CODELAB I

ASSESSMENT 2: Utility App

Module Coordinator: Ms. Lavanya Mohan

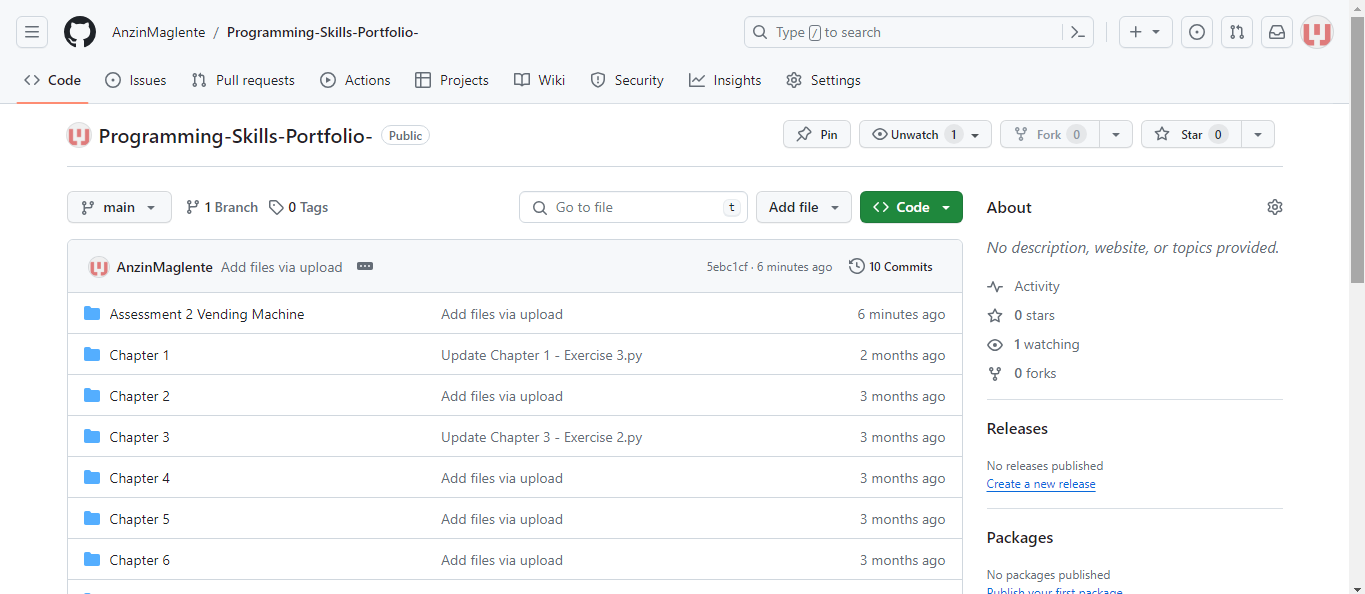
Marker: Ms.Lavanya Mohan

| Contribution towards overall module mark | 60% |
| --- | --- |
| Date set | November 7, 2023 |
| Marked work returned by | February 5, 2024 |
| ASSESSMENT DEADLINE | January 10, 2024 |

Programming Fundamentals

| **Student’s Name:** | Anzin R. Maglente |
| --- | --- |
| **Id. No:** | 2023\_411 |
| **Github Repository Name:** | Programming-Skils-Portfolio |
| **Github Repository Link:** | <https://github.com/AnzinMaglente/Programming-Skills-Portfolio-> |

**Repository Screen Shot:**



ASSESSMENT 1&2

| Contribution towards overall module mark | 100% (40%+60%) |
| --- | --- |
| Date set | September 5, 2022, |
| Marked work returned by | Within 3 weeks of submission |
| DEADLINES | Deadline : Jan 10, 2024 – 23:59 |

**BRIEF**



In this document, the student is tasked with making a functional vending machine that demonstrates their expertise in programming and makes use of the several lessons and techniques learned throughout the module. It will have several features which include selecting the item the user wants to get, a way to manage their money, how much change has been given, etc. It could also include additional features for additional points.

Link: <https://github.com/AnzinMaglente/Programming-Skills-Portfolio->

**Specifications (100 words)**



Products:

| Product ID | Item | Price | Amount | Type |
| --- | --- | --- | --- | --- |
| C1 | Doritos Nacho Cheese | 2.5 dhs | 5 | Chips |
| C2 | Lays Classic | 2.5 dhs | 5 | Chips |
| C3 | Lays French Cheese | 2.25 dhs | 5 | Chips |
| C4 | Flaming Hot Cheetos | 1.5 dhs | 4 | Chips |
| Ch1 | M&M | 2.0 dhs | 3 | Chocolate |
| Ch2 | Kitkat | 3.25 dhs | 3 | Chocolate |
| Ch3 | Hershey | 4.25 dhs | 3 | Chocolate |
| P1 | Chocolate Chip Cookie | 2.5 dhs | 3 | Pastries |
| P2 | Biscuit | 1.5 dhs | 3 | Pastries |
| P3 | Cracker | 3.0 dhs | 3 | Pastries |
| D1 | Water | 1.0 dhs | 6 | Drinks |
| D2 | Milk | 1.25 dhs | 5 | Drinks |
| D3 | Hot Coffee | 3.25 dhs | 4 | Drinks |
| SD1 | Coke | 2.25 dhs | 3 | Soft Drinks |
| SD2 | Sprite | 2.25 dhs | 3 | Soft Drinks |
| SD3 | Miranda | 2.25 dhs | 3 | Soft Drinks |
| J1 | Apple Juice | 1.25 dhs | 3 | Juice |
| J2 | Orange Juice | 1.25 dhs | 3 | Juice |
| J3 | Mixed Berries Juice | 1.25 dhs | 3 | Juice |

Diagram 1: Product list

Description:

The diagram above shows what products are available in the shopping cart, this also specifies how much the product is, the amount of stock each product has, its type, and its product ID. However, it is missing the aspects of how the bundles will work because that will be discussed on a later section.

| Features | Descriptions |
| --- | --- |
| Being able to select multiple products | The user can add multiple items to their shopping cart after ordering a product, or they might choose to finish their shopping. |
| Alternate Options for the  different choices | If the user accidentally makes a grammatical mistake when choosing the item, the system will attempt to find the correct item. Example: instead of inputting “C1”, you could also write “c1” to get the first chips option. |
| Categorizing the products  based on type | Products are separated by their type, this includes chips, chocolate, pastries, drinks, soft drinks, and juice |
| An intelligent system that asks the user if they want to apply for a bundle | The system will detect if there is a bundle associated with the current product the user is selecting, if there is, it will ask the user if they would want to apply for it. |
| A stock system | Each product has a limited amount of stock and cannot be brought infinitely, this is seen in diagram 1: Product list, under the column “Amount”. |
| A way to manage the user’s money | After selecting the products, the user is met with an exchange of value where they have to input a value equal to or greater than the total price they have accumulated. Through their shopping. |
| A way to get back the change after buying a product | If there is any leftover money, the vending machine will ask the user if they want it back or to donate it to charity. |
| The use of functions to make  the code easier to understand | Functions are used to simplify several processes that occur multiple times within the code, this is all stored in another file called “Definitions”. |
| The use of Db Browser for SQLite | SQL is used to store the data for the products as shown in “Diagram 1: Product list”, although some details (e.g. Bundle item and price) are omitted to keep it concise, they will be further discussed in the Technical Description section. |
| A way to see the user’s shopping cart | After selecting an order, users may choose to see what they currently have in their shopping cart |
| A way to restock the database | There is a file that I made to easily restock the database without having to go inside it. |
| A way to cancel an order  while ordering | If the user makes a mistake midway through the ordering process, they can cancel their order. |
| Looping music | Music will play in the background as the user goes through the process although this feature can only be heard in windows. |

Diagram 2: Features List

**System Flowchart (50 words)**



Flowchart

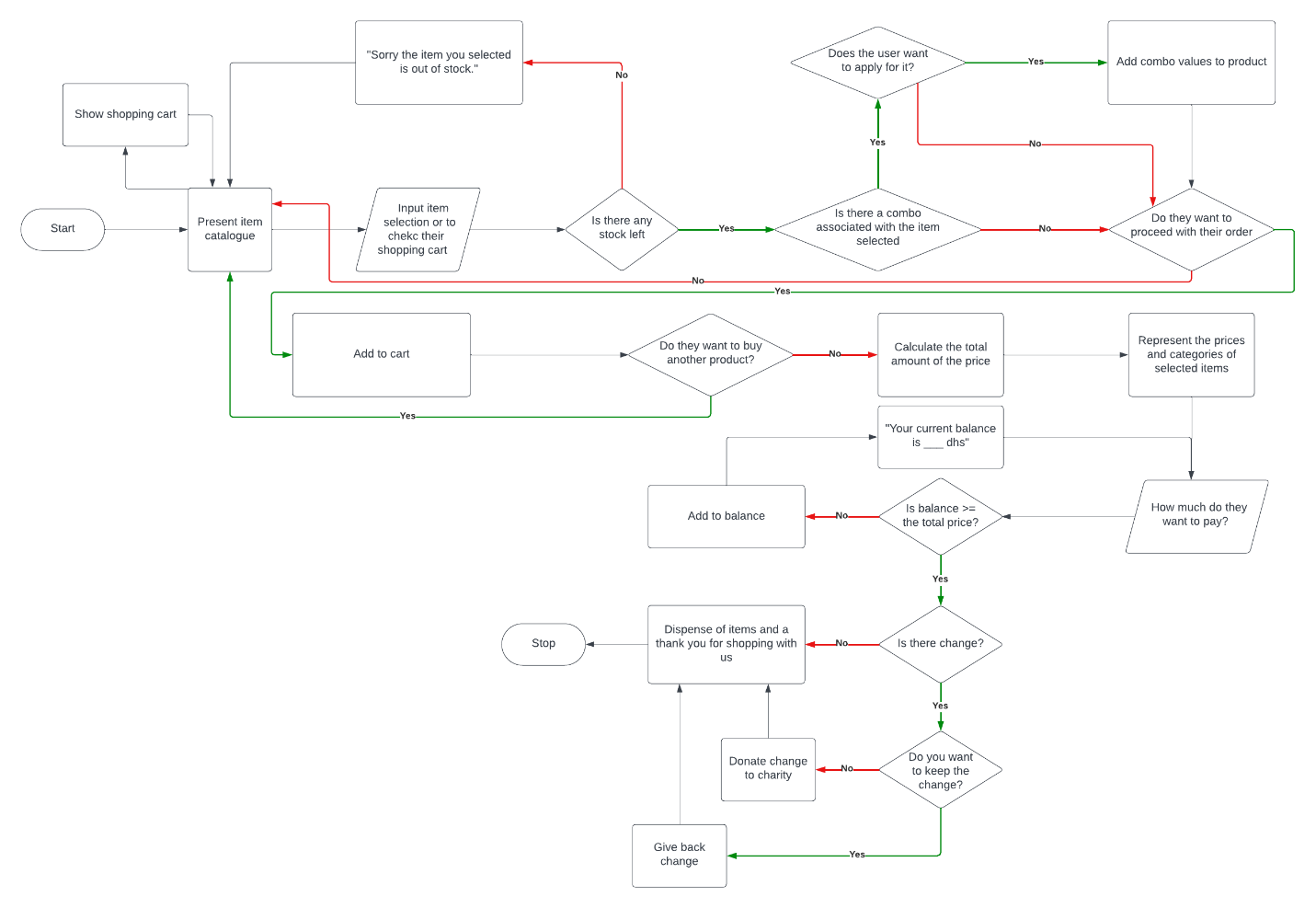


Diagram 3: Project Flowchart

Description:

This flowchart starts with the presentation of the items the user may want to buy, then the system will ask the user to input a product ID for them to buy any item, the system will check conditions like if there are any more stocks for that particular product, if there is a bundle associated with the selected item, and if there is it will ask the user if they want to apply for the bundle. After all, the user can choose to continue to the next part of the code or continue shopping for other products. When they finish shopping, the system will calculate the total price and the user has to input an equal or greater amount of money into the machine based on the total price of their selection. If not, the whole process repeats until that condition is true. Finally, it checks if there is any change, if not it dispenses the items, if there is, it will ask the user if they want to give it to charity or get it back, depending on their choice they can get their change back along with the items they bought

Link: <https://drive.google.com/file/d/1Id1nHqZA_Z_GR13so1kgTE9IS2irhmfS/view?usp=sharing>

