Vincent Muller
Python Fundamentals Course
12/16/2022

Portfolio Project Report

Introduction

For my Python fundamentals portfolio project, I decided to build a single player roulette casino game. Roulette is a casino game named after the French word meaning little wheel which was likely developed from the Italian game Biribi. In the game, a player may choose to place a bet on a single number, various groupings of numbers, the color red or black, whether the number is odd or even, or if the numbers are high (19–36) or low (1–18). A player can choose to place a single bet or multiple bets, with each winning bet returning a payout according to the associated rate.

I wanted this application to have the same options as a traditional digital roulette game, therefore it features the following functionality:

- Registration page for new players to create an account
- Log In page where existing players can log in and access existing balances
- Deposit page where players can deposit starting and additional funds
- Gameplay page where players can choose from a list of different game play options
    - Place a bet
    - Spin the roulette wheel
    - Clear all bets
    - Save and log out
    - Cash out current funds
    - Deposit additional funds
- Option to exit game

Design and Implementation

One of the main reasons I picked to build a Roulette game application was because I knew the multiple variables, necessary data structures, and betting options would make the logic and algorithm challenging. I used multiple methods for designing this application include: whiteboard drawings, thinking and talking to myself while rocking my daughter, and trialing small components of the application separately. Please see below for details on each section of the game:

Creating Player Accounts

I decided to use a class for new player accounts because all of the players would have the same 3 attributes: name, password, and balance. Because I want players to be able to save funds and access at a later date, I also determined a dictionary was the appropriate data structure for storing each player's account details. If "Create Account" is selected then the program automatically creates a new player class and declares 3 attributes. The user can then update those attributes via input values for name, password, and deposit. Once the attribute values have been captured the class values are returned to the app page as a list where the list values are extracted and added to a dictionary. A dictionary was chosen because of the key value and the ability to leverage the key value as a joining value for future dictionaries.

Existing Player Log In

I created a function for log ins that captures temporary input variables from the user for username and user password. These temporary inputs are then compared to the accounts dictionary and responses are provided back to the user depending on if the temporary variables match the registered account details. If the temporary variables match an account dictionary key and associated password value then the user is welcomed to the game and redirected to the gameplay screen where he/she's details are extracted from the accounts dictionary and displayed. If the temporary variables do not match an account dictionary key or associated password value then the user is informed and redirected to homepage.

Game Play- User Interface

I first designed the layout of the page so the game play options were clear and easy to understand. The roulette board needed to be displayed in a way that both resembled a tradition board and also clearly displayed what values were available to submit a valid bet. I also wanted the terminal screen to be cleared continually throughout game interaction to dispose of the irrelevant historical data. I researched how to do this online and leveraged the Python library OS.

Game Play- Bet

I focused first on collecting multiple bets from the end user and storing these in an easily referenceable data structure. Again, I decided to leverage a dictionary data structure with a key value of board value and an associated value of bet amount. This way, once the wheel is spun and a random board value is selected, the spin value(s) could be compared with each betting key value. Then any key values that match the spin values would have their payout stakes added to the end user's balance.

Anyone who has played Roulette knows that a user can bet on more options than just the number. Because of this I needed to create a temporary list that collected the actual spin value and also all of the other associated values based on the spin value (Red, Black, Even, Odd, 1st12, 2nd12, etc.). To achieve this, I created global lists for each betting option and added each betting options associated board values. This way when the spin value is generated, conditional statements can be used to compare the spin value with each global list and add the additional associated board values to the temporary spin list. The logic for this was by far the most difficult to outline, understand, and code.

Game Play – Spin

I used a previous global list that contained all of the board value numbers and leveraged a random function to randomly select a board value. As mentioned above, once the board value is selected, the additional associated values are added to the temporary spin list so it can be compared to the user's bets.

Game Play – Clear

This option allows the user to clear the temporary betting dictionary and add the betting amount back to the user's balance. This essentially resets the game play and clears all bets that are on the board.

Game Play – Save and Log Out

Should the user select this option, the bet amount is added back to the balance and then the user's name and balance are returned. These returned values are used to update the accounts dictionary so the user's account details are up-to-date should they wish to log in and play again.

Game Play – Deposit

During game play the user can add funds to their balance using this selectable option. Once selected, the user is able to input a temporary deposit amount that is added to the end user's balance.

Game Play – Cash out

At any time, the user can select this option and cash out their current game balance. This will first add any live betting amounts back to the user's balance, then display a message to the user informing them of the amount that is being cashed out, and lastly the user's account balance is wiped to 0 and returned so the accounts dictionary can be updated.

Conclusions

This project was fun and one that taught me quite a bit over the last 5 weeks. The workshop assignments are valuable however, they usually include instructions on how to solve a problem or accomplish a desired behavior. Creating an application from scratch and being in a position where the logic is not 100% sorted allowed me to truly test out my knowledge of programming and Python.

While I am pleased with the outcome, I do believe there are cleaner ways to achieve some of the game play functionality. I am fearful the code is difficult to read and this makes editing it in the future a bit difficult.

If I had more time, I would've revisited each function and section of the application…further iterated, revised, consolidated, and simplified the code. I would also like to incorporate color so the UI is more attractive and transparent to the end user.