Michael Sanchez

CPS 3320: Python Programming

Robert Domanski

2/26/20

Project Assignment #1 Write Up

**Background (Summary of Original):**

For Project Assignment #1, I created a Sandwich recommendation program. The original goal was to come up with a program that would be able to take a user’s preferred sandwich and make changes to create a similar sandwich based on the user’s likes/dislikes and compatibilities between food items. The Data used included various lists/dictionaries of food items (from bread to primary items (Meats, Cheese, Spreads) and secondary (Condiments, veggies, fruit), likes/dislikes food items, and compatibilities rankings(1-3) between items. The Analysis viewed using a text file to keep/store the data, a general gist of the interaction between user and program, and output being the printed list of sandwich recommendations alongside showing that results followed the compatibilities & like/dislike system. There were many potential roadblocks such as Lack of data, possible interactions/inputs from the user to the program, and how comparisons & putting items together (such as two primaries) would work.

**Program Complete Description:**

Side note: If looking alongside the program python code, This Description will follow right along the python code itself.

Imports: The program makes use of a few imports. Json for writing/reading data through loads/dumps. OS path to create a txt file to store the data. Time to use the sleep function to create a sense the program needs to load and “think” about decisions. Random for the random choice function when picking items for a sandwich.

Set-up: (At the time of this write-up) Lines 18-19: print(“Checking for Data!”); time. Sleep(1), are one of many lines of code through out the program that keep the user “updated” with the current steps being taken place by the program and take advantage of the sleep functions to increase readability. When Setting up, The program uses an IF- ELSE function that checks if the default list text file exists. IF not found, then the hard-coded data within the program is saved into a new created text file, followed by the text file being loaded back into the program. ELSE the program has found the data and loads the data into the program. Once that data is loaded as a list variable, it is separate once more into various dictionary variables, that will be called by the functions.

Functions: *change Username* function asks if the users wants to change the username, if yes then the new username is saved to the text file. *change LD* function allows the user to review through a list of the food items contained within the data and set/place them into one of three list (Like/Neutral/Dislike*). Select Next Item*, *select Bread* and *msw* (Make Sandwich) functions all work together to create a sandwich. *select next item*, creates a list of items based off the compatibility rankings (mid once, 1/3proability; High twice, 2/3 probability) and selects the next item as long as it is not contained within the dislike list. *Select bread* only checks the dislikes when choosing an item from the bread list. For both the *select* functions, if all items were disliked, it would be treated as random (with compatibility still in use) instead. The *msw* (make sandwich) functions makes use of the *select* functions to “prepare” a sandwich for the user while walking them through the steps using the lines 18-19 method of communication and once the sandwich is done, the function prints out the result (result is not saved). *Wild card* function creates a random sandwich by mixing all items into one list and select each item without regard to the likes/dislikes or compatibility. *CreateSW* function serves as a menu for the overall process of creating a sandwich and asks the user what primary category they would like (Meats, Cheese, Spread, Wild Card). The Main Menu function is the menu which after every action/function will eventually return(through recursion), from the main menu users can [1] get a sandwich recommendation(calls *createSW* which eventually calls the other sandwich-related functions), [2]Like/Dislike setup(calls *changeLD*), [3] change username(calls *changeUsername*) , and [4]quit which uses the quit function to exit the program.

Main Process: It contains a list of greetings, with one being chosen as a random greeting (to be used later). The program checks if the username is default, if so it runs through the *changeUsername* function else it proceeds to the greeting which prints the random greeting alongside the username, such as “Hello Mike”. After this print, it proceeds by calling the *MainMenu* function which can continue if the user wants, through recursion, and only closes the program till the user inputs the value to quit.

**Results:**

My original hypothesis included making a list of sandwiches (which eventually changed to only presenting one sandwich at a time and “The Program should also be able to make ‘rational’ sandwich-related decisions based on compatibility and Likes/Dislikes.”. Overall I would say the Program can make ‘rational’ sandwich making decisions based on the two systems (even if it’s simple). I could see it expanding if I created a function/process to compare adding an item to sandwich to at least the first item of the list. The most challenging part of the entire project was syntax and indentations, especially when it came to the hard-coded data which was a list of dictionaries that also contained dictionaries and lists. I had to make serval changes which included: Giving recommendations based on type/primary categories (Meats, Cheese, Spread) rather than a more complex string/options such as “Turkey Cheddar Lettuce On Bagel” and Giving one recommendation at a time vs the original intention of a list of 3-5 sandwiches and Different overall presentation of the program. If I could do it again, I would change most of the set-up code into one function, which is still possible but requires some extra work, this would probably make it easier to create error handling when it comes to faulty data within the file or if data is deleted from it.