**Technical Description (400 - 500 words)**



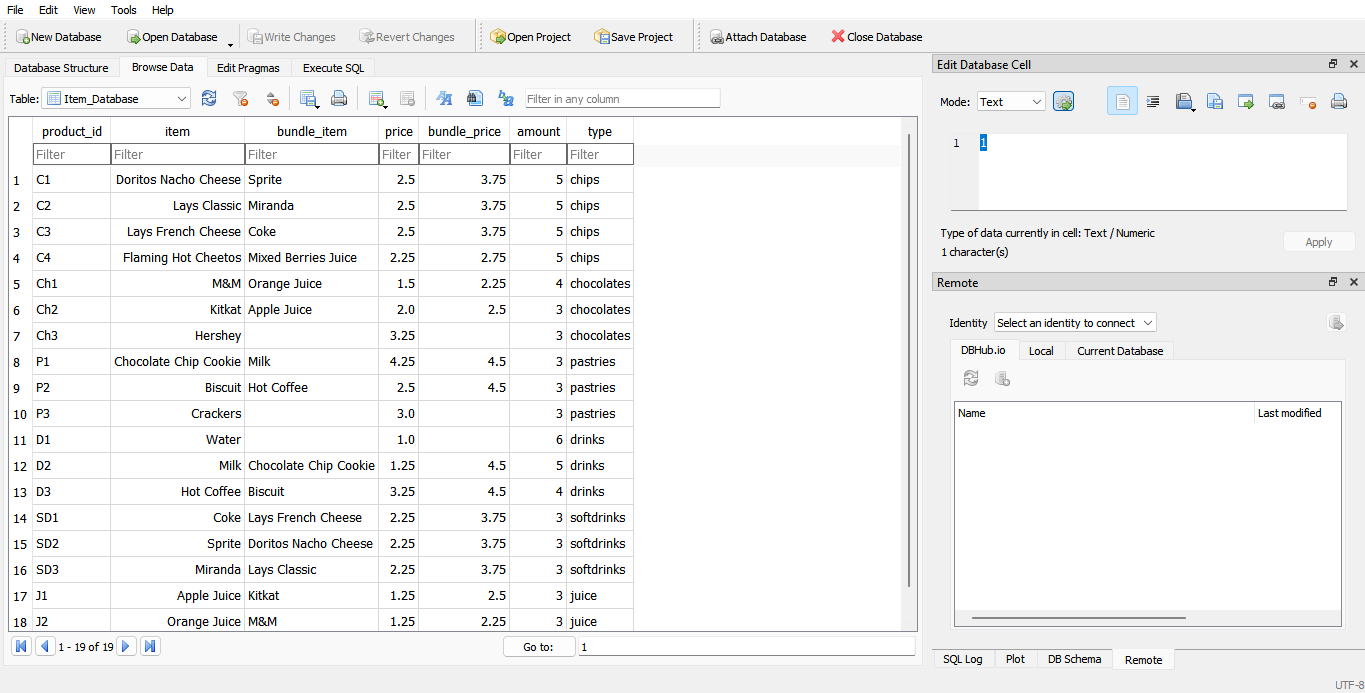
Table of Contents:

1. SQLite Database Overview and the restock function
2. Definitions of functions

* product\_list
* decrease\_amount
* increase\_amount
* bundle\_check
* bundle\_decision
* show\_product

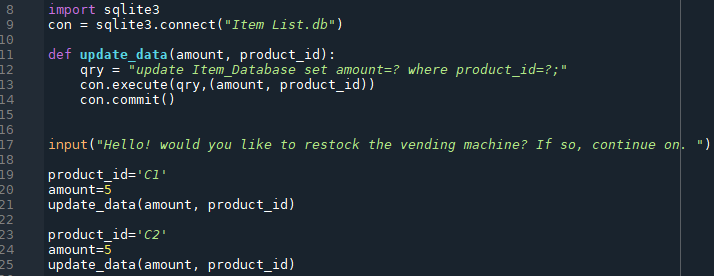
1. Main Code

SQLite Database and restocking the database



Database:

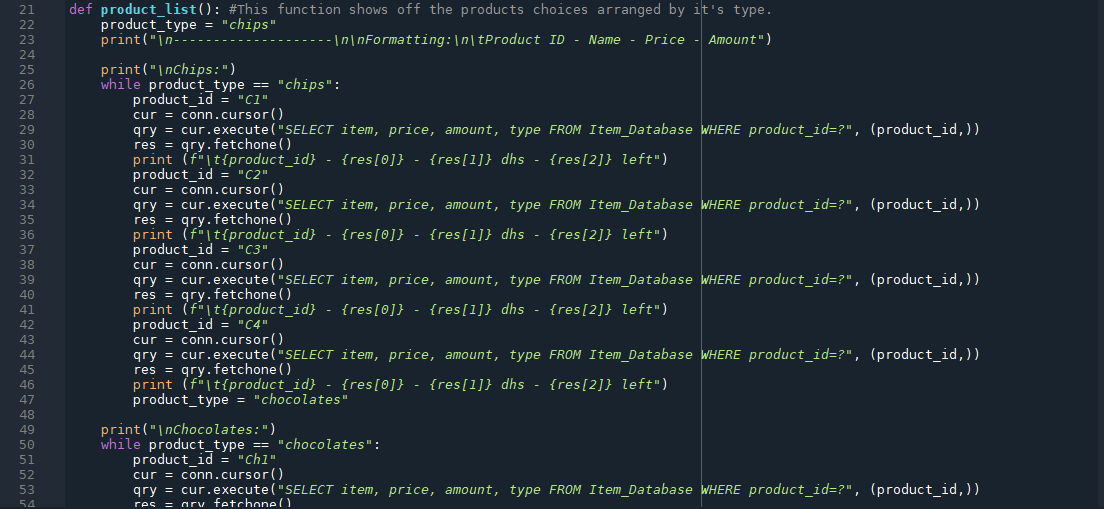
This shows my database where all my values are stored, this includes the product id, item name, bundle item, price, bundle price, amount, and type. One thing I want to note is that the amount changes each time the vending machine is used, which is why there is a separate file wherein it can restock the items automatically. If the product reaches zero then the program will tell the user that the item they order is unavailable as of right now.



Restocking the vending machine:

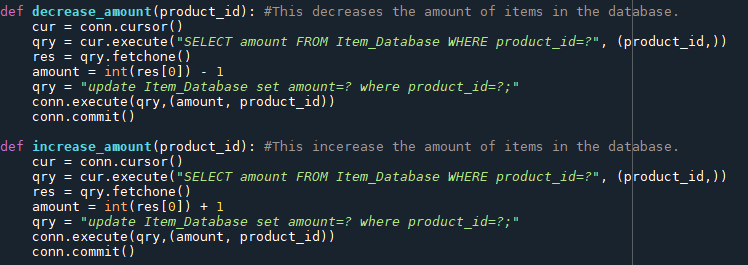
This shows how the vending machine updates their values to refill the products included. The variable “qry” is used to specify the conditions/parameters that allow the system to update the database which is then executed then by con.execute, the variable being updated is the amount while the way the code tries to find it is by it’s product id. con.commit() sends the action to the database

Definitions



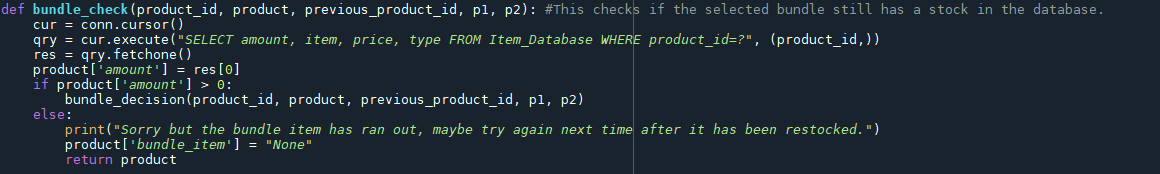
The Product List

This prints out the items, their price, type, current amount, and product id in an ordered manner. Using this the user is able to choose what they want, there is also a bundle section but that just shows what the bundles are, to access it yourself, you have to order one of the products included in the bundle (it’s 20% off) and the system will ask the user if they want to buy it. This also shows that the user can exit after buying any one item and can show the current shopping cart of the user. This is all done via the connection to the database, it gathers each item via the cur, qry and res variables this includes the price, amount, item name and type in the database to actively update the product’s amount each time the user uses the program to buy an item(s).



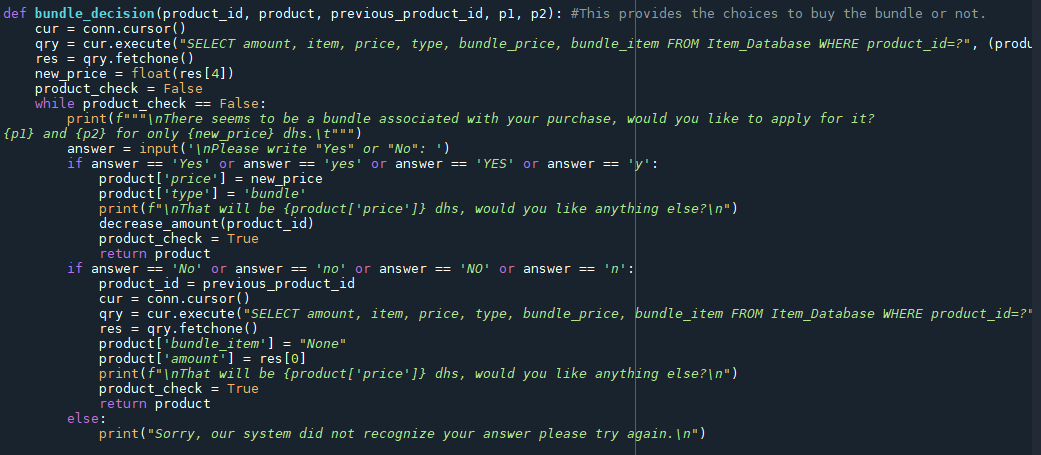
Increase/Decrease amount

This increases or decreases the amount of items from the database if it is being bought or if the order is being canceled. This again is done by the cur, qry and res variables. It selects an amount by checking the specified product\_id then removing one stock to the amount before updating it and committing that data to the database



Bundle Check

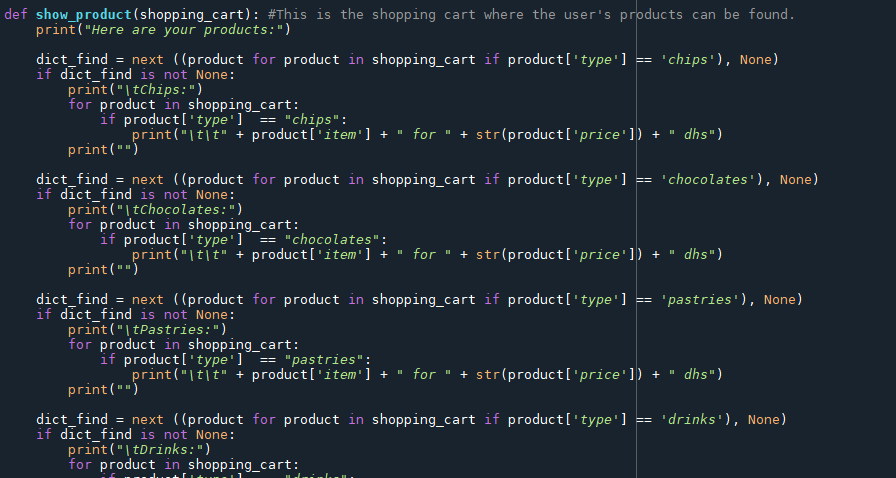
This uses the database variables to get the product amount and a if condition to check if the amount is greater than 0, if it is it will move on to the next function called “bundle\_decision” i, if not it will tell the user that the item has run out and changes the product bundle item to “None” before it uses the return command to return the variable named “product” to the main code.



Bundle Decision

This asks the user if they want to buy the bundle associated with their purchase, if they answer “yes”, the program will add it to the cart, if “no”, then the program will only add the selected product. This also affects the final price of each item. This uses a while loop in order to repeat the code if the user wrote something wrong, it uses the database variable to get the amount, item, price, type, bundle price, and bundle item. Creates a variable named “new\_price” to change it later to the bundle price. Then the program will ask the user if they want to apply for the bundle, if they answer yes then the product price and type changes and the amount of items for the bundle product is reduced by one in the database then it returns the product. If the user’s answer is “no”, then it will change the product id back to its original state and change some values of the product to its original values. before returning the product.

