**PRACTICAL REPORT**

**ALGORITHM PROGRAMMING**

Week 4



Group 1

Name : Bintang Permata Putra –––– (24091397101)

Axl Rafael Gladwin –––––– (24091397019)

Adamma Muammar Ishaqi – (24091397132)

**INFORMATICS MANAGEMENT STUDY PROGRAM**

**FACULTY OF VOCATIONAL**

**SURABAYA STATE UNIVERSITY**

**2024**

**TASK : - Make a Pizza Ordering Program Using Python**

**-Create a Flowchart that Simplified the Program**

**-Create a Report and an Even in Depth Explanation of the Program.**

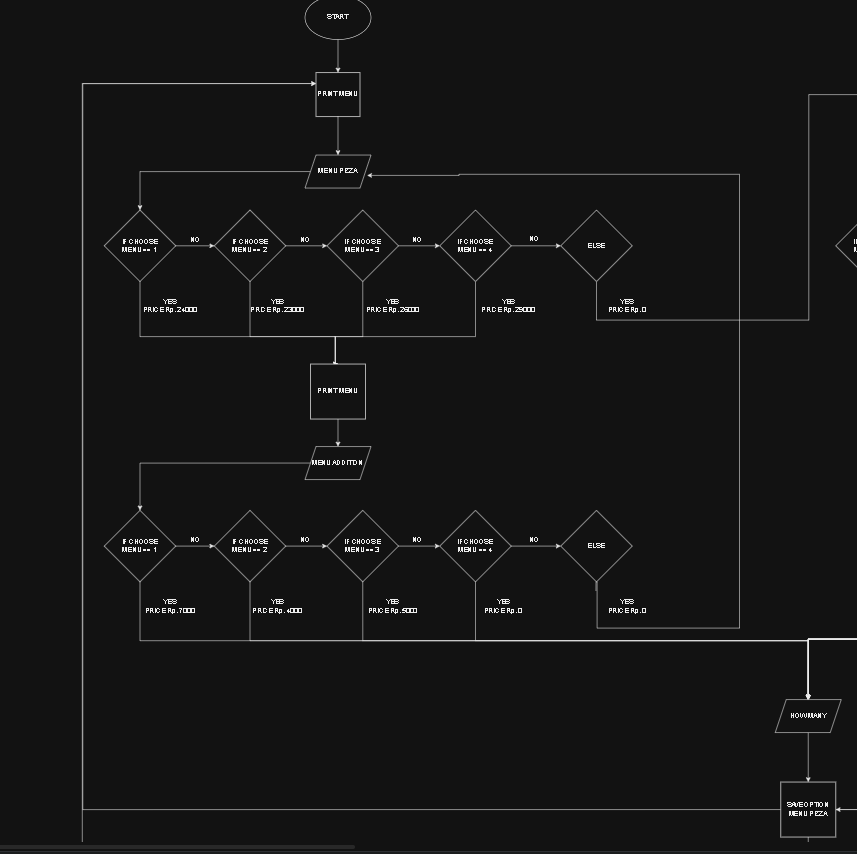
1. **FLOWCHART**

The flowchart and it’s function can be divided into four chunks.

* Pizza ordering
* Side ordering
* Drink ordering
* Receipt

Those chunks work in accordance to the following

* 1. Pizza Ordering



1.1. Pizza menu flowchart

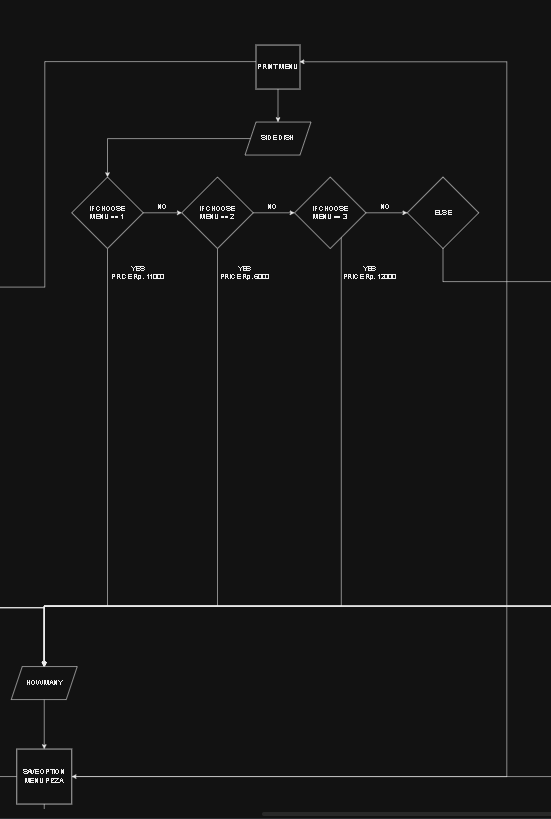
1. Start the program,
2. Print the pizza menu,
3. Ask for user input,
4. If input is in accordance to the menu continue to the extra topping,

\*Else continue to the next chunk of the program,

1. Print the extra topping menu,
2. Ask for extra topping input,
3. If input in accordance to the menu ask how many of those do they wish to order,

\*Else return to the previous pizza menu

1. Count the prizes of those and saved it
2. Return back to Pizza menu
   1. Side Ordering

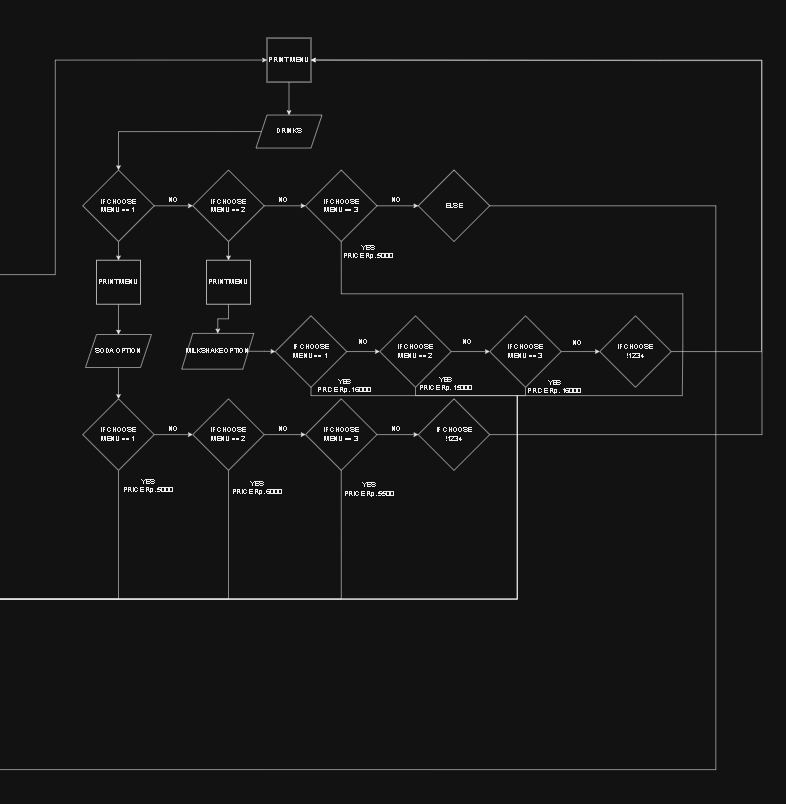


1.2. Side dish menu flowchart

1. Start by inputting unrecognize input on pizza menu,
2. Print side dish menu,
3. Ask for users input,
4. If input match the menu ask how many of the item do they wanted,

