How to Use Menu Builder

The Menu Builder is designed to be run from the command line. Note that in order to use the Menu Builder, you need to download the four raw data files (the Cocktail, Appetizer, MainCourse, and Dessert csv's) and place them in the same directory as menu.py. Please ensure that you have already installed Pandas, as the program imports the four csv files as data frames.

Items I Would Finish Later

I completed everything I set out to complete, but have identified a few additional items that, if given the time, I would undertake:

- Additional user input checking - while I have included fairly robust user input checking, I would go a step further and look for additional ways of checking for errors. Specifically, if a user enters a single letter when typing in the primary liquor for a cocktail, like "g", the program interprets that as "gin" because of the df.str.contains() method I deploy. I used this method because some cells in the data frames have multiple values. In the future, I would likely split the string value in the cell into a list to avoid this behavior.
- Skipping courses - at the end of this project, I considered allowing a user to skip a course altogether. Not everybody wants a cocktail, appetizer, and / or dessert for every meal.
- Quitting at any time - the user currently has an option to quit upfront, but otherwise needs to complete the flow of the program (unless force quitting). In the future, I would incorporate allowing the user to quit whenever he / she would like.

Challenges

The biggest challenge I faced was deciding where to build the user input methods (i.e., within the class to which they pertain vs. a separate class to get user input). I ultimately determined that those methods belong within the class to which they pertain because the prompts and the input returned are not consistent across classes. Additionally, keeping them separate would violate the best practice of encapsulation.

I also faced challenges with formatting and error checking against data within the data frames. Working with strings within data frames can be tricky, as can accessing specific locations based on user input. It was helpful for me to comment out blocks of code and iteratively test different combinations of methods on Boolean-filtered data frames (i.e., cocktail = cocktail_df[cocktail_df["mood_pairing"].str.contains(mood_choice)].sample()["name"].to_string(index=False).lstrip()).

Ultimately, I learned a great deal about object oriented programming with this project and will use my program frequently!