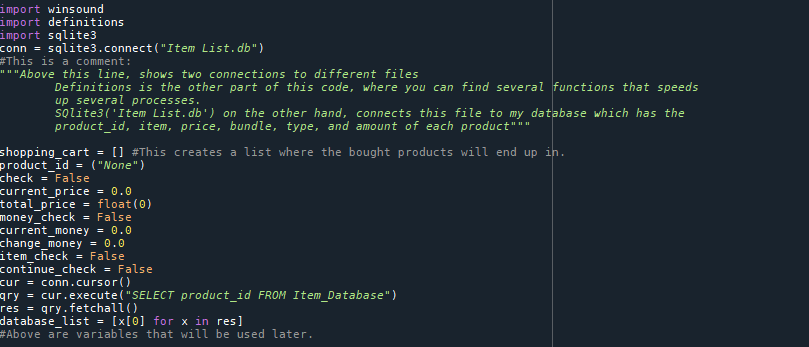
p1, p2, and new\_price is used to display the bundle.associated with the product.



Show Product

This shows off the products inside the shopping cart, using a command called next and some if conditions, it cycles through the whole shopping cart/list to find if the type of item is present in the list, if it is the type of item and the items in the shopping cart will be printed out, if not, then the if condition is skipped.

Code



Main Variables

These are my main variables:

*shopping\_cart* is where all the items of the user is stored using a list

*product\_id* is what helps the system detect what item is selected

*check* is used to repeat the process of buying a product using a while loop.

*current\_price* is used to keep track of the amount of money of the item being selected just in case there is a bundle associated with the item.

*total\_price* is the value of the items being currently selected.

*money\_check* is the while loop check for the transaction process to make sure the user inputs the right amount of money (being equal or greater than the total price).

*current\_money* is used to keep track of the amount of money the user has inputted in the transaction process.

*change\_money* is the amount of money leftover after buying the product(s).

*item\_check* checks if the user has bought an item in the vending machine in order to leave during the selection process.

*contunue\_check* checks if the user wants to continue shopping or move on to the transaction phase.

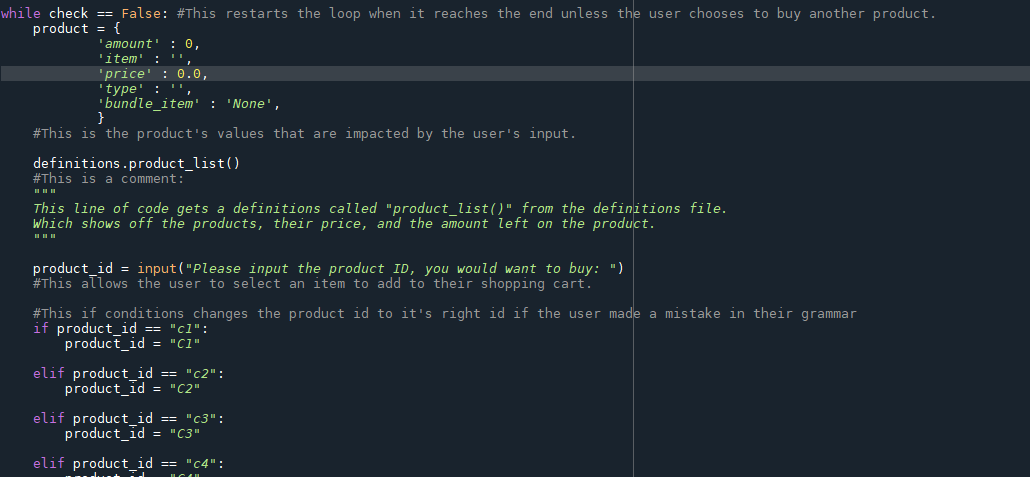
*cur* is the cursor that connects the database to the code.

*qry* is the action that the system will do to the database.

*res* is the selected items that the query got from the database.

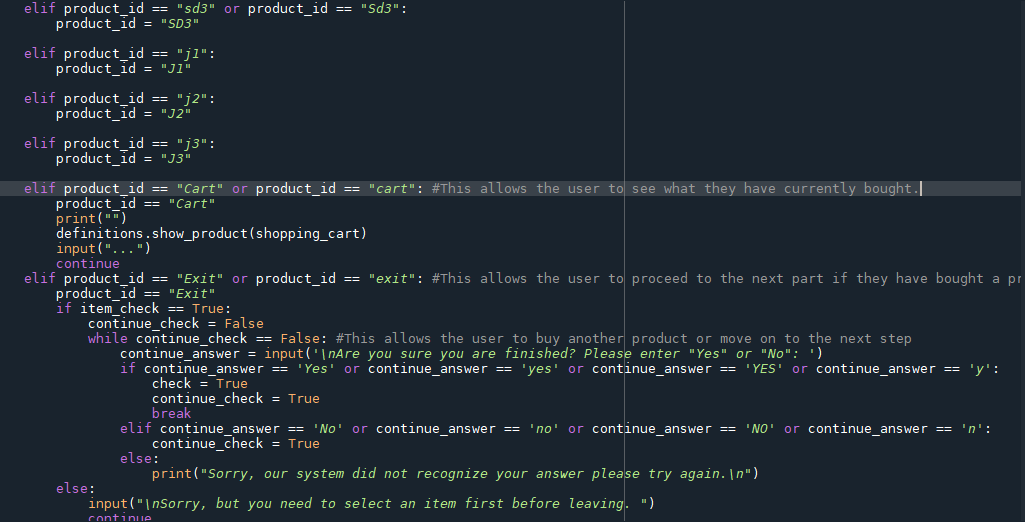
*database\_list* is the whole product ids of the database, this is used to check if the inputted product id can be found inside the database, if not, it will ask the user to try again.

product, although not shown in the picture above (it's in the picture under this). This keeps track of the selected product using a dictionary which will be placed into the shopping cart as soon as the user is done with their selection,



The start of the product selection process and the selection of product ID

This starts the product selection process by printing out the product list and asking the user what they might want from it. This also starts the ID corrections which will be explained in the next part.

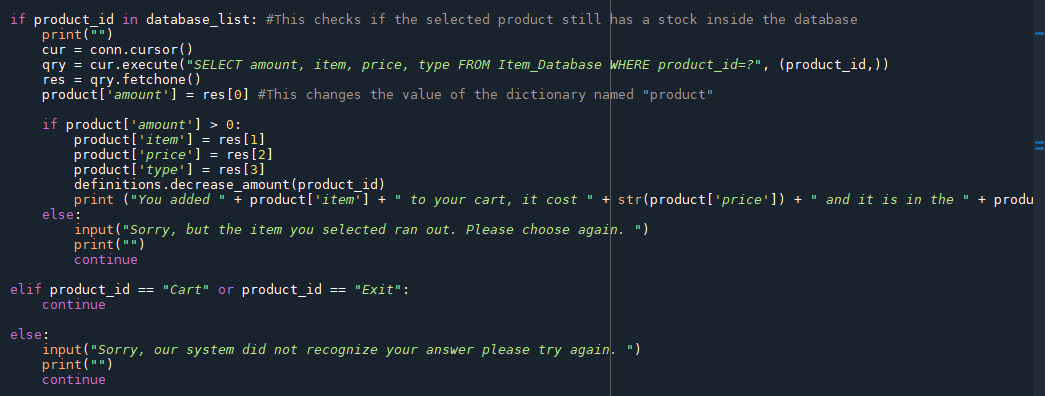


ID Correction, View Cart, and Exit

ID corrections basically correct the inputted ID if the user uses the wrong capitalizations in order for a much smoother process. This is done by simply changing the value of the product\_Id to it’s correct value using several if conditions

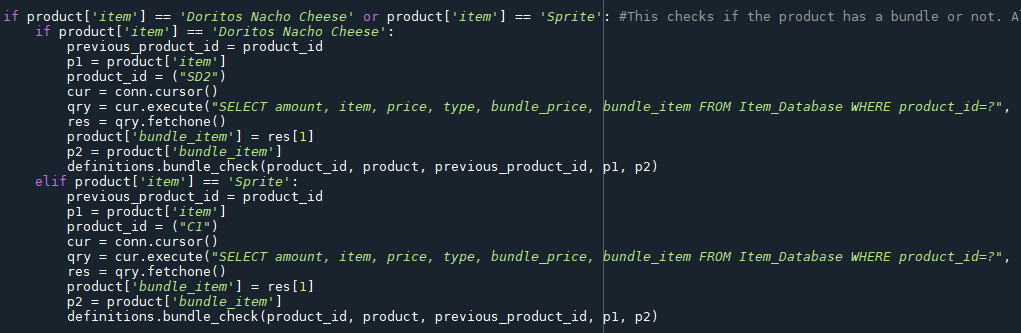
View cart is seen if the user writes down “Cart” or “cart” and it shows off the current products the user has already selected using the show\_product function inside the definition file.

Exit is seen if the user inputs “Exit” or “exit” and it allows the user to leave early if they have selected an item before. It will run a piece of code later on the program. Which is why I won’t be explaining it here.



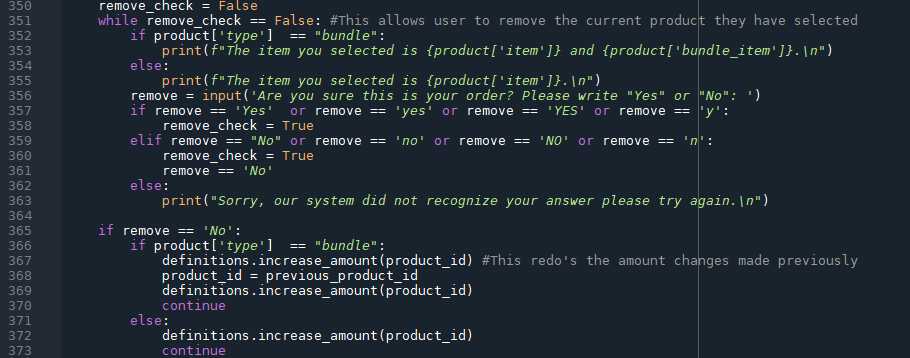
Product amount inspection, and adding first product to cart

This part checks if there is any more stock in the selected item using the database variables, if the condition returns false it will tell the user that it has run out and they need to choose another product. If there are, then it will change some values, particularly item, price and type inside the dictionary named product while decreasing its amount in the database. At the end it restates what the user has added, it’s price and their type.



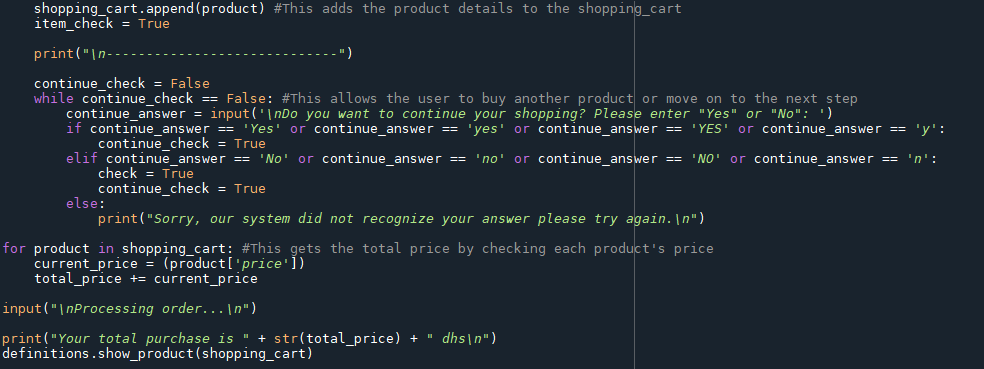
Bundle checking

This line of code is just a sample for the rest of this section, it checks if the selected item is part of a bundle if it is then it will check if the other item is still in stock, if there are, it will ask the user if they want to apply for it, if not then it will tell the user that the bundle item has been sold out. This is done by several if conditions checking the product ‘s item and then creating some new variables called previous\_product\_id, and p1 which will be used later, and changes the value of the product\_id to the respective bundle item, and adds it to the product bundle item using the database variables. It then moves on to the bundle\_check function in the definitions file.



Current item(s) selected checking

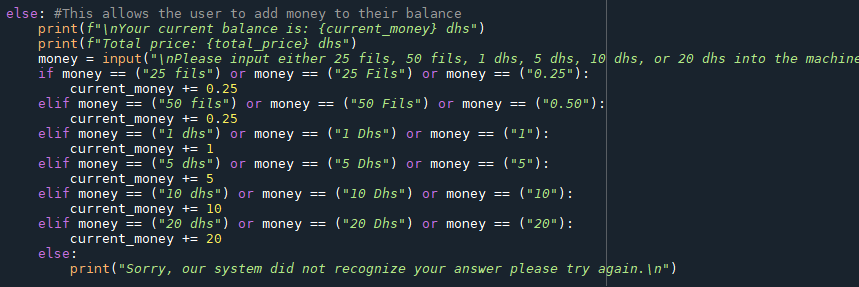
This part of the code will ask the user if the item(s) they selected is corrected, they are able to write “yes” or “no”, If they selected “No”, it will redo the item selection process from the beginning, if “Yes”, It will continue onto the next code piece of code. This is all done using a while statement to make sure that they user doesn’t input a wrong response and if conditions to go back to the beginning by increases its amount via the increase\_amount function in the definitions file if they select no, it also checks if the product is a bundle and adjusts the amount accordingly. If the user answers “Yes”, they skip this code entirely and move on to adding the product to the shopping cart.



