Iteration v2.5

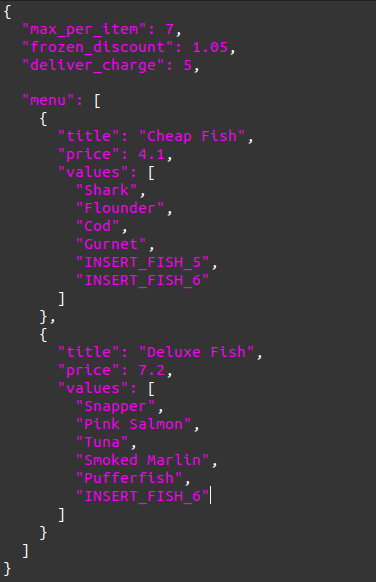
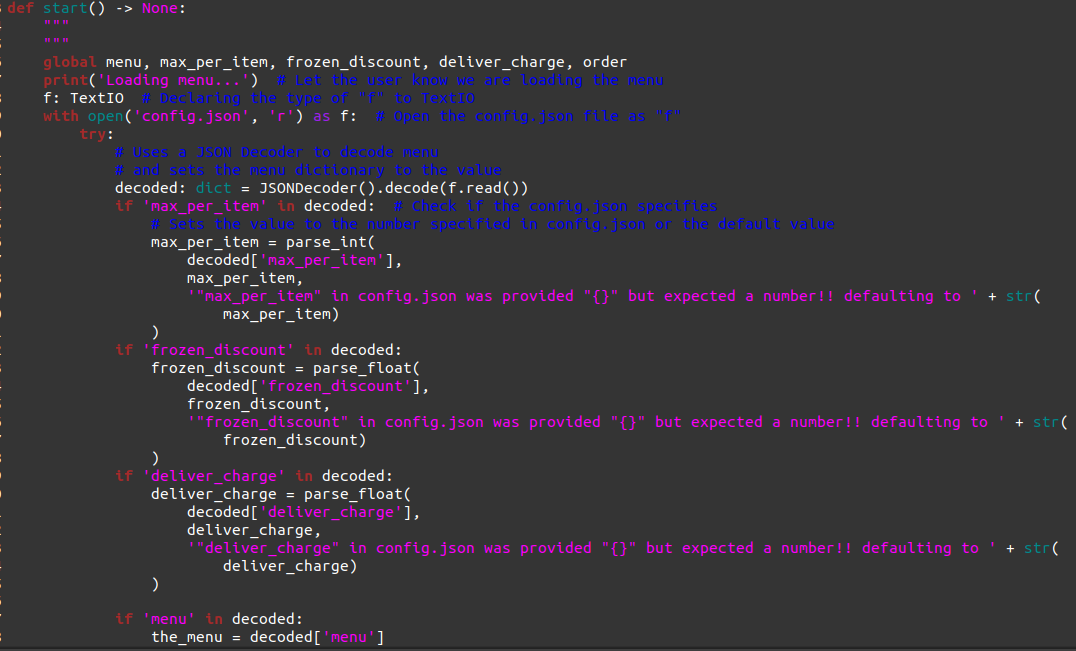
*Freddy’s Fast Food*

*Information*

My ordering system went through many changes during development and it had a few major changes but most changes were small tweaks to the way things were displayed or optimizations to how I was providing, validating or storing data

