Index

For my course project, I made a functional Discord bot that users can interact with that is both fun and intuitive. According to Wikipedia, “Discord is a VoIP, instant messaging and digital distribution platform designed for creating communities.” I decided to integrate Discord into the project because Discord is very IDE / Developer friendly and is an application I use every day.

The type of bot I chose to build is a “command bot” that when a user sends a message leading with an alpha-numeric character executes some sort of command. The bot first needs to be added to a Discord server, and then given permission to send messages, view message / user data, etc. The bot is then linked to python code / IDE through a “Token” this token is provided when the bot’s account is created on Discord’s developer website.

I decided to go with is a Blackjack bot that plays the role of the dealer. Blackjack is a card game mostly played in casinos. The goal of the game is to get as close to “21” as possible without going over, and then competing with the dealer to see who can get closer to “21”. The player closer to 21 wins. My The rules of the game made building the bot challenging. For example, if the user is dealt an Ace, the user decides whether to have the card add one or eleven to their total score.

One of the main methods I used in this project is “import random” in the function “cardout” (line 25) as “random.choice”. I created a list of each unique playing card and this function randomly chooses one. The random method is also used in the !buy command, explained later.

As the game progresses, data is stored in filetype “.json”; essentially a text file with each unique discord user’s information. There are two JSON files: one for the user’s chip count and one for the user's card count. At the end of each hand the user's card count returns to zero via overwrite. The user's chip count information will only be overwritten to increase or decrease by 50 if they win or lose.

The bot executes different commands based off what a user sends in the chat box. Each user has a unique ID that the bot registers for storing information. If !start is sent, the bot will initiate a game by dealing the user two cards, find the sum of those two cards and present the user with their total card count. If the user is dealt a total that equals 21, they will win, and the bot will add 50 chips to their "wallet".

If the user's card count is less than 21 after !start is used, they have two options: !hit or !check. !hit will deal the user another card, and increase their card count. If the user breaks 21, they lose the game and chips. If they do not, they will have the same options again. !check will keep their card count the same, and end their turn. The "dealer", an integer variable, will be dealt random cards and either win or lose against the user. The dealer does not win chips, but chips are deducted from the user.

The final two commands are !buy and !bal. !buy 'deposits' and random number of chips between integers 0 – 100 into the user's wallet. The JSON file is then overwritten to increase their balance by that random amount. !bal will display the user's chip count, reading it from the JSON file in a friendly text format.

It is a long process to get my code running.

If you do not want to follow the entire below process, a video of example runs was uploaded to Blackboard under "Term Project Video".

DISCLAIMER: All embedded links are to websites with instruction, not forced downloads.

1. Discord must be installed by the user, and the user must have ownership over a server. Instructions on how to do that are here: Join Discord / Create a Server.
2. The user must import and install discord.py, which can be done here: Install discord.py
3. The user must import and install jsonlib.py, which can be done here: Install jsonlib.py
4. The user must import and install python-dotenv.py, which can be done here: Install python-dotenv.py
5. You must save down the .env & .json files included in the .zip in the same folder as your code before running the script.
6. You can now run the code.

Code minimum inclusion requirements:

(1) at least one of the container types (list, tuple, set, or dictionary)
- Lines 25, 227

(2) at least one iteration type (for, while)
- Line 224

(3) at least one conditional
- Majority of code is conditionals.

(4) at least one user-defined function
- All "async def" are user functions

(5) at least one user-defined class (could be very simple)
- Line 24

(6) file input or output
- Lines 332 – 334
- All "await ctx.send" lines are output to send messages in Discord

Final Note: I would appreciate any feedback on cleaning up / improving my code! I have no prior python experience and am keen to learn how to make my code cleaner.