Adding to shopping cart, Option for more shopping and calculating for total price

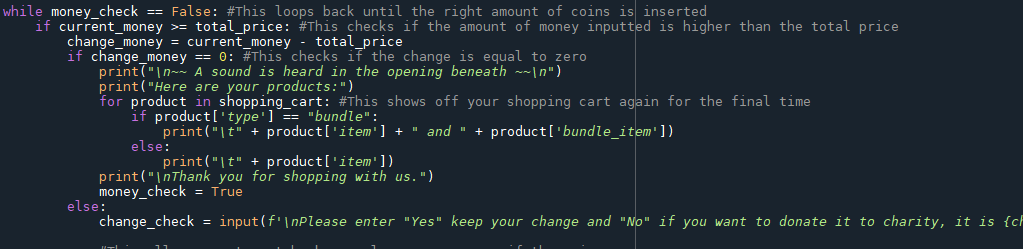
This part adds the item to the shopping cart then asks the user if they want to buy more items, if “Yes”, the process repeats from the very start with a new item in the shopping cart, if “no” then the system will calculate the total price of their shopping cart and tells the user the total price before showing their shopping cart to them. This is done via the append command to add the variable to the list named shopping carts, the input command for the response of the user, while loops to make sure that the user doesn’t accidentally write something they are not supposed to, and if conditions to cycle through the choices.

The calculations on the other hand is a basic while loop for every item in the shopping cart, it will use the += assignment operator to add each price to the total amount, and finally it uses the show\_product function in the definitions file.



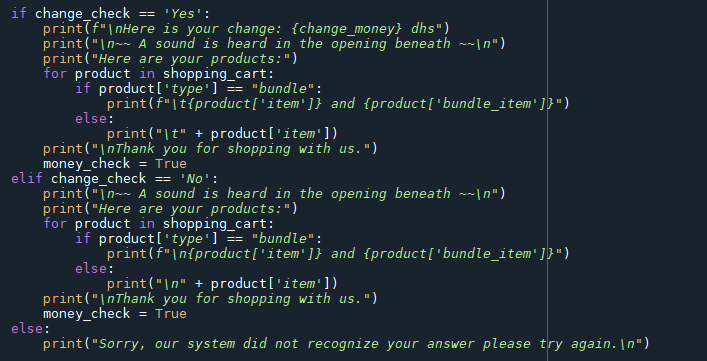
Money exchange system

This part is part of a while loop that allows the user to select the amount of money they want to input into the vending machine using a while loop, an input command, if conditions, and assignment operators. It will first ask the user to input a wide variety of options and then it will add it to the “current\_money” variable using the += operator which will continue to add up until the “current\_money” variable is equal or greater than the “total\_price” variable, if false, it will repeat the process.



The right amount of money and finale 1

This while loop occurs at the end of the code, it will check if the current\_money is equal/greater than the total\_price using an if condition, and it gains change\_money by subtracting “current\_money” to “total price” if it results to “0”, then the system will disperse the products without asking for anything else, thanking the user in the process. If it is more than 0 then the system will ask the user if they want it back or to donate it to charity. This is all done via while loops (it will keep looping the code until it reaches the end), if conditions to see if the current money is equal/greater than the total price, to see if there is change, and to print out the items, and user input to get the user’s response according to their answer of getting back their change.

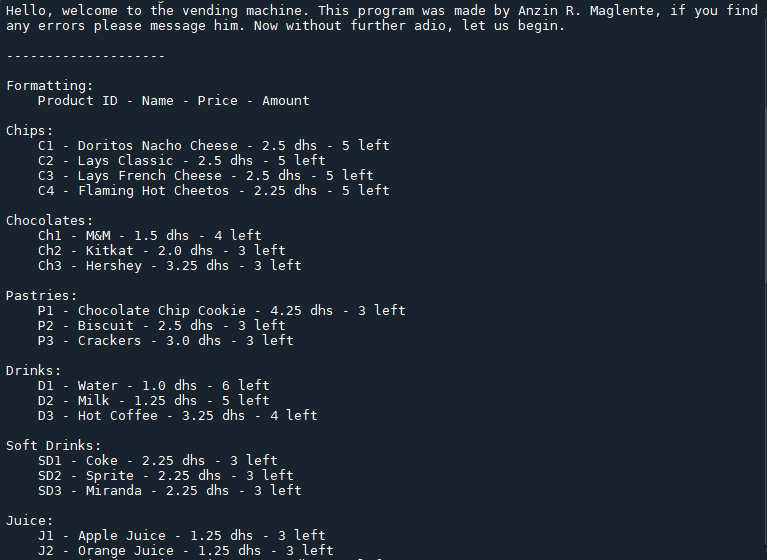


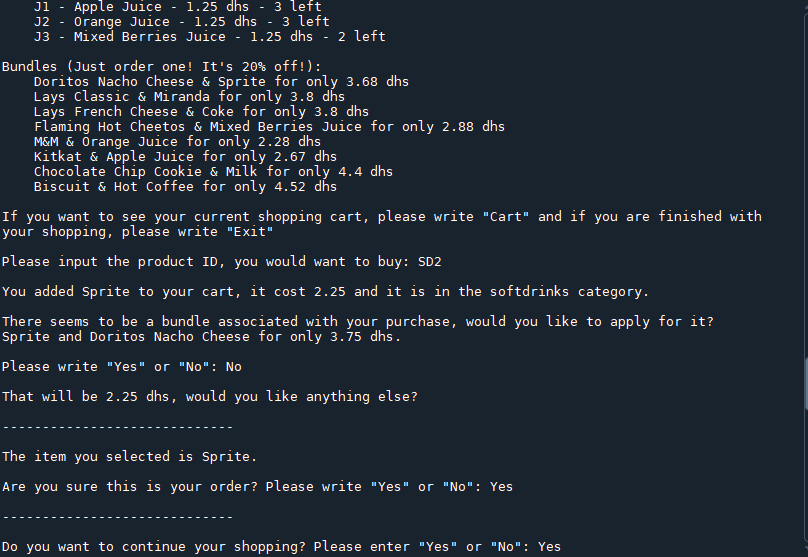
Change system, and finale 2

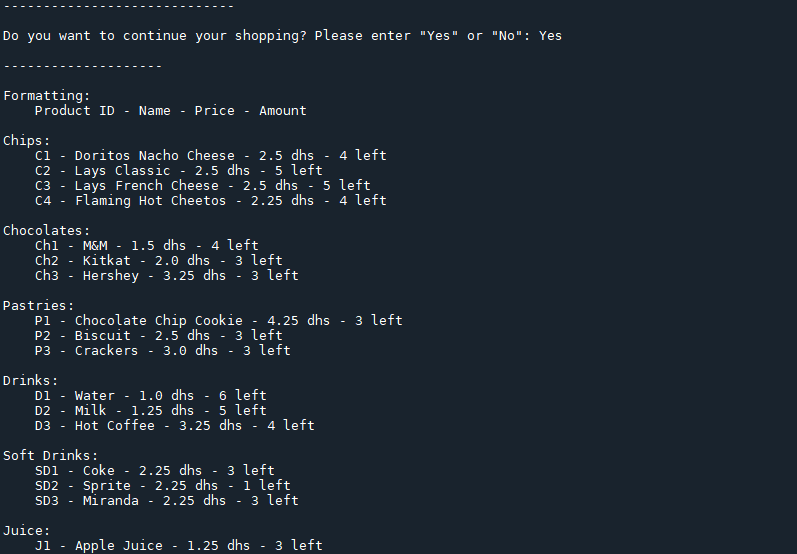
This part of the code is part of a while loop at the end of the code, this shows off the if conditions which comes after the response of the user. If the user selects “yes” then the system will give back the change and print out every product in the shopping cart. If “no”, then the user is only given the the products while the change is donated to charity, if anything else, the user is asked to repeat their answer as the system did not recognize their input, this is done by a while loop, user input, if conditions and the print function.