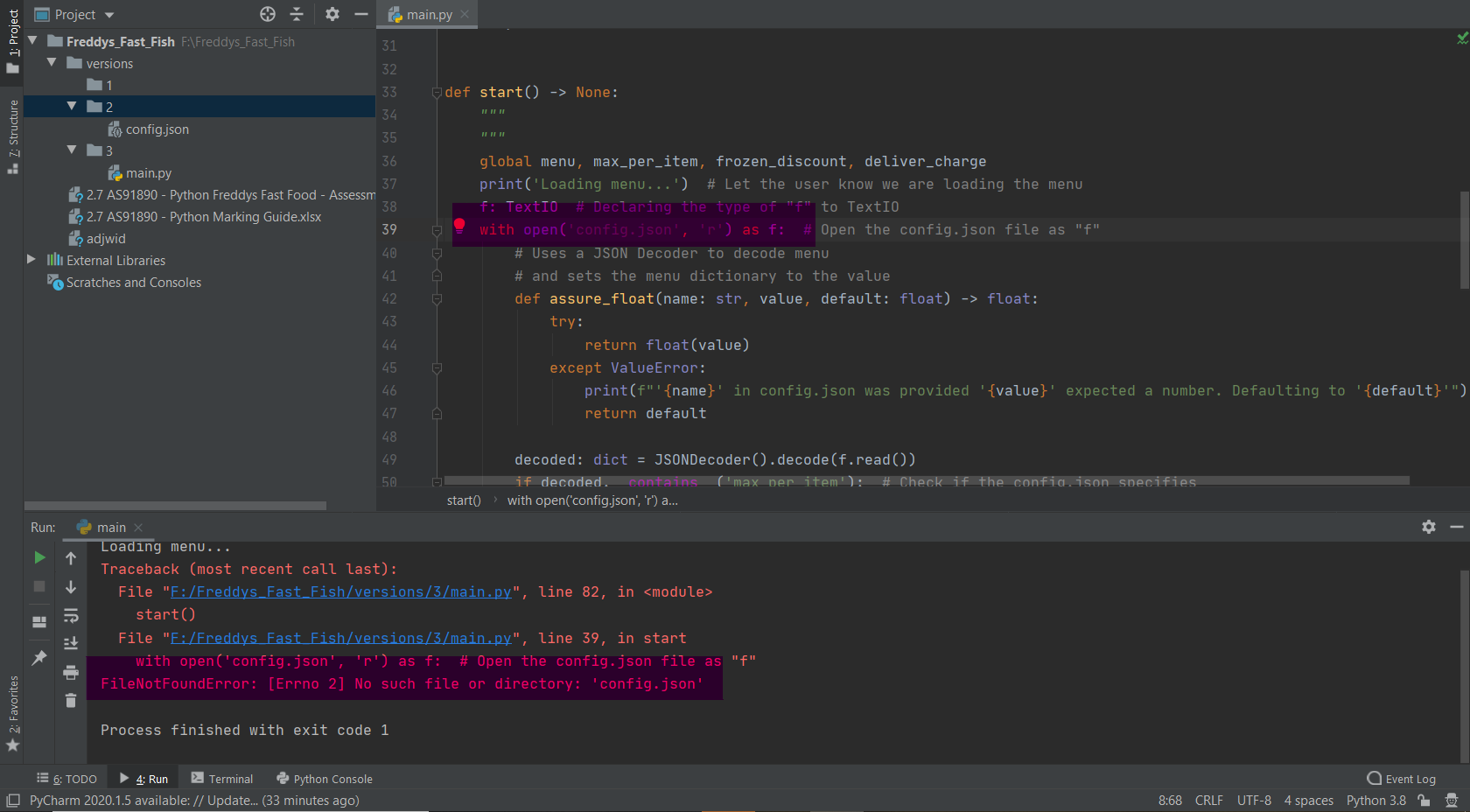
*Storage of the menu*

My first major idea was having my menu and variables just as discounts and delivery charges stored in a separate JSON file that was loaded and parsed when the program was launched