\*Else continue to drink menu,

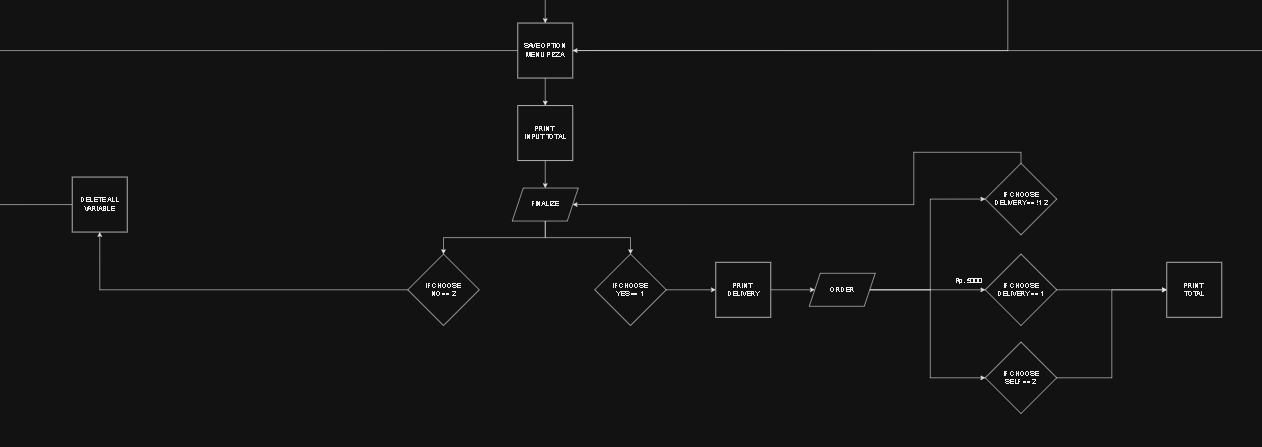
1. Count how many of those item along with their total prize and save them,
2. Return back to side dish menu.
   1. Drink ordering



1.3. Drink menu flowchart

1. Start by putting wrong input on side dish menu,
2. Print drink menu,
3. Ask for user input,
4. If input match requirement print a sub menu in accordance to the input or skip to asking how many,

\*Else went to the receipt menu,

1. Ask how many of items do you order,
2. Count and saved the price of the ordered item,
3. Return to drink menu.
   1. Receipt menu

1.4. Receipt flowchart

1. Start by putting unassigned input on drink menu ,
2. Print total prize the users order,
3. Ask if they want to reorder or not,
4. If yes then delete all value and go back to pizza menu,

\*Else print if they want delivery or not,

1. Ask input if they want delivery,
2. If yes add delivery fee to total,

\*Else if don’t add delivery fee,

\*Else return them to the receipt interface,

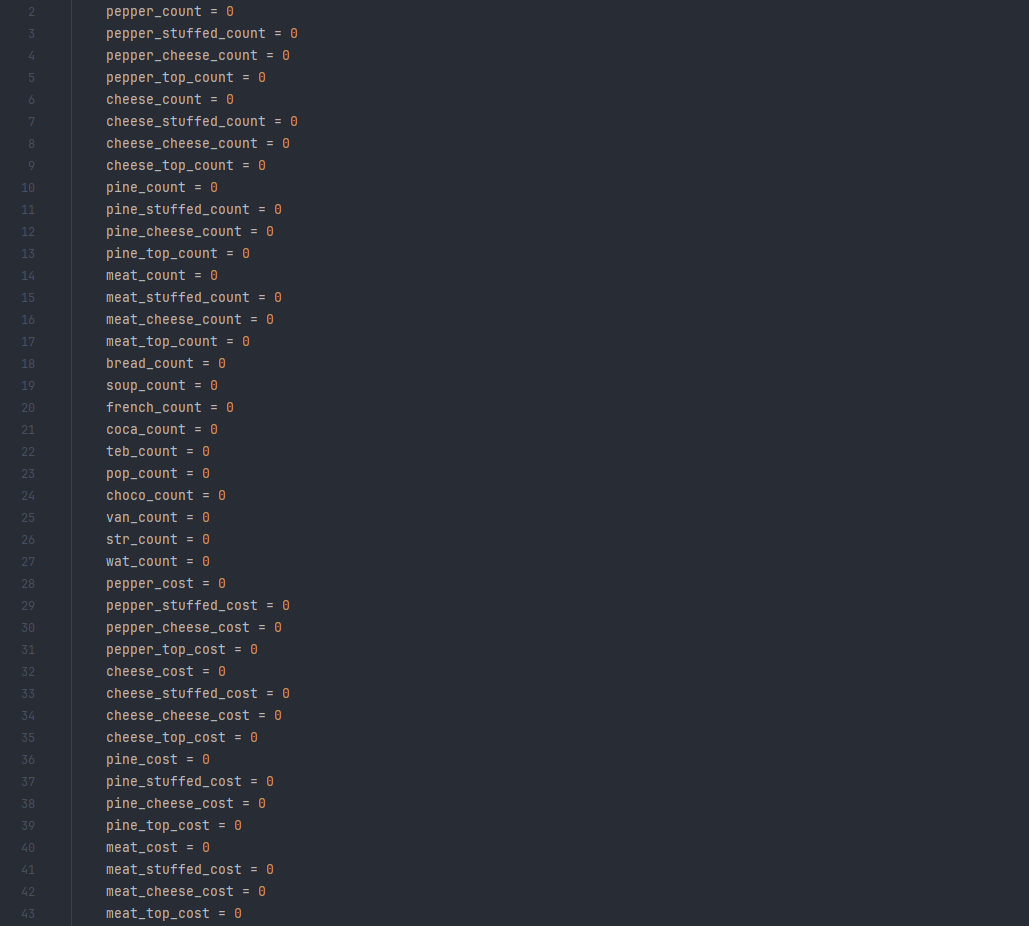
1. Print the total cost and said good bye.
2. **CODE OVERVIEW**

* Inside a Function

2.1. Pizza function

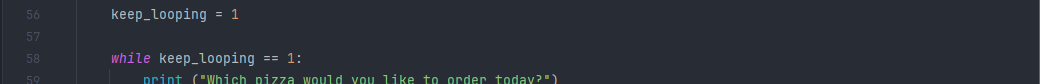
Every single bit off the code is being put inside a function. The reason for that is so that later on down the lines we able to rerun the program for our reorder system.

* Establishing Variable

2.2. Establishing and resetting the value of variable

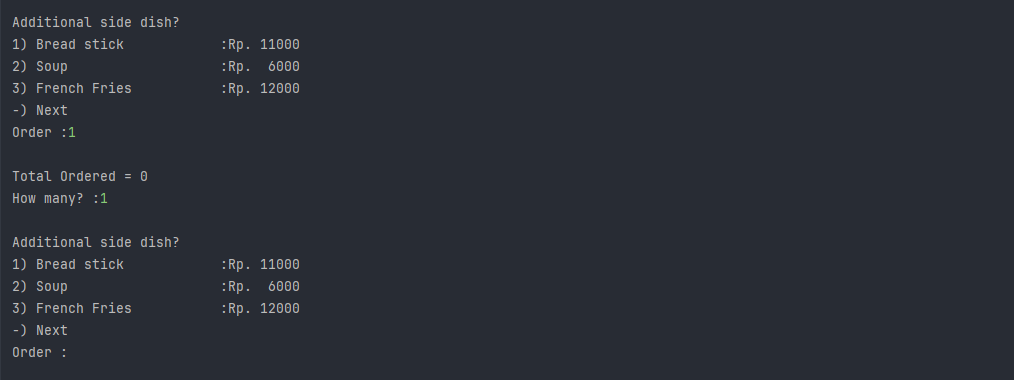
There’s multitude of reason why this entire block from line 2 to 53 is here. Firstly it’s to establish all variable inside the code. Second and somewhat more importantly it reset all variable listed above to zero which help the program to run it back for our reorder system.

* The while loop

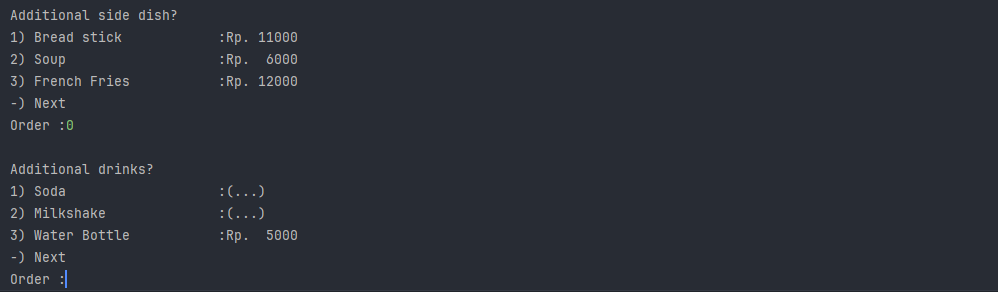


2.3. Establish “keep\_looping” and a while loop of the menu

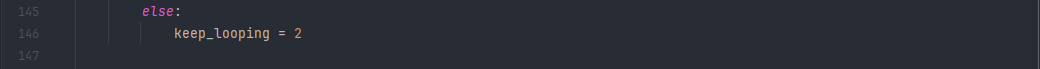
Inside the code exist a variable called “keep\_looping”, the point of this variable is to make sure the customer fully ordered everything in the current menu before continuing to the next menu. Let’s use the up incoming side dish menu as examples.