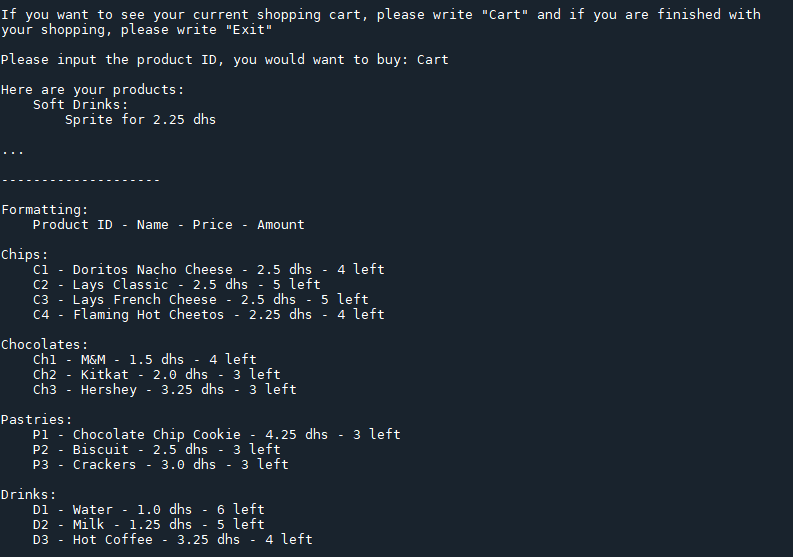
**Output**

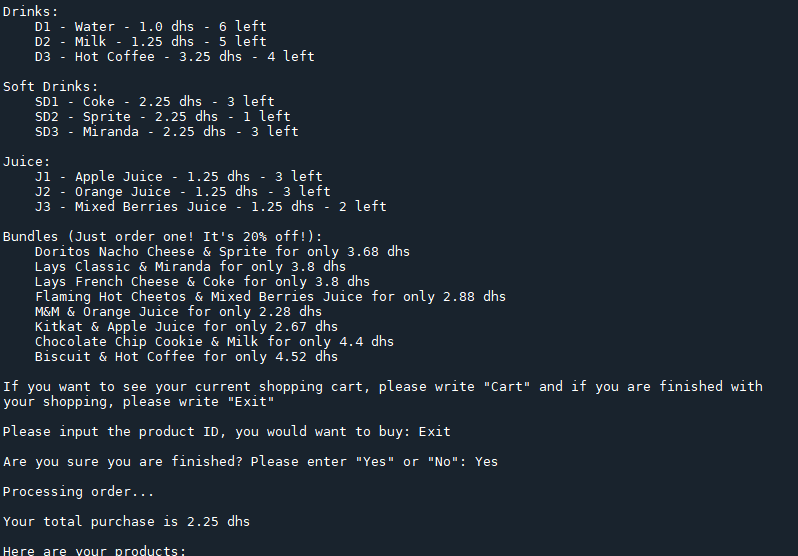


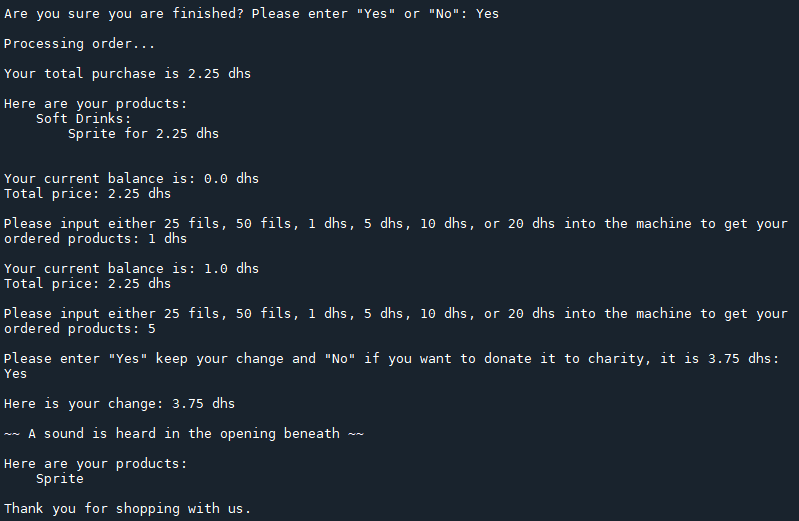












**Critical Reflection**



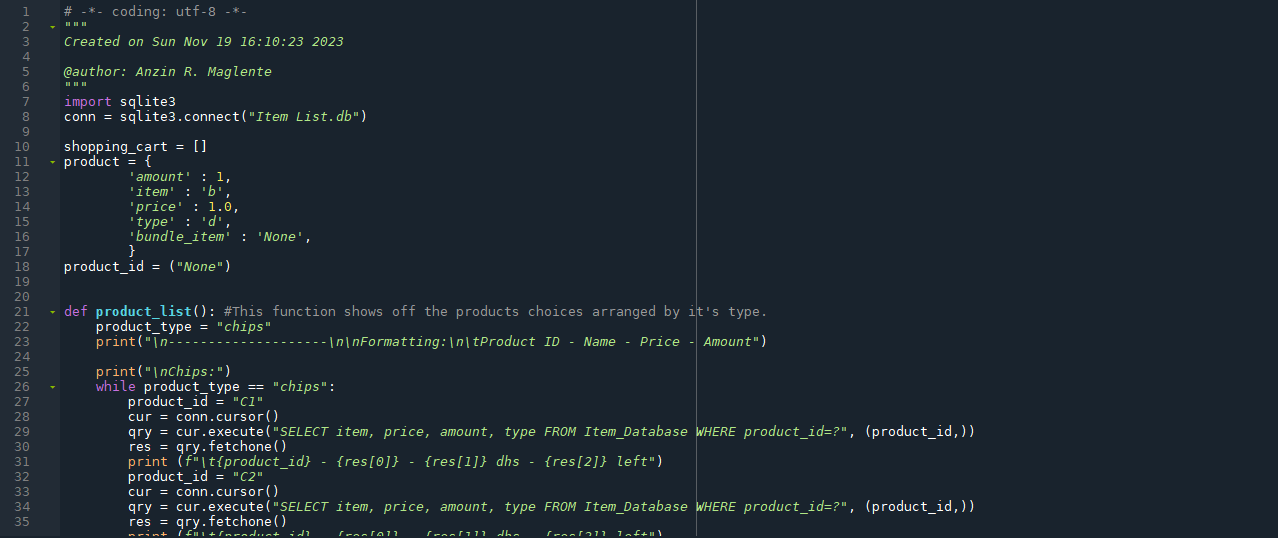
During this assessment, I had fun trying to figure out how to implement every feature I wanted to add, finding and promptly fixing the errors I found while debugging, and learning/researching new lines of code that not only improved my overall project but also my ability to code and fix problems. Needlessly to say, I am happy with how it all turned out. However, if you asked me, there are several ways my code can still be improved, for example, I believe some parts of the code could be lessened while still keeping the quality I have if I knew more about Python and its intricacies with SQLite. My code could implement some pictures and a bit more audio, and know more modules in Python. I could’ve also added more products that the user may choose from but that may confuse them if I add too much. If I may add, knowing what I know now, I bet I could make the code better like adding a new database to the code that stores the user’s shopping cart, or adding graphics/pictures to my program.

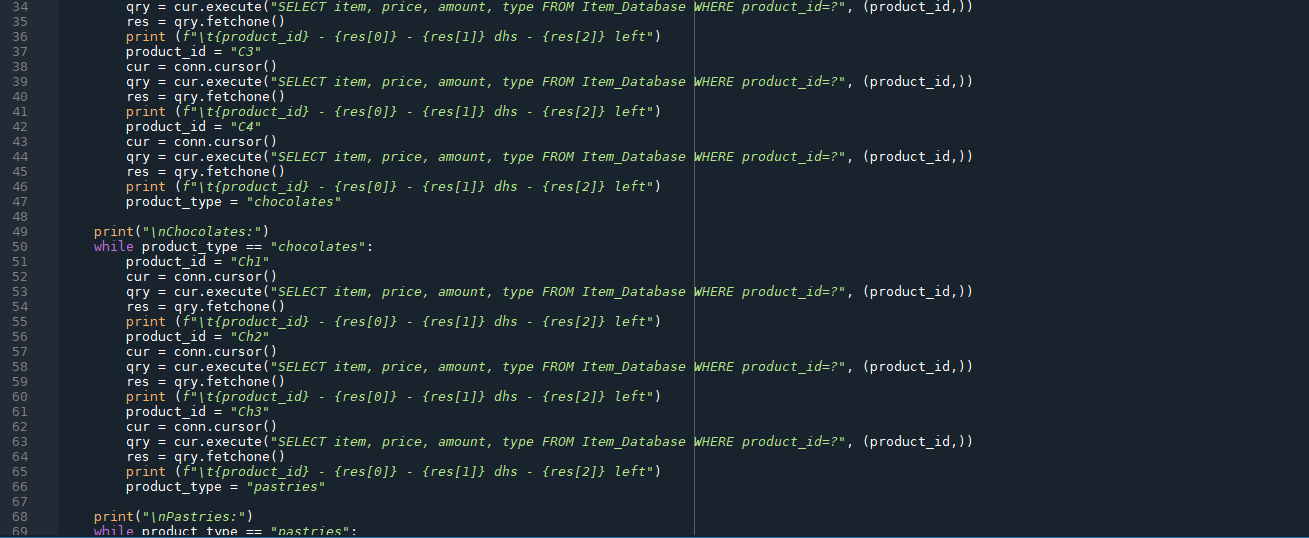
I learned a lot from this exercise and wish there were more like it in the future, it helped show me the technicalities programming as a skill has and how useful it is moving forward. Almost everyone in this world uses it one way or the other, it simplifies chores to do more in the little time we have, it helps us study and learn about things we would’ve never come across, and lastly, it teaches us that we are able to do anything if we put our mind to it.

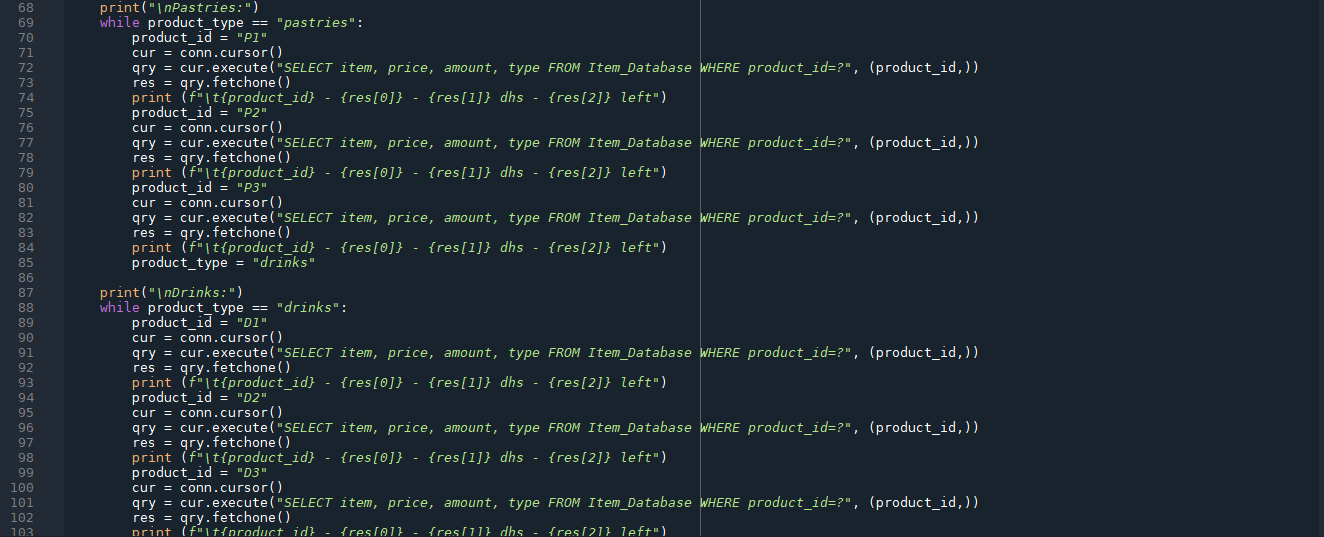
But I digress. Knowing that I was able to make something to this scale makes me excited for what’s to come, everyone starts somewhere, and my life has just begun. At the very least I can say that I tried my best. Thank you for reading this far, I hope you learned a thing or two during your readthrough of this document, below is a copy of my code.

