo	QTo	JTo	TT	T9s	T8s	T7s	T6s	T5s	T4s	T3s	T2s
A9o	K9o	Q9o	J9o	T9o	99	98s	97s	96s	95s	94s	93s	92s
A8o	K8o	Q8o	J8o	T8o	98o	88	87s	86s	85s	84s	83s	82s
A7o	K7o	Q7o	J7o	T7o	97o	87o	77	76s	75s	74s	73s	72s
A6o	K6o	Q6o	J6o	T6o	96o	86o	76o	66	65s	64s	63s	62s
A5o	K5o	Q5o	J5o	T5o	95o	85o	75o	65o	55	54s	53s	52s
A4o	K4o	Q4o	J4o	T4o	94o	84o	74o	64o	54o	44	43s	42s
A3o	K3o	Q3o	J3o	T3o	93o	83o	73o	63o	53o	43o	33	32s
A2o	K2o	Q2o	J2o	T2o	92o	82o	72o	62o	52o	42o	32o	22



Source: [Beasts of Poker]