Sharad Varadarajan Project 1 Reflection

For my project, I prompted a single user to play the casino game roulette. For background on some of the basics of Roulette, please see the Appendix at the end of this document.

Final Design

Class	*Methods*	*Major Class Variables*
player	*main_menu* *bet* *payout* *funds* *results* *catch_exceptions_and_return_numbers* *quit*	*Name* *Net_profit* *Funds* *Numbers* *Wagers* *Past_bets* *Past_winnings* *Bets* *Bet_outcomes* *Winnings* *Num_of_games* *Betting_categories* *Betting_odds* *Wheel_number*
bet_type	*straight_up* *corner* *split* *column* *street* *line* *color* *even_odd* *dozen* *high_low*	
wheel	*spin* *current_results*	

How To Use My Program

The prompts are very descriptive, and should prevent you from getting lost within the program. However, below I will provide a high-level overview

When you run the program, you will immediately be prompted to enter your name → once you do so, you will arrive at the main menu with a bank of $2000.

At the main menu, you have 4 options to choose from:

1) Place a bet
2) View funds
3) View results from previous bets
4) Quit

During your first iteration of the game, it is expected that you will first place a bet but feel free to choose the other options and see what they look like to start with.

When you choose to place a bet, you will see the 10 available bet types. Enter the number that corresponds to the bet type you would like to place. Once that is done, you will need to provide the amount you would like to wager on that bet type→ you will be unable to enter money that exceeds your current funds.

Following this, you are given a description on the bet type you chose and will be asked to choose a specific bet associated with that bet type. You have now successfully placed your first bet! Your funds balance will update and the possible winnings you can reap from that bet are displayed. You now have the option to spin the wheel or place another bet → feel free to place as many bets as you want. If you choose to bet all of your funds in one round, then following your final bet of the round, the wheel will be spun automatically (rather than asking if you if you want to place any more bets). Therefore a user will never have the chance to place a bet with zero funds

Once you decide to spin the wheel (or the wheel spins automatically), you'll receive a quick summary of your round activity and then the winning number will be revealed!

Once you enter yes, to proceed to the results, each bet will be analyzed against the winning number. Detailed outcomes for each of your bets will be shown, including your winnings/losses for the round and your updated funds.

At this point, if you have no funds remaining, the program will end and you will see a good-bye message

However if you still have funds remaining, you will be re-directed to the View Funds menu, showing your current balance. You can choose to return to the main menu or cash out. → Cashing out will end the program and you will see a good-bye message

If you return to the main menu, you can now make use of the View Results from Previous Bets screen. This page will include summarized results from each round you play within the program. Feel free to keep playing more rounds, until you lose all funds or cash out. Have fun!

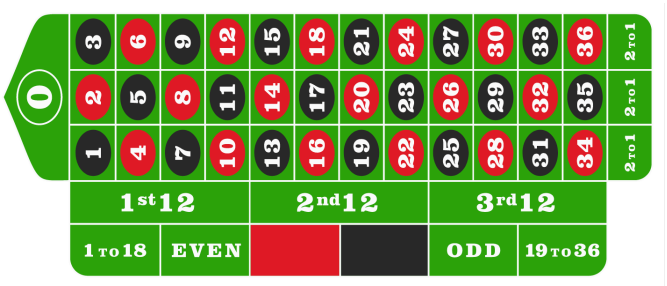
Challenges

This was definitely a fun project! Really neat seeing a lot of the concepts we've used come together to form an interactive program. One of the hardest challenges for me was trying to keep true to the initial design I proposed. It turns out deeper analysis was required on my part to sort out how all of the different classes/methods were going to interact with one another. It was also a challenge keeping track of the countless variables I was using to store all of the different game details. There were a lot of "uh-oh" moments when I ran my code and saw numbers weren't adding up.

However the hardest challenge I faced came after I completed the program; I had well over 600 lines of codes. It was tough condensing my code, because I was scared to break apart a program that was working. However, I was able to make my code more efficient by creating the catch_exceptions_and_return_results method, which took input from each method in my bet_type class, allowing me to remove repetitive try/except statements. Additionally I leveraged dictionary objects to remove complicated if/elif conditionals when users choose from one of the ten available bet types→ After my changes, I was able to condense my code down to 422 lines

Appendix

"In this game, a player chooses to place bets on either a single number, various groupings of numbers, the colors red or black, whether the number is odd or even, or if the numbers are high (19–36) or low (1–18)."



"To determine the winning number and color, a dealer spins a wheel in one direction, then spins a ball in the opposite direction around a tilted circular track running around the circumference of the wheel. The ball eventually loses momentum, passes through an area of deflectors, and falls onto the wheel and into one of 38 colored and numbered pockets on the wheel."

There are numerous bets that a player can make in a single iteration of the game, each with different payouts/odds of success. Additionally there is no rule preventing a player from betting on more than one option.