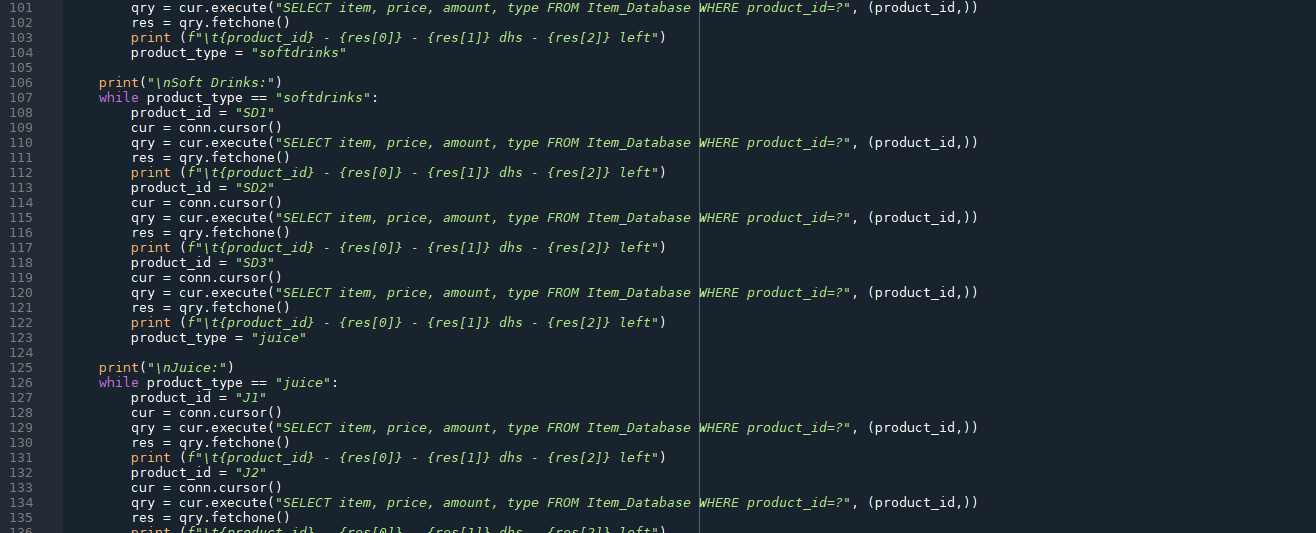
**Appendix (Definitions)**

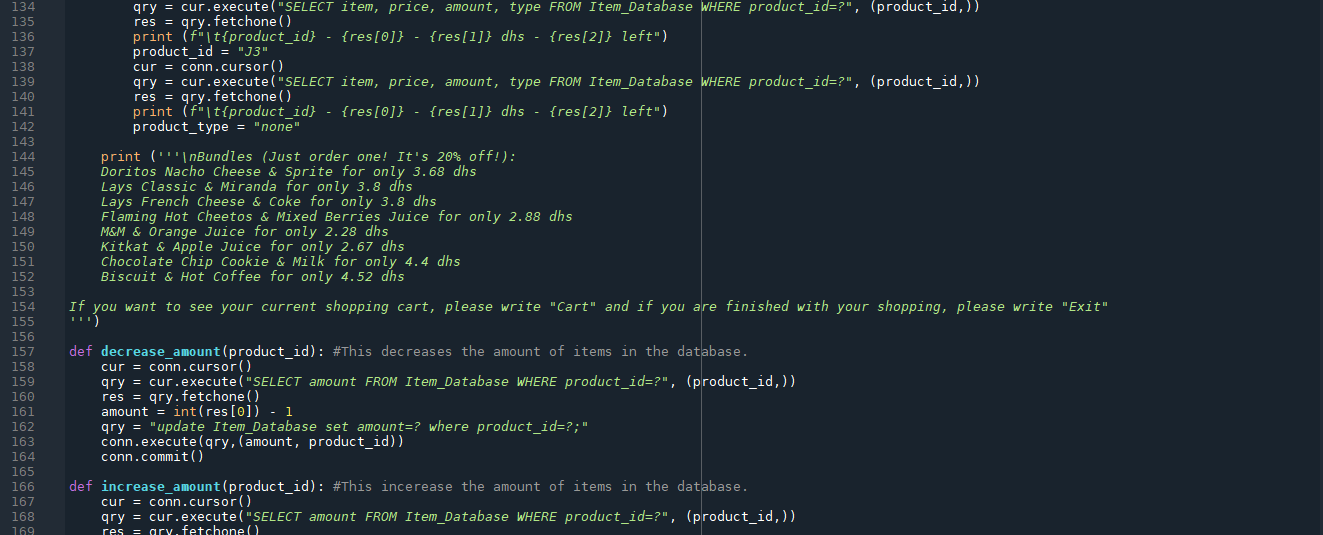


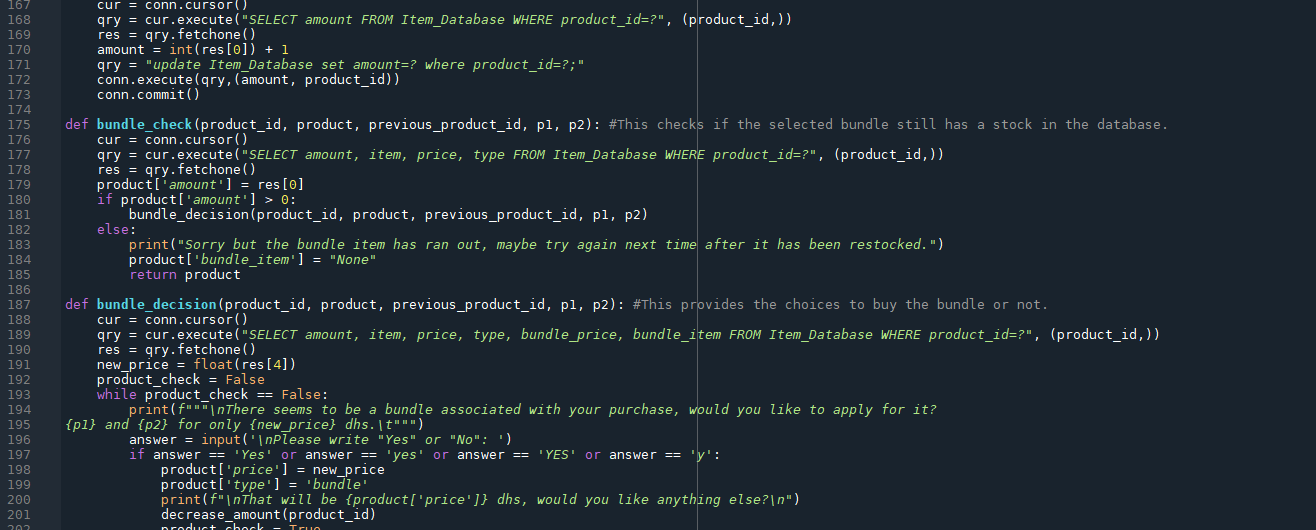


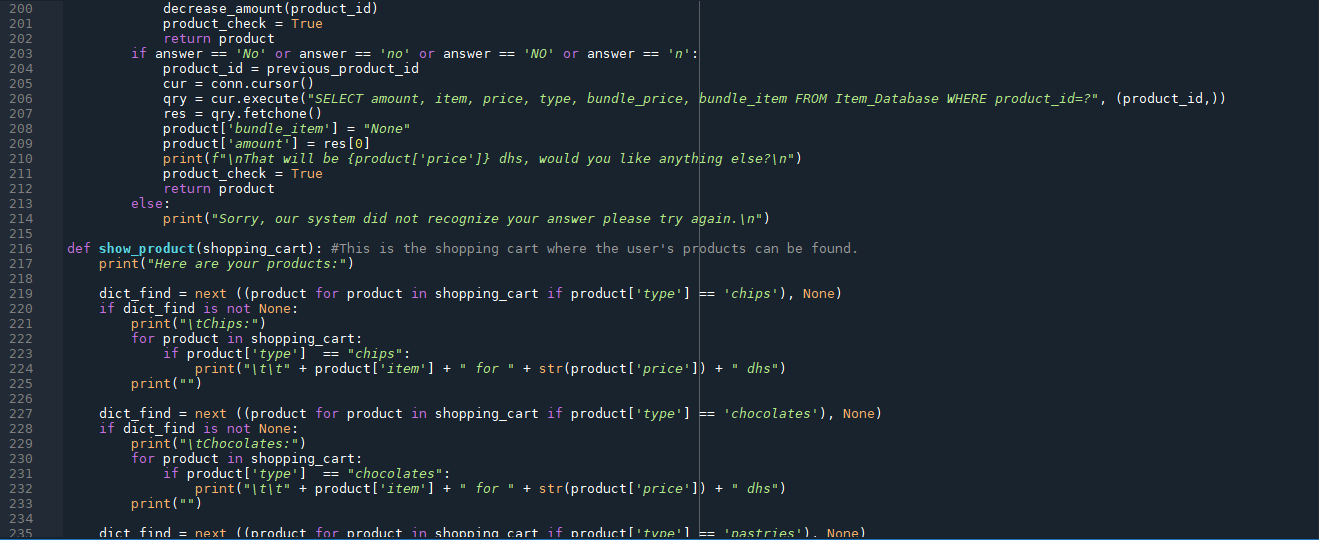


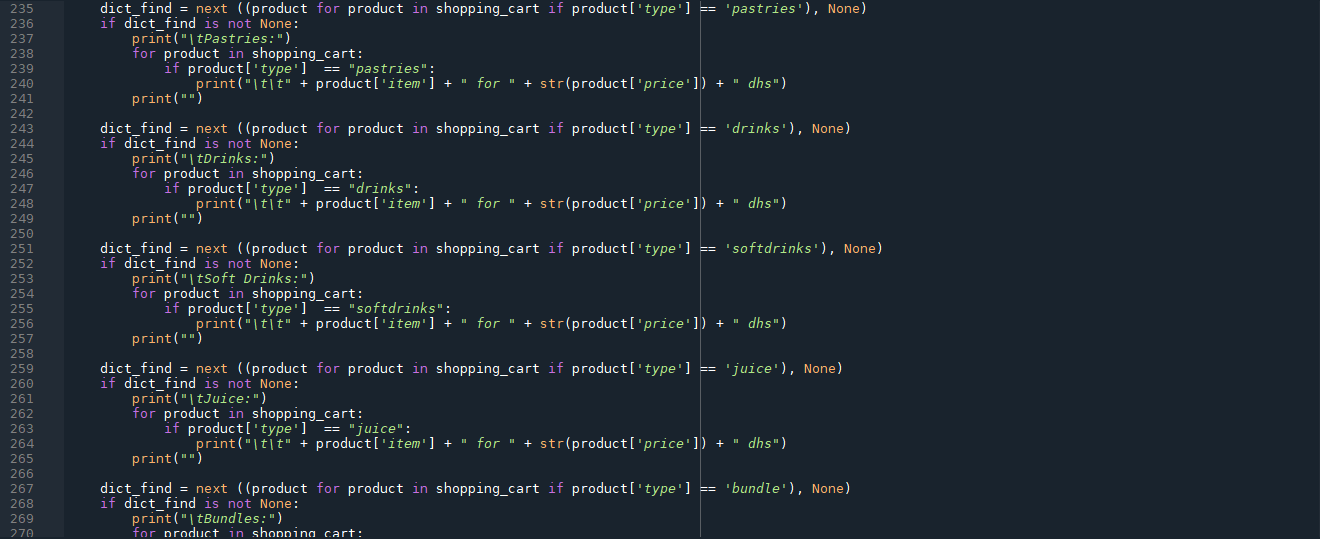


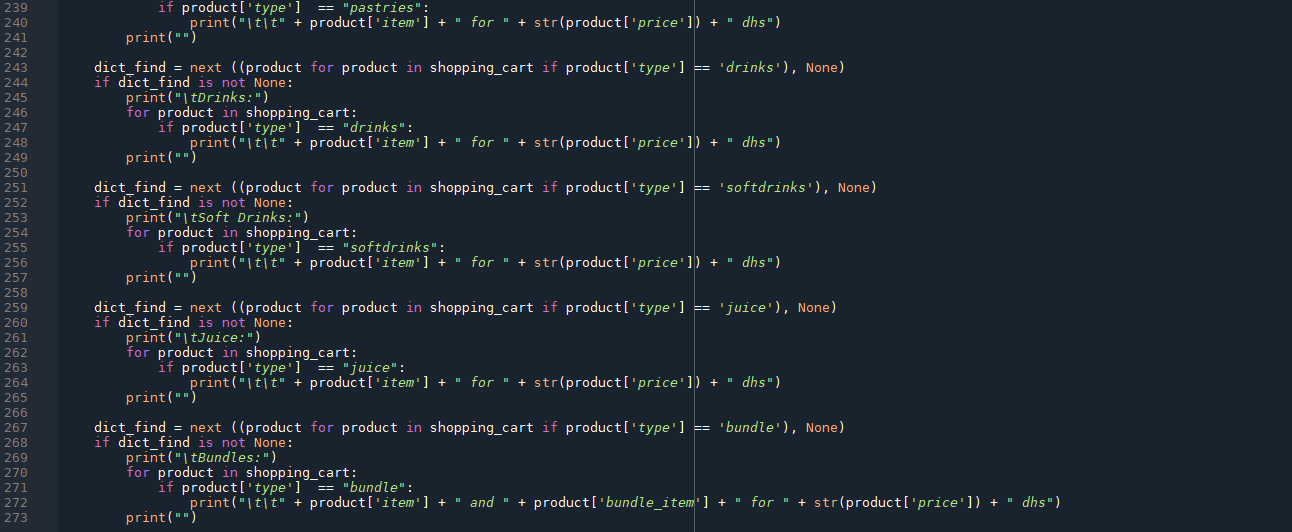