2.4. Example of menu looping back

After entering, ordering, and putting the amount want, instead of the code bleed into the next menu and keep continuing, we make it so it repeat the current menu just so if the user want another menu item or even repick menu item if the amount isn’t in accordance to what they want.

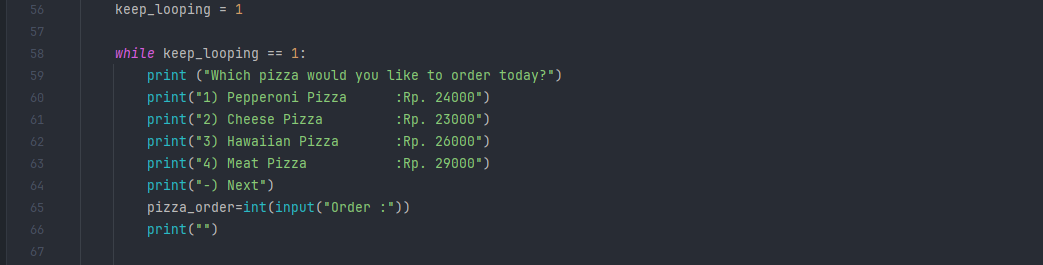
2.5. Example of menu continuing

The only way to escape the loop is by inputting unassigned value (in this case the input being 0) and by doing that the program will continue and run the next lines of code.

2.6. Changing “ keep\_looping ” to 2 and deactivating the loop

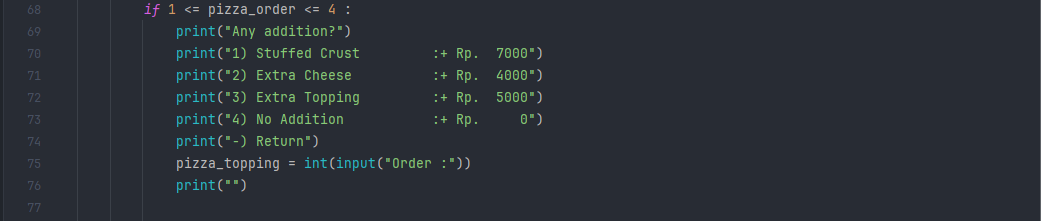
The reason why it work is because established earlier in images 2.3 that there’s a while loop with condition of it deactivating being if “keep\_looping” is any other value other than 1. So by just changing that value for unassigned interger value to something other than 1 (let’s say 2), it will break the loop and continue to the next lines, and when the next menu come around “keep\_looping” value is reverted back just like in images 2.3 and the cycle continue. This is essentially how every menu (Pizza menu, Side dish menu, Drink menu, even the Recipt) work.

* Pizza Menu

2.7. The beginning part of pizza menu

Almost all of the begining part of the menu of code began identically with the following lines of code.

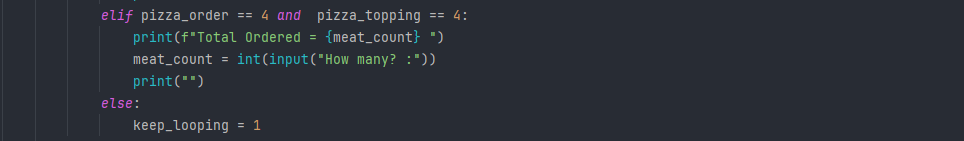
* Changing a “ keep\_looping ” variable back to 1,
* Establishing the while loop of the menu with the lock being “keep\_looping”,
* Printing the entirety of the menu,
* An input where the user able to order something,
* An empty print function to make the print in command promp less clutered,
* And continuing to next menu via inputting an unassigned value into the input function.

However each menu have their own different quirk that makes them different.2.8. The topping menu directly under the pizza menu

With the unique factor for pizza menu is it contains a secondary menu that if you pick an assigned value will lead you to be asked as to the additional topping you want on top of your pizza. After this there are two things that could happened.

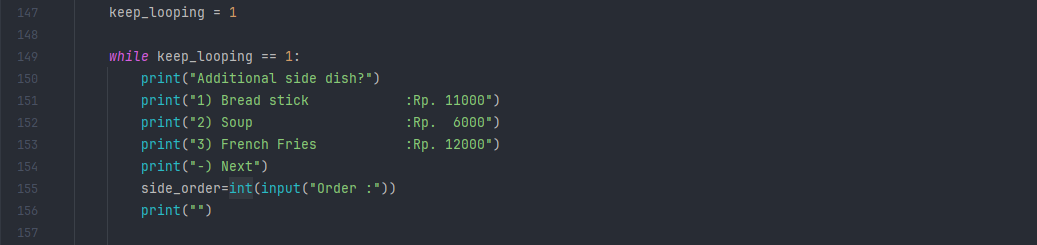
2.9. Second input for the amount of the ordered combination

Either you will be ask how many of the combination of pizza and the extra addition do you want to order using another input function.

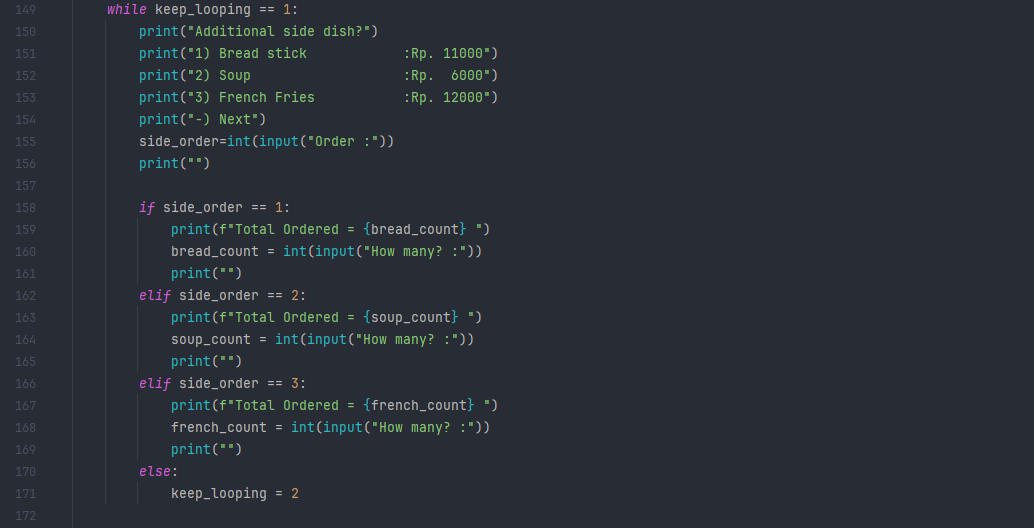
2.10. Reestablishment of “ keep\_looping ”

Or users will return back to the main pizza menu since the variable “keep\_looping” still at 1, and without that being changed into something else it will return you back to the main pizza menu.

* Sidedish Menu

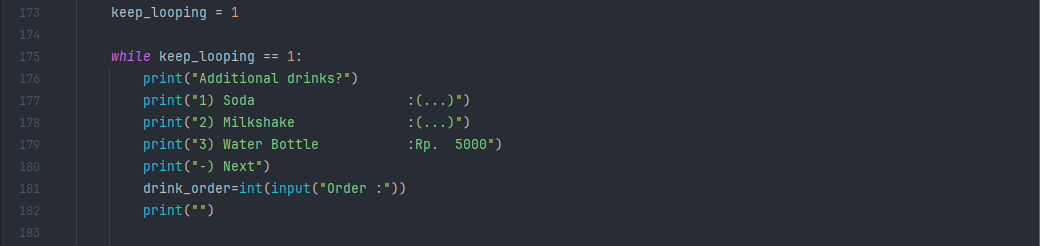
2.11. The “begining” of sidedish menu

After you input an unassigned value on the pizza menu you will continue to the sidedish menu. The uniqueness of this menu is that there’s nothing unique.