These photos show the old menu code which read from config.json

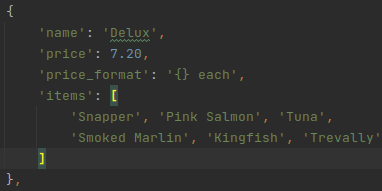
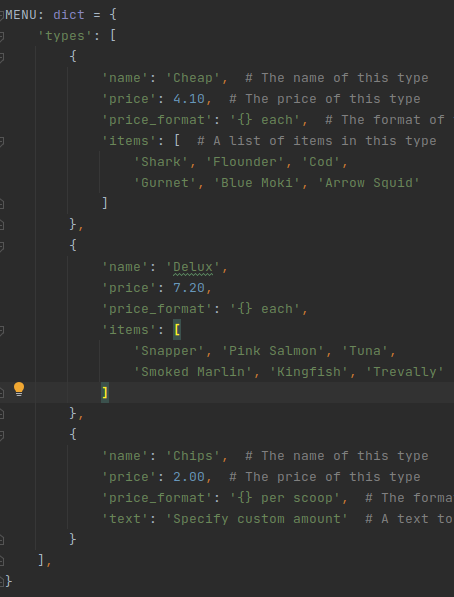
Although this design would seem to make it easier I found out that by having my menu system inside of a JSON file I bring many issues into my program such as the possibility of malformed JSON, missing files, or incorrect formatting which would require a large amount of code to ensure that would not happen. So after a few changes I decided to store the menu inside of the main.py file inside of a dictionary

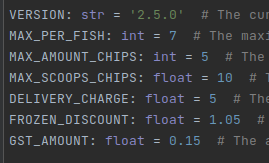


An example of a problem that could occur by using JSON stored in another file

This alternative way of designing my menu removed the issues that came along with JSON it also allowed my editor to better analyze the code that I was writing and allowed me to write faster without having to check back and forth between the JSON

My variables are also declared in constant format at the top of the file above MENU

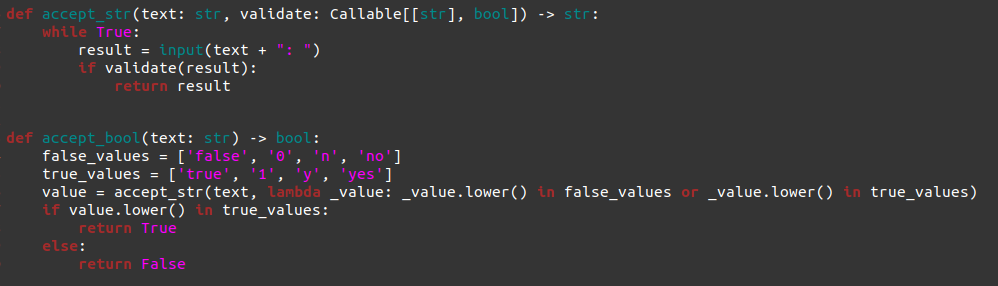




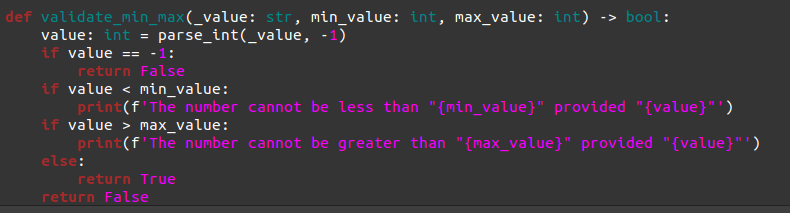
These photos are of the new dictionary based menu system

*Validation of input*

When designing my way of accepting user input in my first version of the ordering system I came up with the basic idea of having a accept function which takes in a message and a callback function that determines whether or not the user input is allowed



This photo shows the original accept function which is used to take in user input

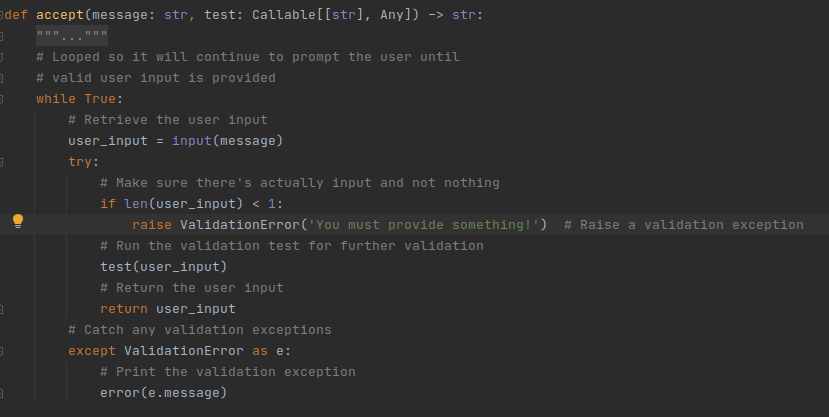


This photo shows the original min\_max validator which was used to validate the user input