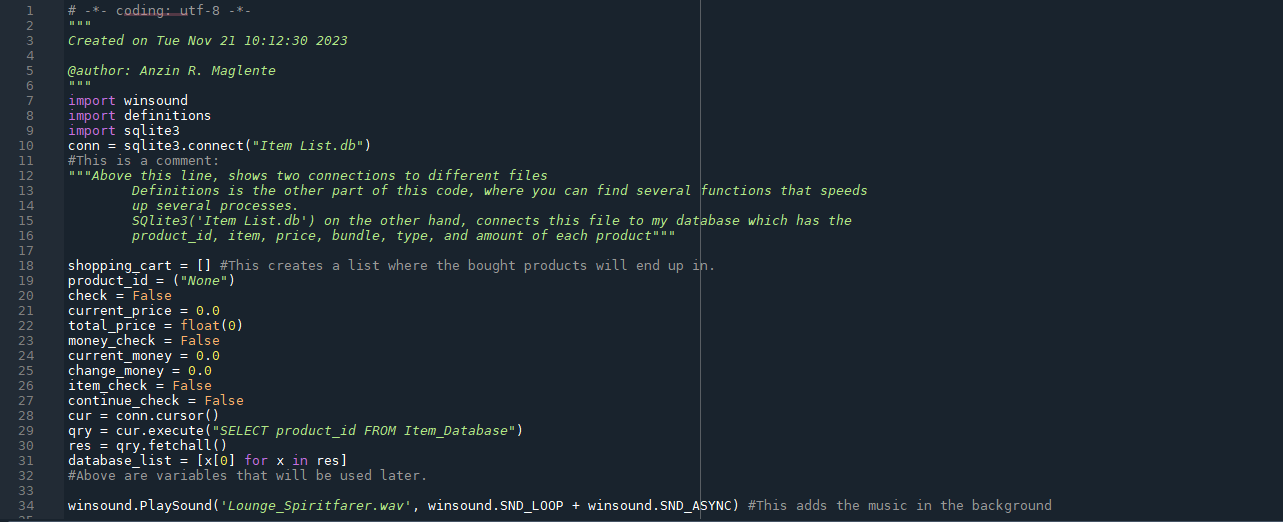


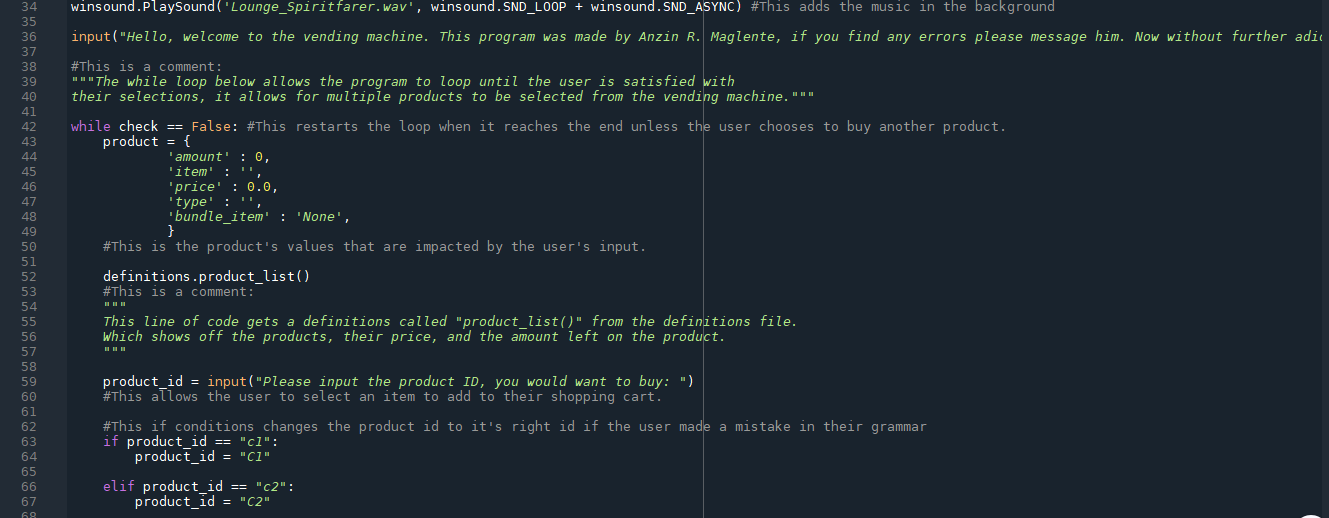


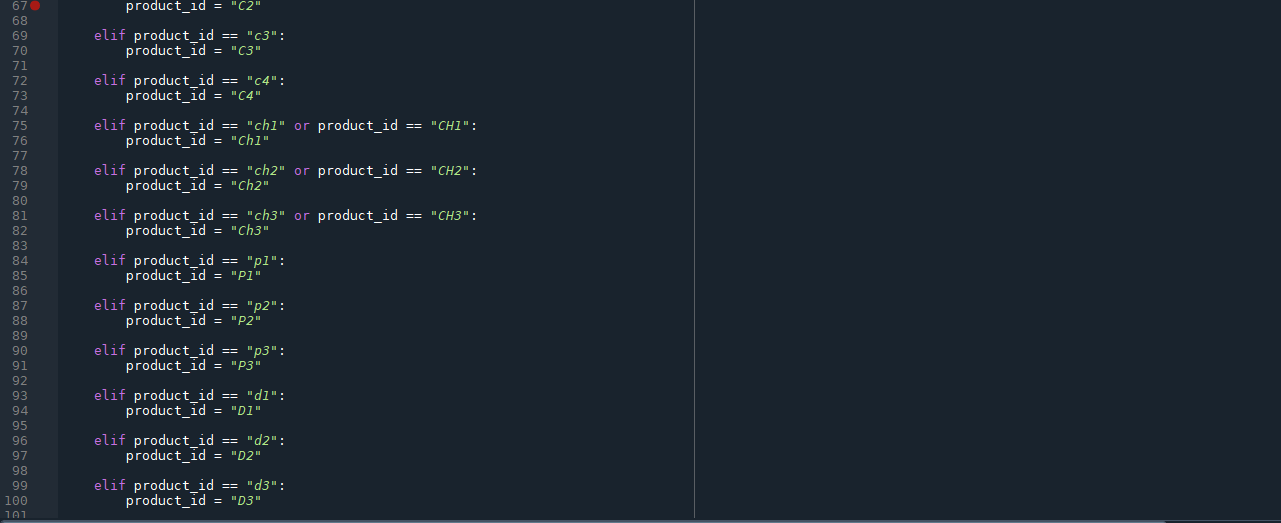


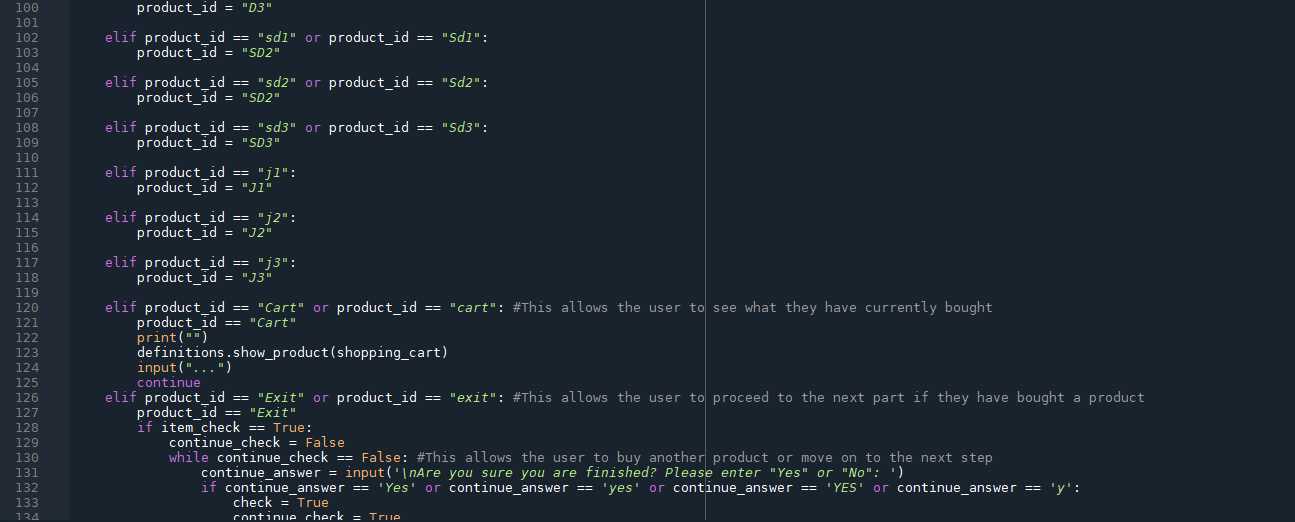
**Appendix (Main Code)**

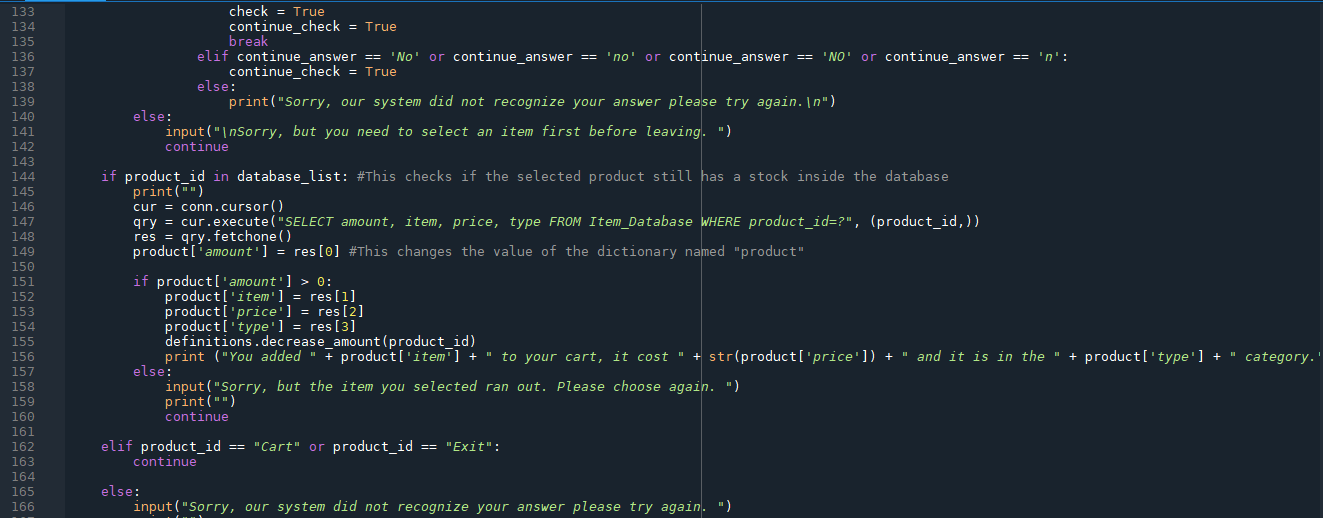


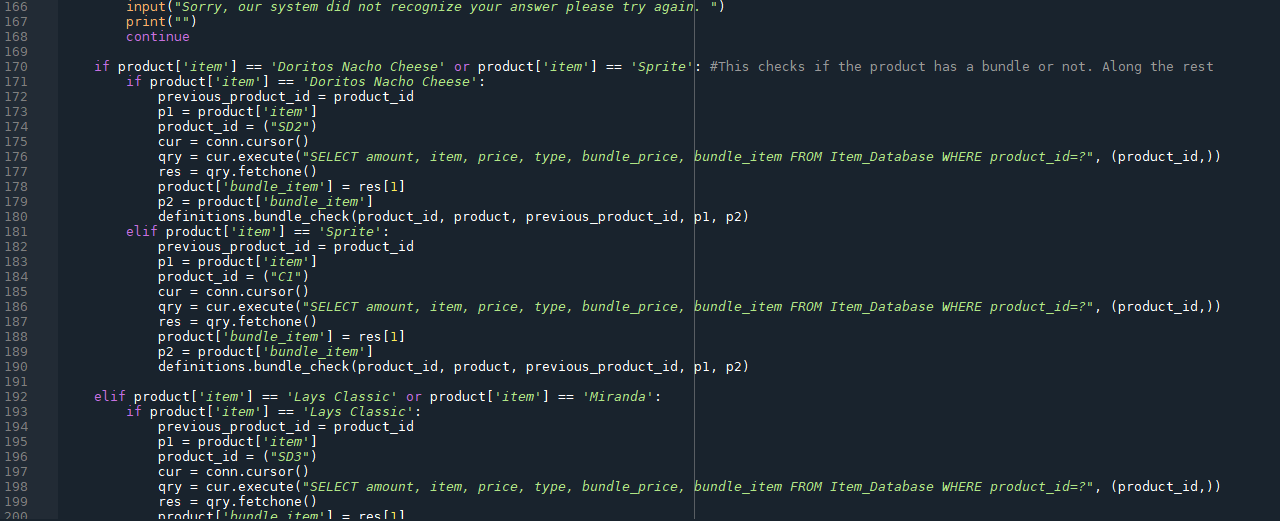


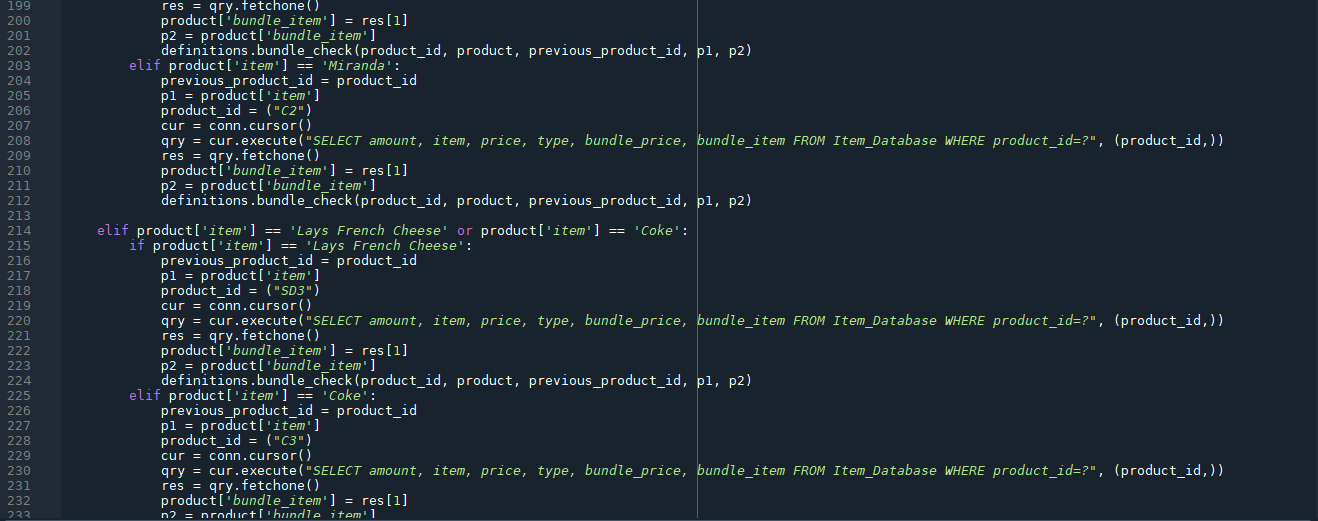