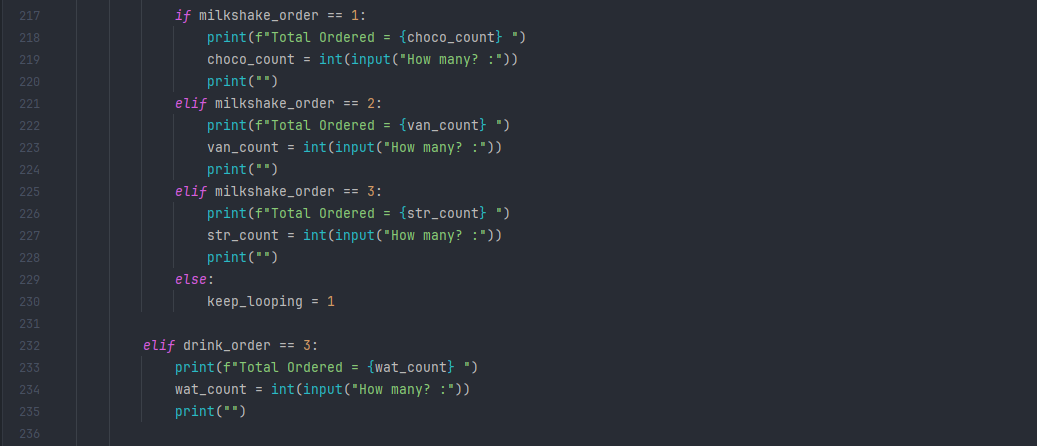
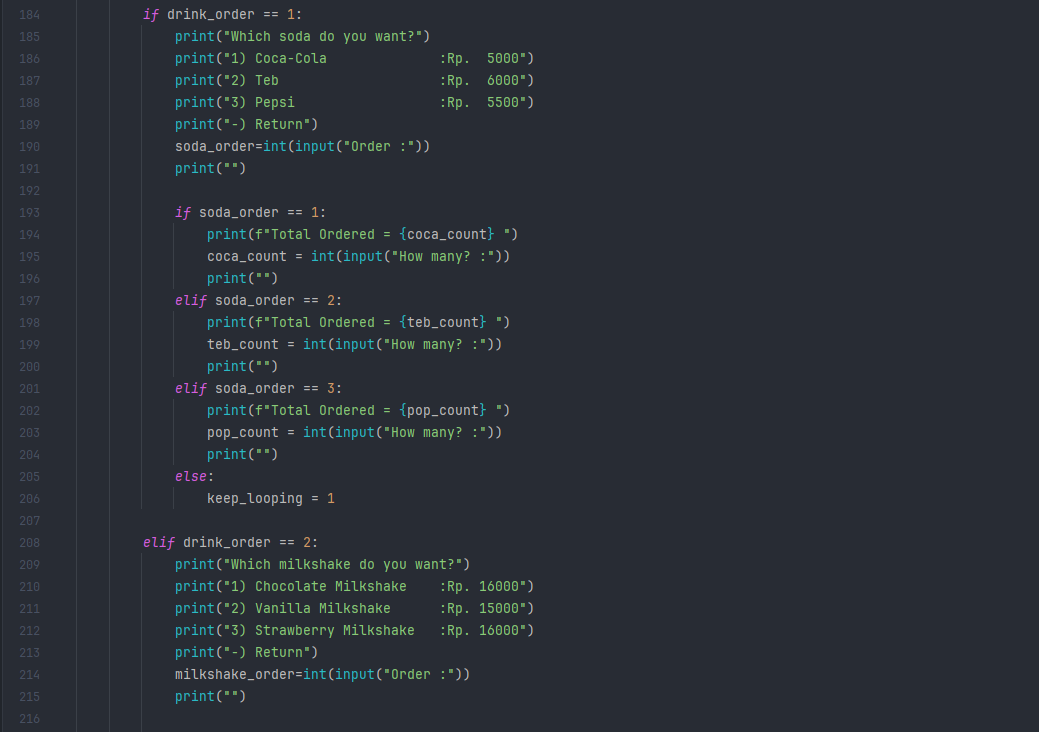
2.12. The entirety of sidedish menu code

After inputting an assigned value you will be asked how many of the items you want using an input function and after that back to the menu you go. There’s nothing inherently spesial about this outside of maybe the while loop that in this context isn’t even spesial. Which after inputting unassingned value will lead you to the next menu.

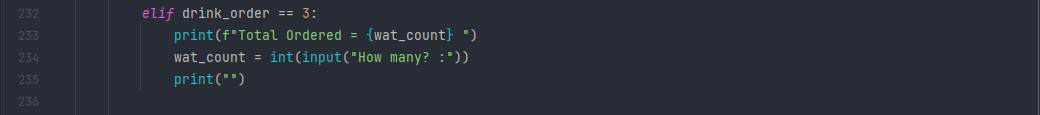
* Drink Menu

2.13. Beggining part of the drink menu

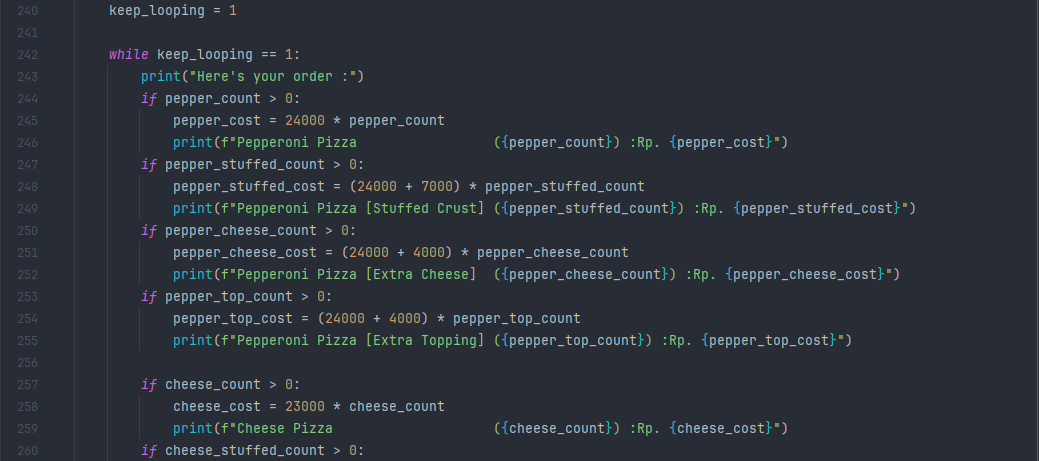
After inputting unassigned value on the sidedish menu will lead you to the drink menu.

2.14. The two submenu code inside the drink menu

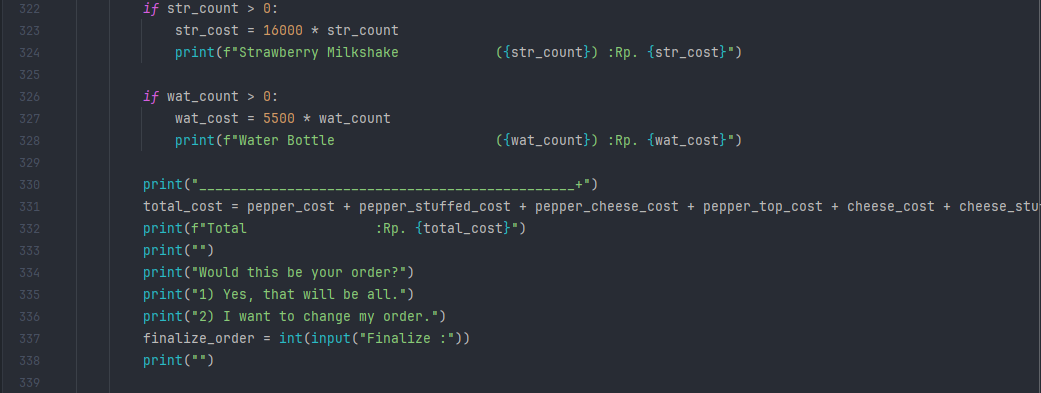
After inputting value that assigned with the “(...)” menu it will put you through a submenu just like the pizza menu. However unlike the pizza menu where the submenu is universal and every single one of the assigned input lead you there, in drink menu each one of the value that was assigned with a submenu lead you to their own submenu, each with their own unique menu and option. After which you can pick how many drinks inside the submenu using the assigned value on their menu or return to the main drink menu by inputting unassigned value to the input function.

