Project Showdown: Reflection Document

Lucas Brossi Barbosa

July 5th, 2020

Project Description

Showdown project consists in a 2-player version of Blackjack, where you play against the dealer (computer).

Game Rules

The project follows the standard game rules as described in the link below. I strived to implement a full version of the game with all the playing options like: insurance, surrender, splits (up to 4 hands), and double-down.

Blackjack:
https://www.pagat.com/banking/blackjack.html

Instructions

The code guides the user in a very didactic and intuitive way during the game. At most of user prompts, there are only few options the user can select (usually by pressing only a key). If the user selects an unavailable option, the code alerts the user and prompts it do it again. It was built to be user-input error prone, and intuitive to use.

Reflections

I ended-up implementing only Blackjack, instead of a suit of three different card games. That's because Blackjack only took me a long time to implement and consumed the limit of 750 lines of code!

My project ended-up having five classes:

- PlayingCard: as designed;
- CardDeck: as designed;
- CardHand: new class, intended to represent a hand of cards in the game – either the player's hand (original hand or any of the splits), or the dealer's hand;
- GameRound: new class, intended to represent a round in the game;
- GameAction: my old class user_interface, which represents a full game (with many rounds) and does all the interface with the user.

GameRound and GameAction took most of my time to complete. They represent together more than 600 lines of code! I had not anticipated how many different paths a Blackjack game could have, and how trick it is to code all of the different paths in terms of methods and user_scripts.

Other issue I faced during the project was that I was not able to print the suit characters of the cards in my terminal due to an encoding incompatibility. It worked well on Jupyter Notebook but not in the terminal. After some research, I realized that the default encoding standard of Python for Windows it is not the utf-8 standard, so some unicode characters are not recognized by the python interpreter. To solve that, I implemented these four lines of code I found on Stack in the beginning of my file, and since then everything worked well:

```
import sys
import io
sys.stdout = io.TextIOWrapper(sys.stdout.detach(), encoding = 'utf-8')
sys.stderr = io.TextIOWrapper(sys.stderr.detach(), encoding = 'utf-8')
```

It took me a lot of testing to feel comfortable that the code was properly working. As the game has many different paths, I had to run a lot of testing. After fixing many errors and inconsistencies, I decided that I would consider the game 'certified' if I was able to run 15 different games of 20 rounds each with absolutely no error and unintended behavior.

Just finished the tests now, and it seems to be behaving well. Let's see if it works in the class as well!

Finally, I would like to say that this project assignment was amazing for me. It was a lot of work, for sure, but the feeling of accomplishment and learning at the end is fantastic! I feel so much proud for my code (my very first one in Python!) and I have a deep gratitude for all I've learned in this past 2 months. Really amazing! Thanks for all the learning and support!