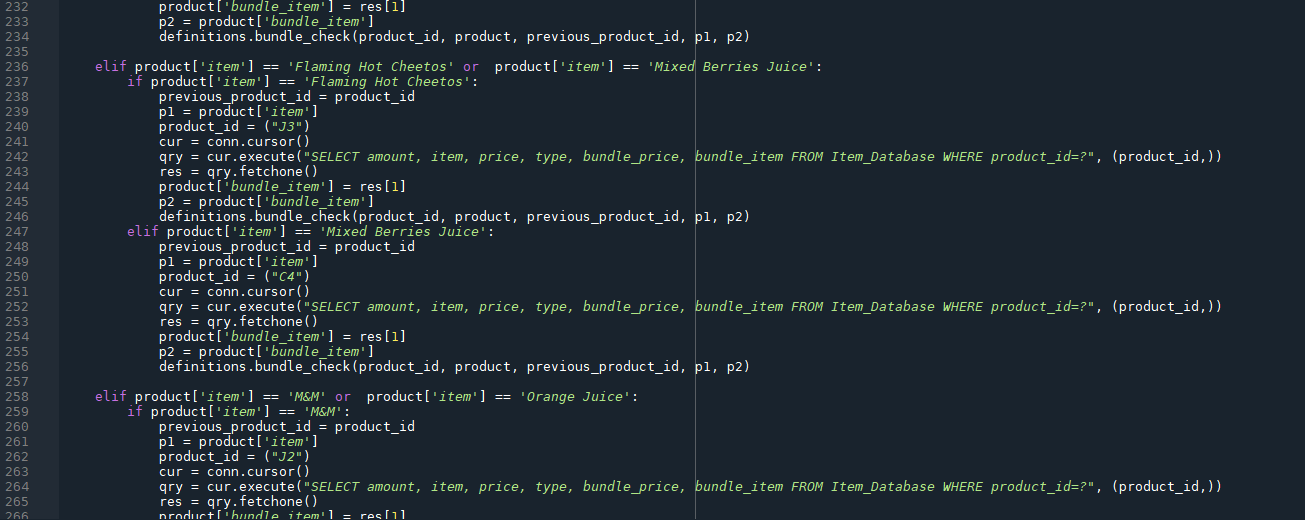


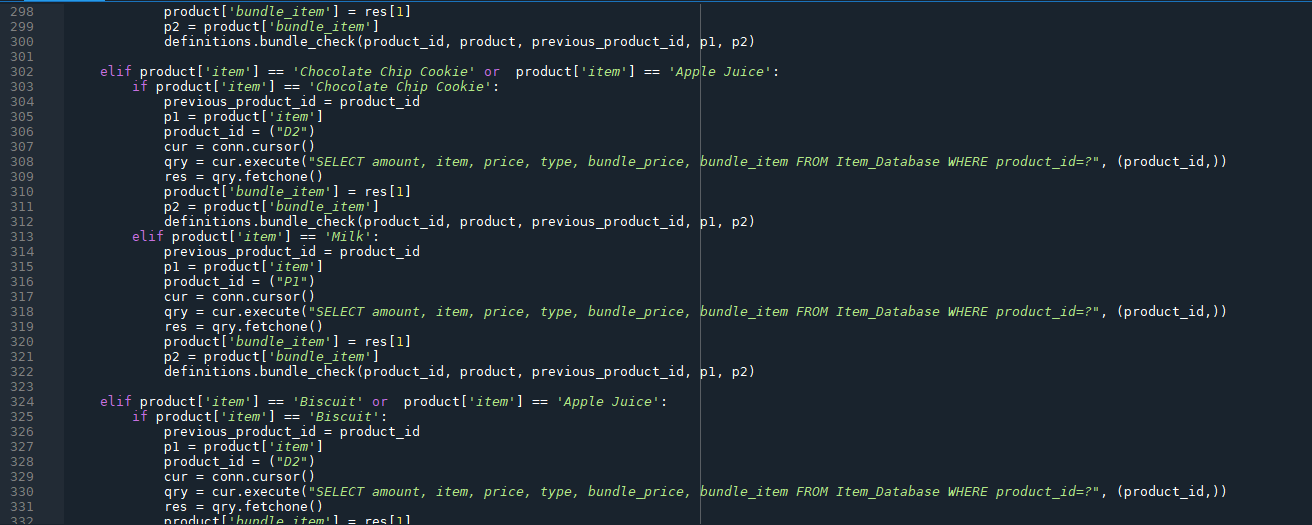


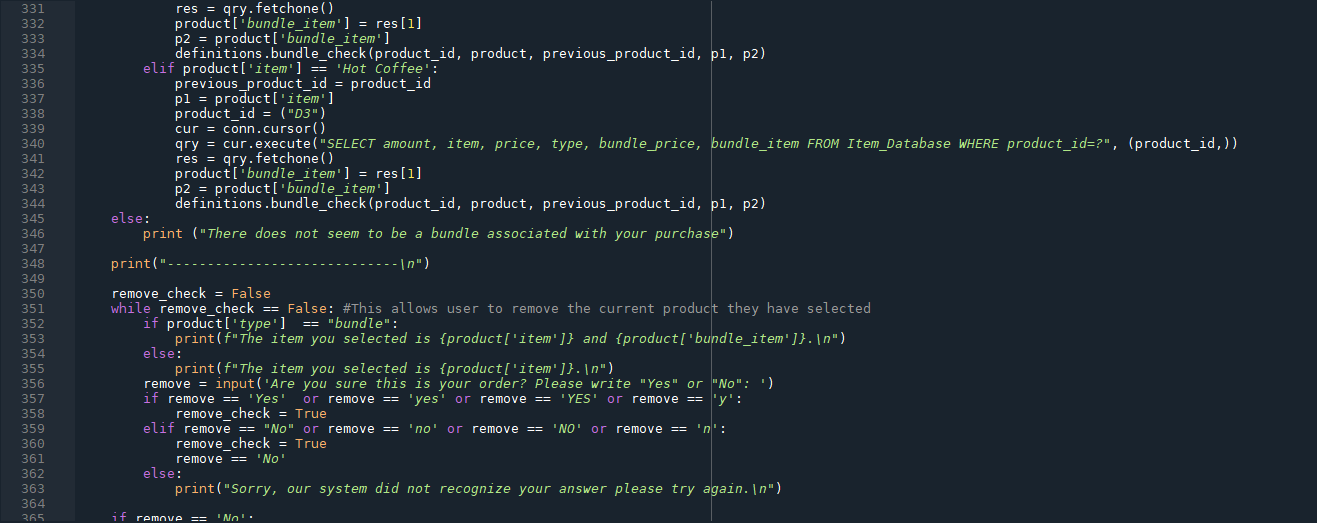


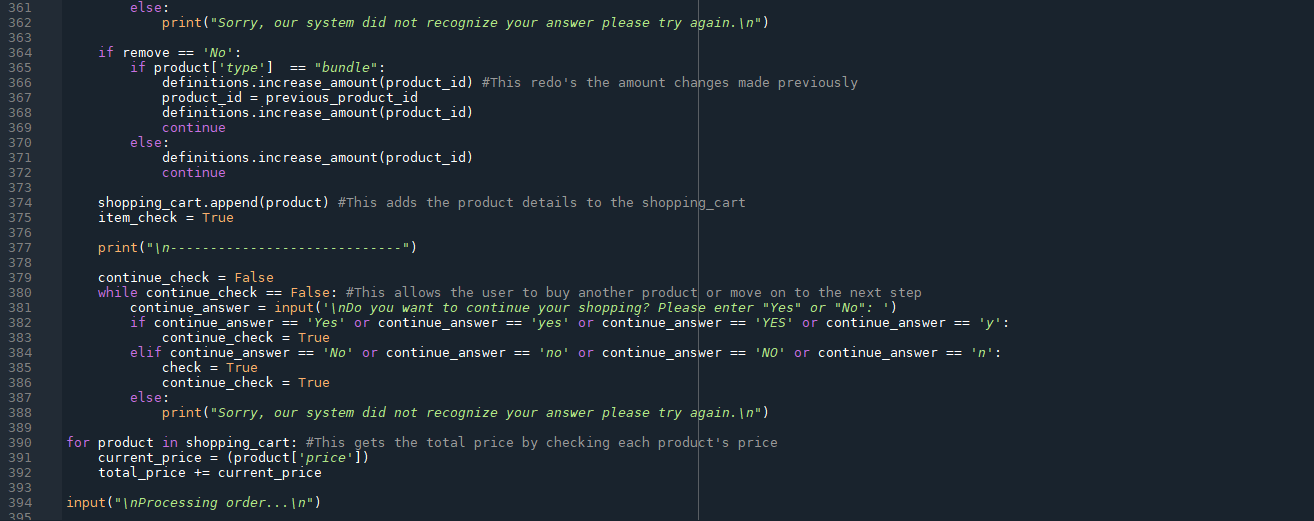


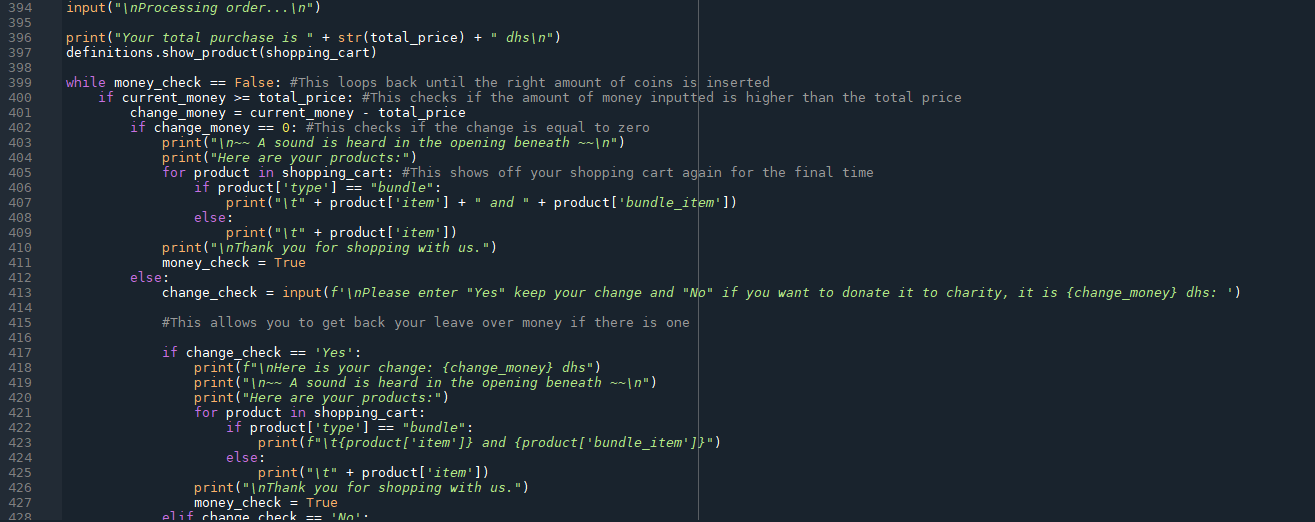


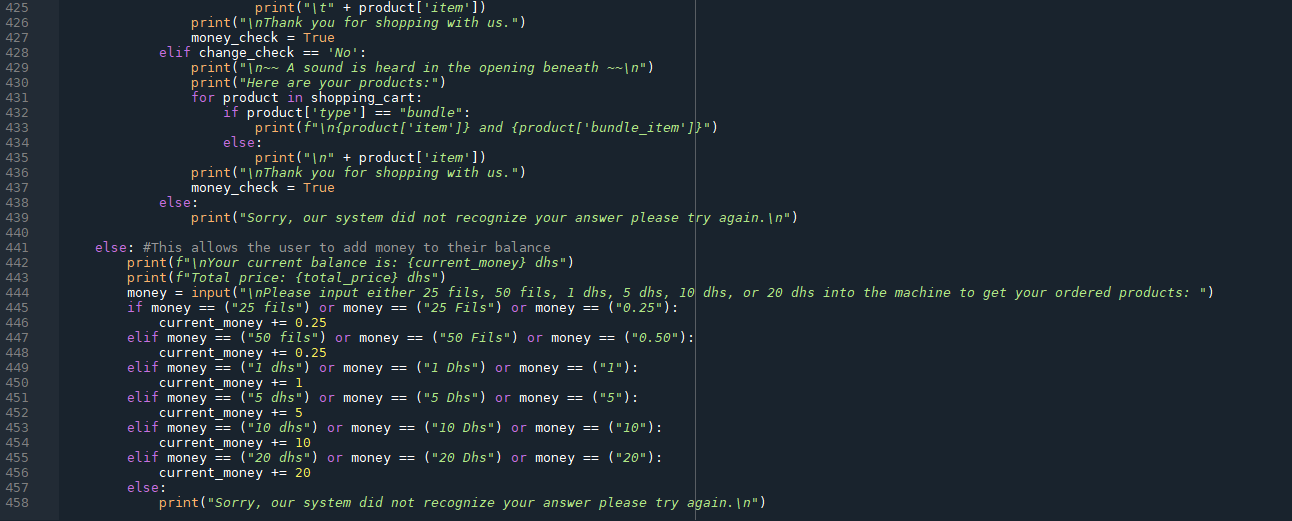












**Appendix (Restocking)**