2.15. The other drink “menu” without a submenu

However if you input an assigned value with no submenu in the main drink menu it will lead you to the usual how many question and loop back to the main drink menu and will continue until you input unassigned value in which case you will be lead into the next menu.

* Receipt

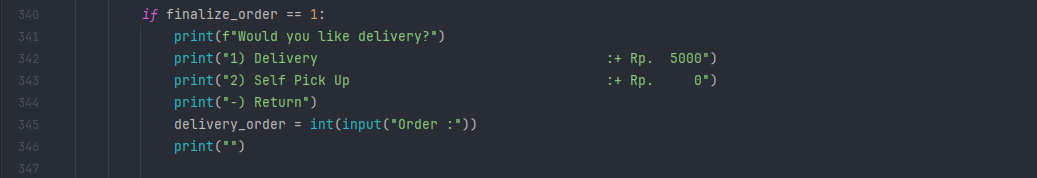
=======================================================

2.16. The begining part of receipt

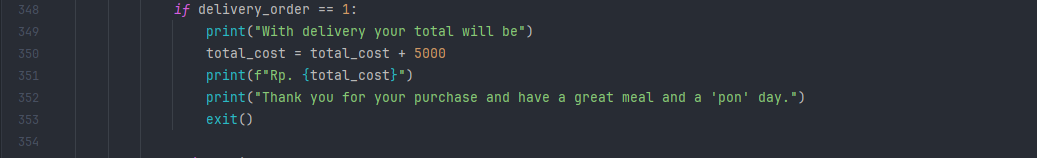
After inputting unassigned value in the drink menu users will be lead here to the receipt. There’s a lot going on, so here’s a tldr of everything from top to bottom.

* Establishing “keep\_looping” to 1 for later purpose,
* Establishing also the while loop for again later purpose,
* Checking the order being made and calculate the price of them,
* Printing them to show the amount of individual item being bought along with the price,
* A “\_” train to seperate the menu order from the rest of the print,
* Calculation of the total item bought,
* Printing the total cost of the order,
* Printing the message if users want to reorder or not,
* Asking user input of users about the discussed topic,
* Blank print so that it look neat.

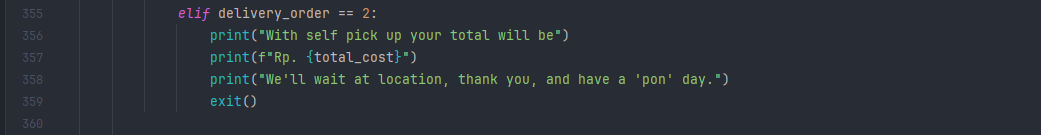
After all of that you will be given another option whether you want to reorder or finalize the buy.

2.17. Submenu after inputting assigned value “1” in receipt for delivery option

If you input to finalize your order you will be asked for delivery option.

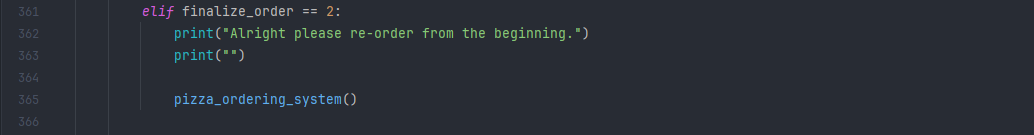
2.18. An exit command for the delivery option

If you pressed assigned value “1” you will be greeted with additional cost to the total cost and more importantly an exit from the “pizza\_ordering\_system” function we’re in.

2.19. An exit command for self pickup option

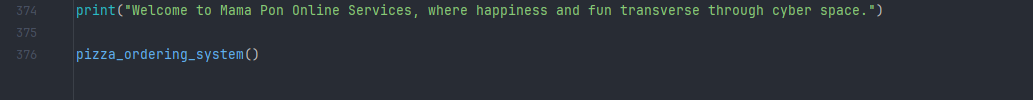
Else if you pressed the second assigned value you will just be given your regular total with no addition.

And if you pick an unassigned value because of the while loop being established earlier the receipt will be reprinted and you will be allowed to pick the second option outside of finalize. This is also what happened if you pick unassigned value on the “finalize\_order” input.

2.20. The reorder system.

If you pick the second assigned value you will go to the reorder system, which is putting you back all the way to the pizza menu with all your order resetted. This is the reason why the program is inside a function, so that it will be easier to loop it from the very end to the very beggining.