However this way of validating input is quite cumbersome and didn't handle multiple types of error messages or accepting multiple types without large or bulky amounts of validation code

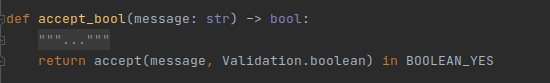
So in my second version I decided to scrap this system instead I decided to go with an Exception based system so that I can easily do multiple layers of validation or accept different types of data with very little code

The following code is used for accepting string user input this is also wrapped by the accept\_bool, accept\_int, and accept\_float functions



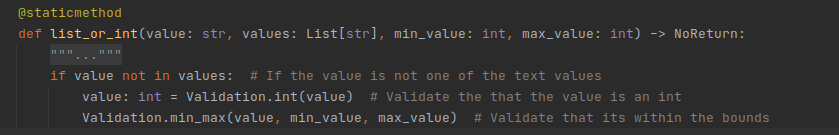
This photo shows the accept function which is used to take in user input

The following screenshot shows the function for accepting boolean user input from the user using the new exception based system



This photo shows the accept\_bool function which is used to take in user input as a boolean

The following function showcases the ability to easily chain these validation calls onto each other to create easy and short validations

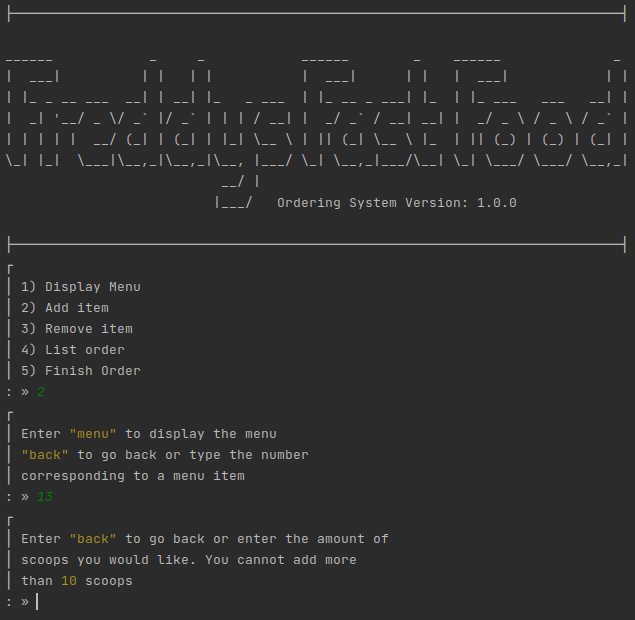


This photo shows the list\_or\_int validation function which is used to validate user input

This is a major improvement compared to the original system and I think it was a worthwhile change

*Menu descriptions*

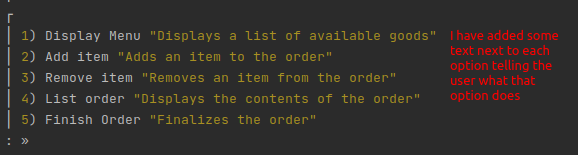
My original menu prompt system lacked information on what each section did and may have been hard for users to understand the meaning of each screen



This photo shows the original layout of the menu in version 1

My solution to this problem was to add little descriptions onto the end of each of the

Menu names I also added ASCII colour formatting however this change is only visible to users using linux operating systems as ASCII is not supported in most windows terminals