* Starting the Function

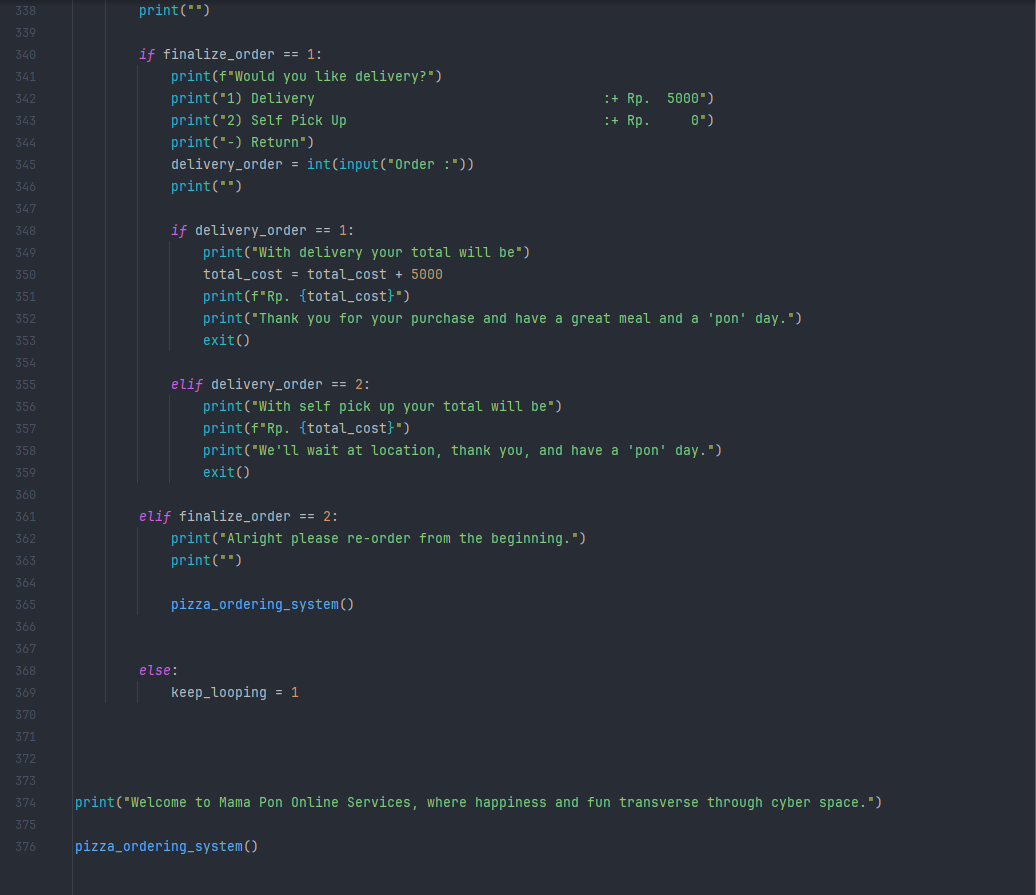
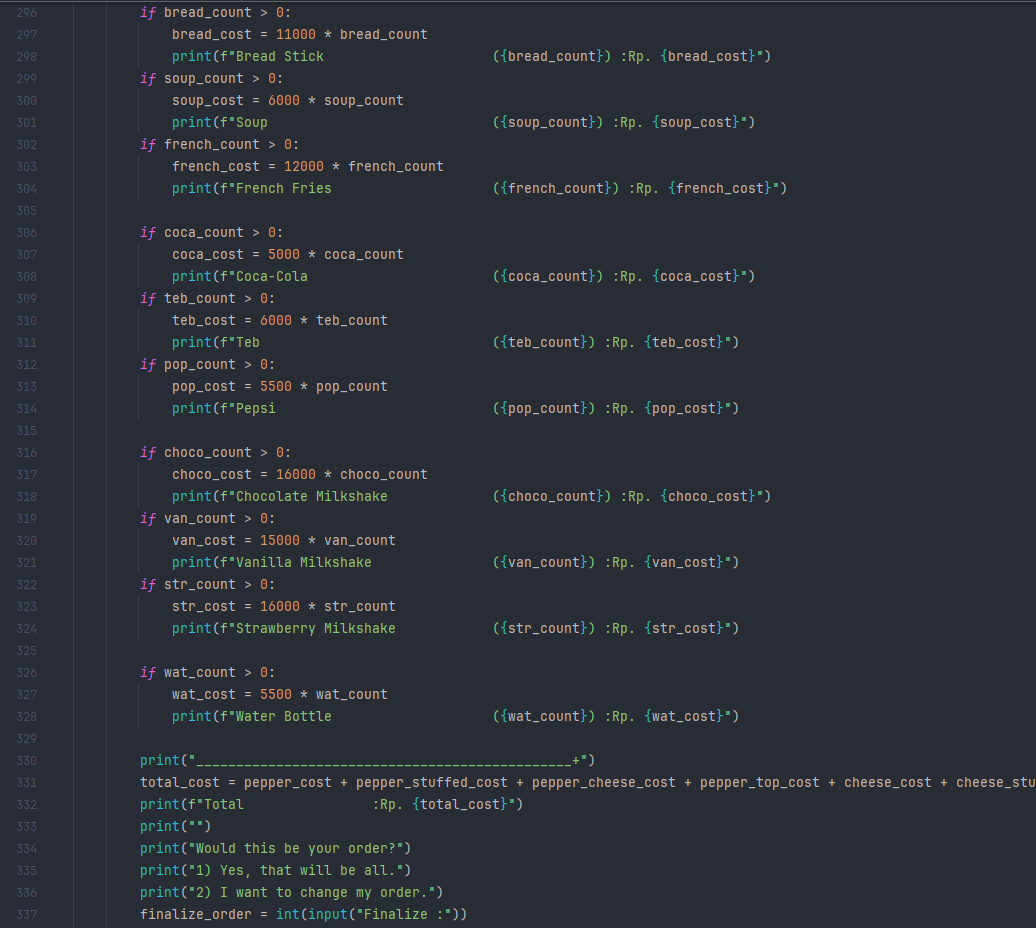
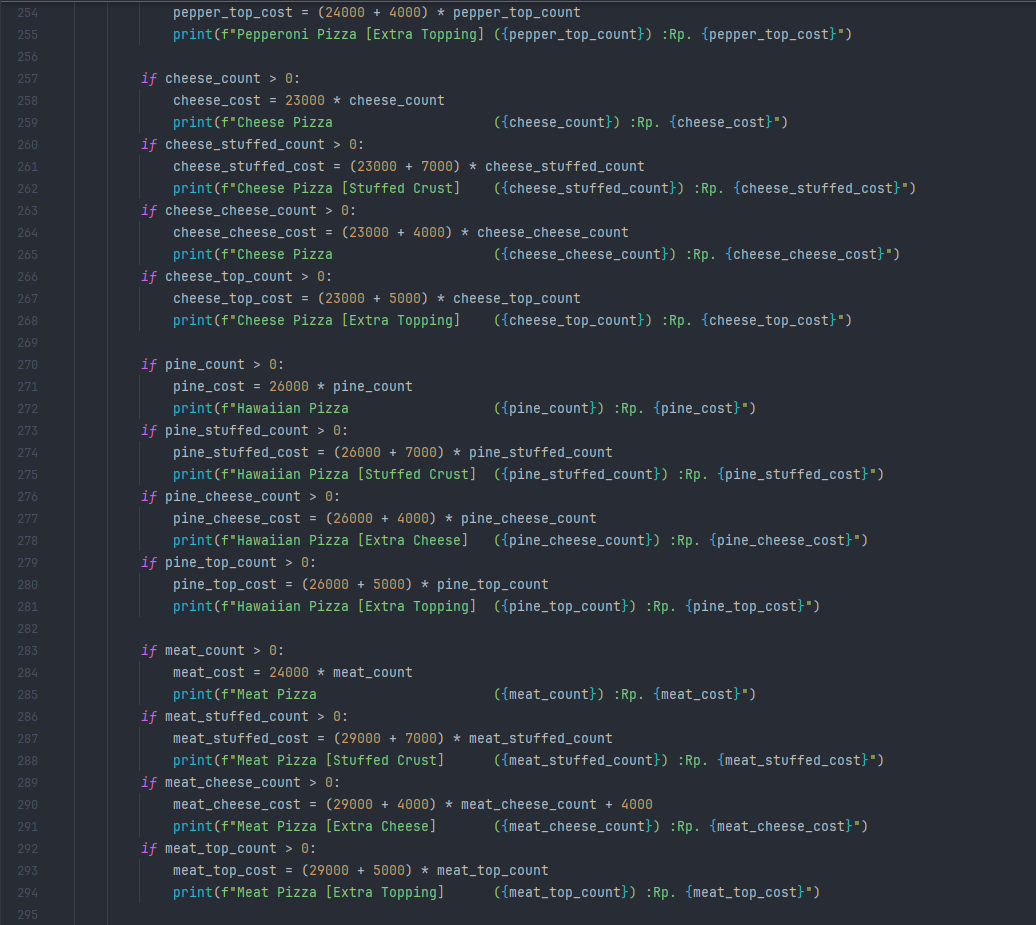
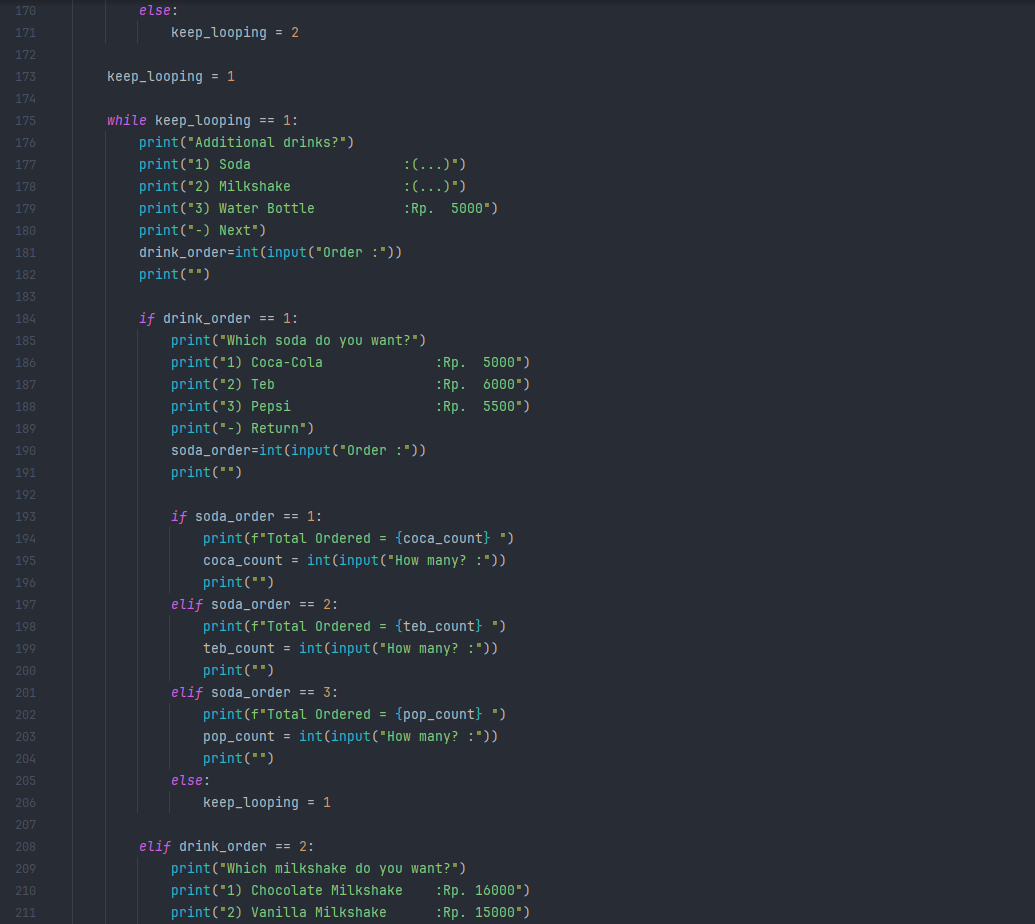
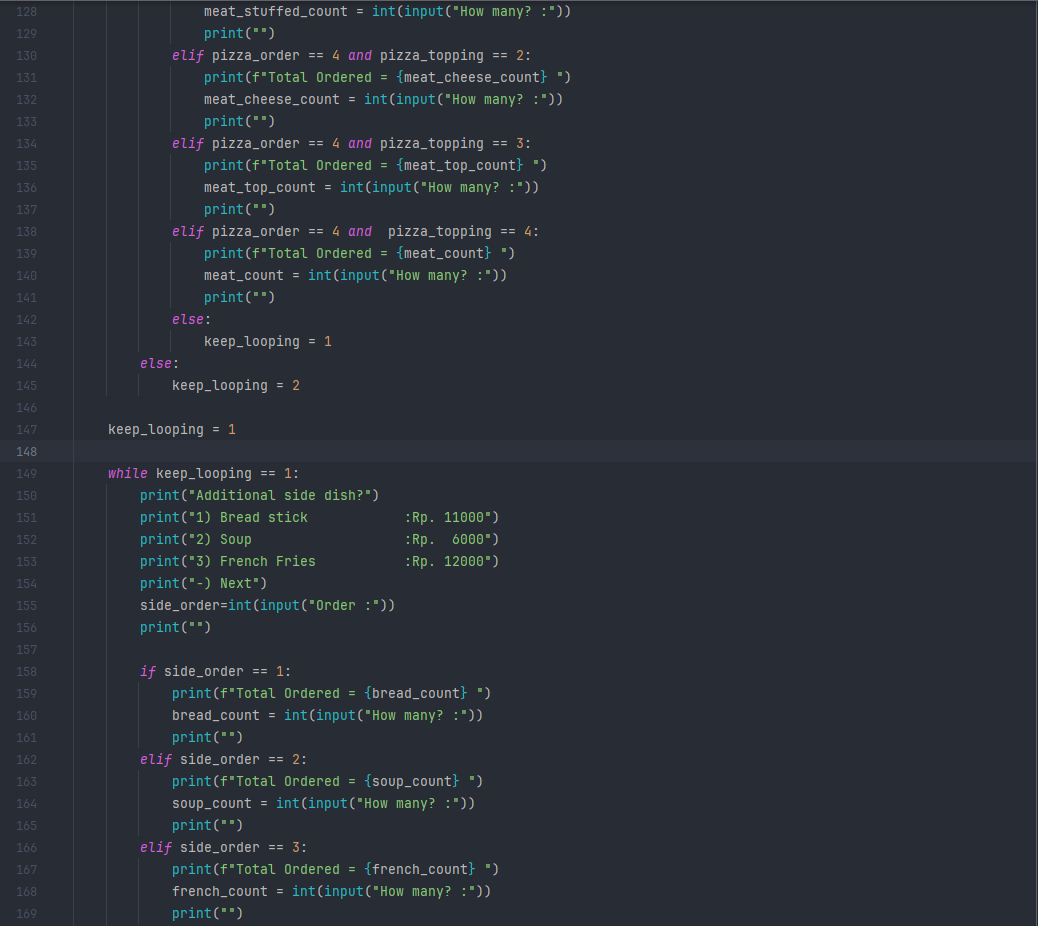
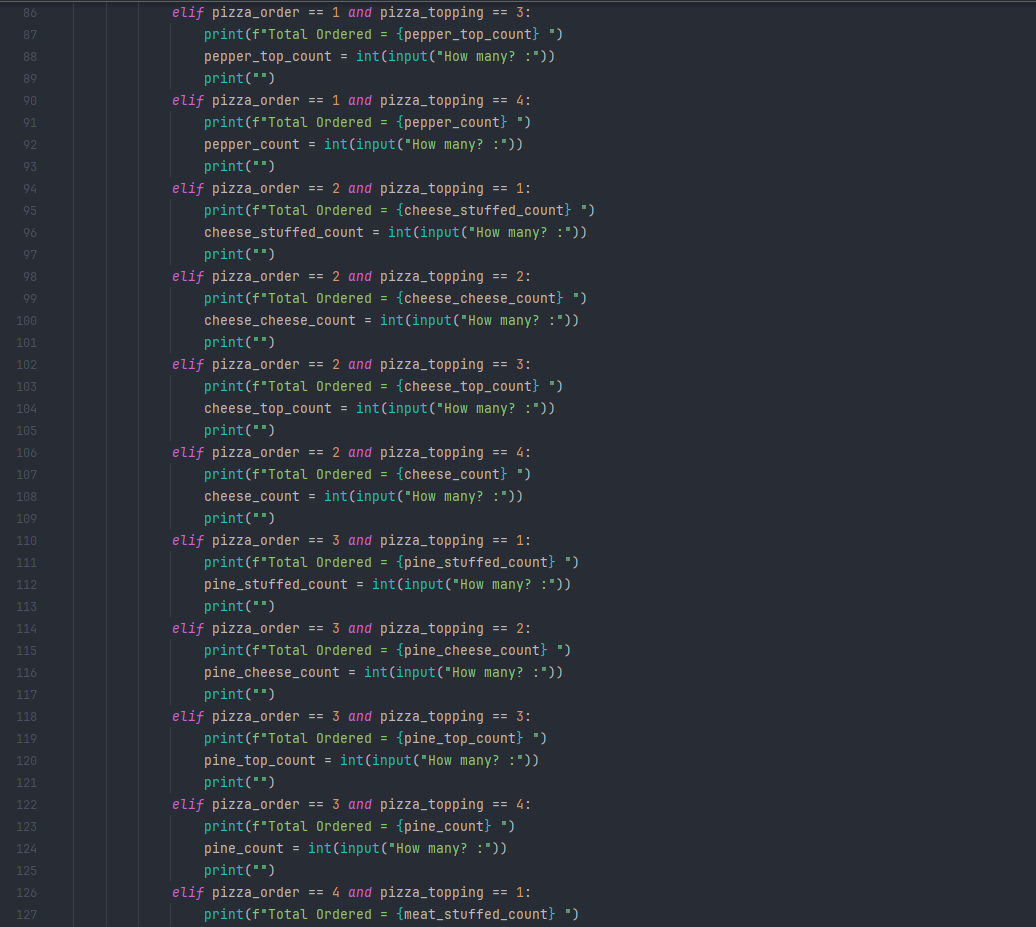
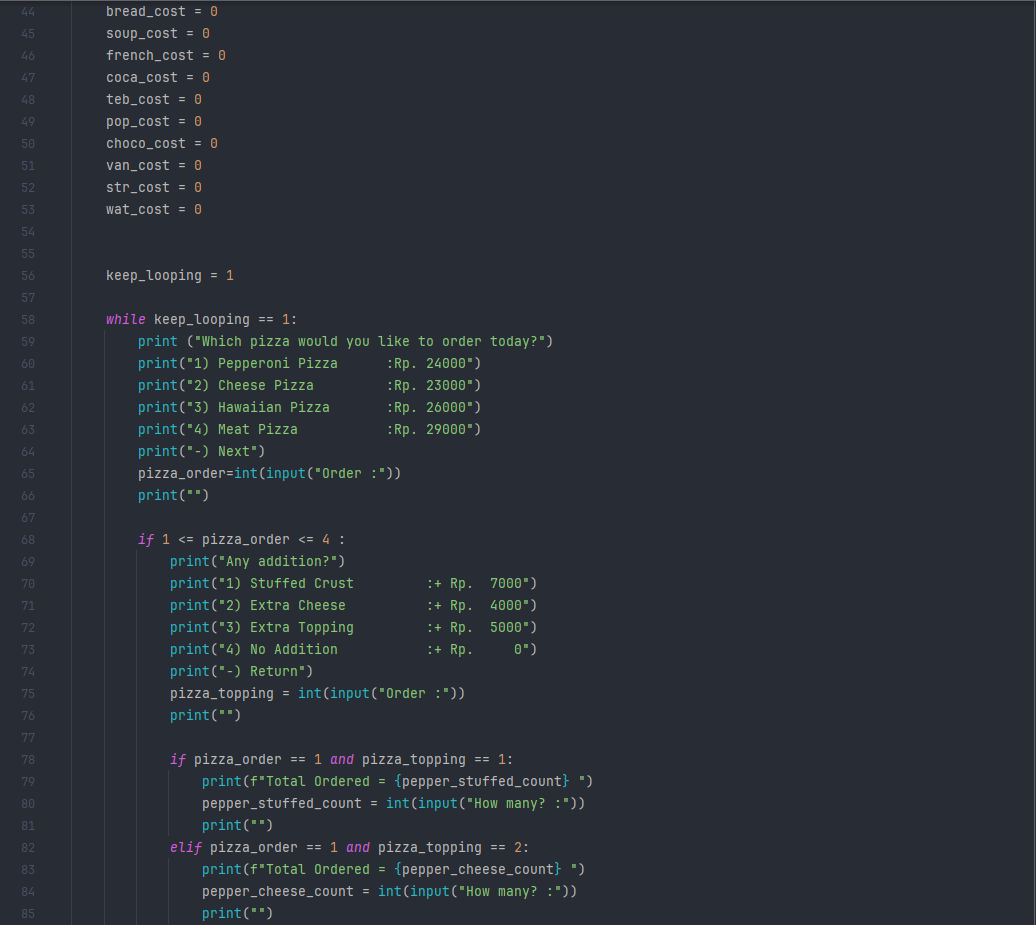
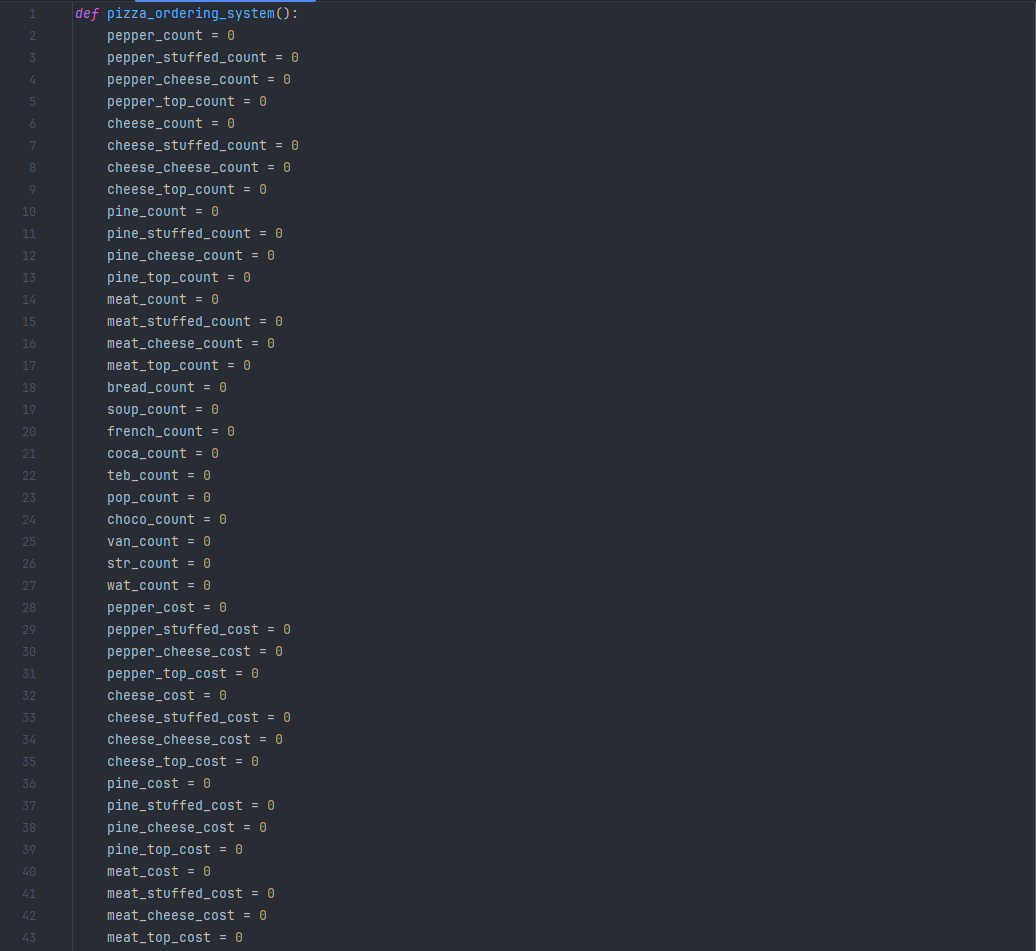
2.21. The call for the function