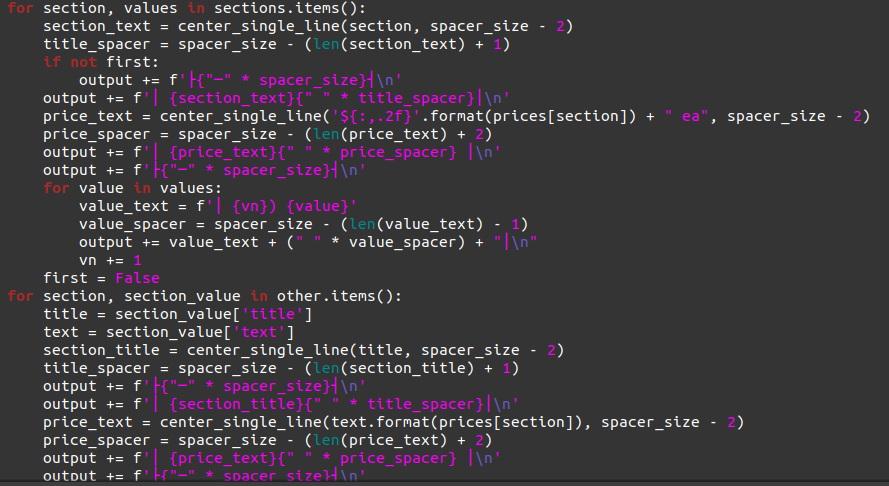


This photo shows the new layout of the menu in version 2

The quoted descriptions help give context to what each menu is making it easier for first timers to navigate through and the bright colours help draw your eyes towards the descriptions and the numbers that correspond to them

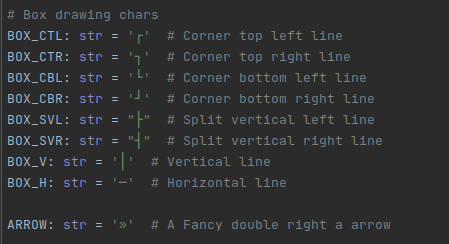
*Box drawing*

My original menus and prompts were all drawn with the box drawing characters inline like so I felt like this was a bad design and it made the code less reusable and involved a lot of copy pasting characters



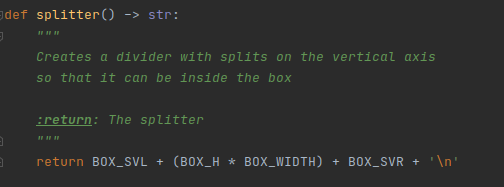
This photo shows the original code for generating the menu

So I decided to move these out of the code and declare them as variables this allows me to use the box drawing chars outside of the function easily without having to copy paste all the different corners and sides



This photo shows the variables for box characters

I also split up commonly used shapes or formats in the menu into small functions that do it automatically which decreases the amount of code required / repeated over and over again making the menu more dynamic and less copy paste

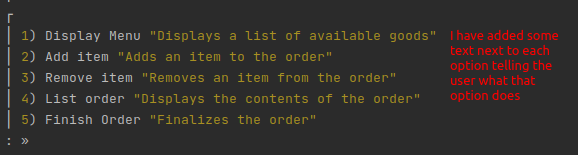


This photo shows one of the small functions that creates a part of a box

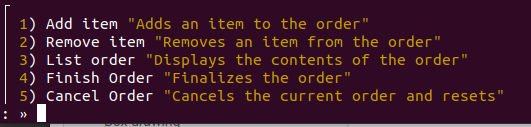
*Shortened Main Menu*

I was told that my original program was slow/confusing I followed up on this by asking a peer of mine for advice and they suggested that I remove the first option from the main menu “Display menu” because this feature is already available when adding an item

Another suggestion of theirs was to always print out the menu every time a new item was added or removed however I declined this suggestion as doing so would hide any errors of information provided to the user after adding an item due to the size of the menu



This photo shows the original menu



This photo shows the new menu after removing the item