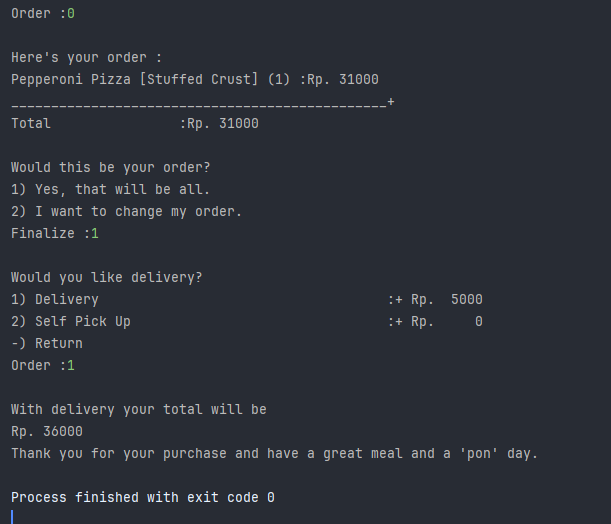
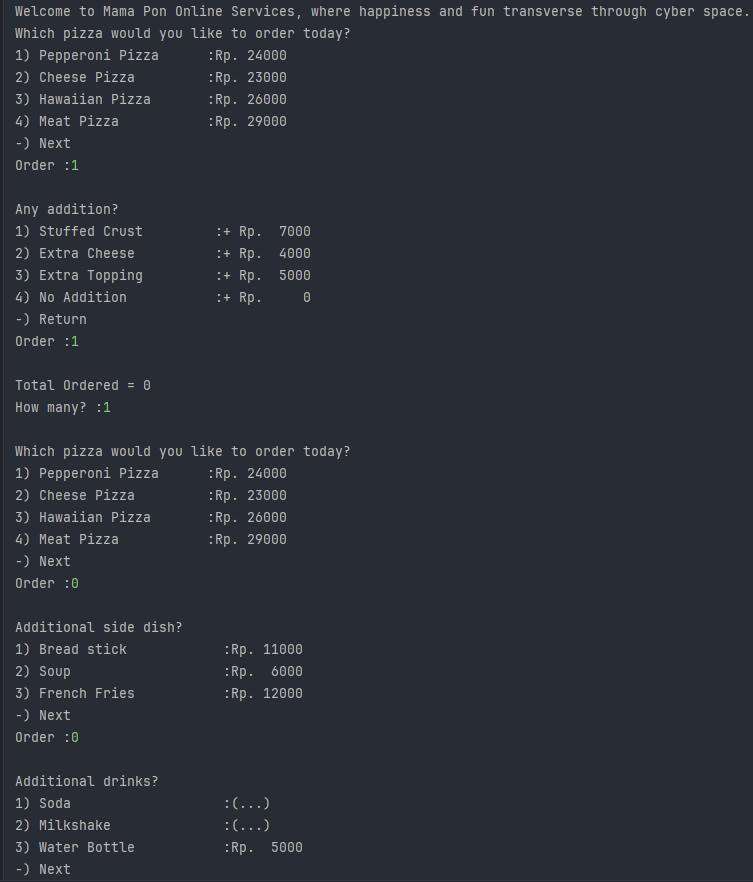
In the very bottom after everything has been defined and clean out just remain the function call, along with the greeting print for the first time running up the program. Nothing more can be said.

1. **CONCLUSION**

For the most part the code has been optimized as much as for it still make sense as a part of “if and else” lecture. There’s definitely some place to optimize it, like the pizza order having to many variable maybe could be shortened using true or false, and a different way to rerun the code outside of putting everything inside a function and rerunning the function, etc. However for just a couple week of lecture this code is more than satisfied our curious need for exploring the world of coding and we will wait for the next step for an even optimized code and program.

1. **RAW CODES**

****4.1. The entire code

4.2. Examples of the program running.