Task 0: Explain what you are doing/ going to accomplish

Will be the ability to see how many pieces of food have been sold

Task 1: Sketch interface design

*Draft a rough design for the interface that allows the user to trigger functionality in task 1, while also annotating where the information in task 2 will be displayed. Create another sketch listing the interface widgets used to create the interface.*

Task 2: Identify any classes required

*Explain what the class will represent, plus listing what information will be stored in the class and any functions the class will have.*

Task 3: Identify information to be displayed

*What information will the interface need to display to the user?*

How much of each item has been sold

Task 4: Identify user inputs

*What program functions can the user trigger through the interface?*

Task 5: Identify any constants or existing data if required

Uses existing function of product page

Task 6: Identify indexed data structures

Task 7: Determine what calculations are necessary

*Write out the calculations the program will have to compute.*

Add how much stock you have bought to total amount of stock sold.

Task 8: Develop a modular structure for your program

*Describe any functions that the computer program will have, identifying any sub-functions where required.*

Uses existing function

Add new variable to my object \_\_init\_\_ called amount\_sold

Task 9: Define the functions identified

*Describe the functions for both the main program and any classes in terms of input and/or output where required. You may choose to do this with flow charts or pseudo-code (not Python code!). Add in additional steps or explanations using sequential, conditional, iterative statements where required. Identify global and/or local variables.*

SET self.amount\_sold TO amount\_sold

found\_item.amount\_stocks + 1

Task 10: Address any relevant implications such as usability, functionality, legal/ethical requirements.

Task 11: Document test cases for testing the program

*Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.*

Task 12: Refine the plan

*Note any modifications here when iterating through the development cycles.*

Task 13: Document testing

*Show screenshots of your program working with descriptions of each image. These images should test the tests cases listed above.*

Tried purchasing each item. Works as intended. The amount\_sold increases by how much food was sold.

When I restock my food items it does not change the amount\_sold variable. Works as intended.

Task 14: Evaluation

*How did your version turn out?*

This version worked as intended. It was just a small change which entailed adding another variable to my Food\_Item object called amount\_sold. Whenever someone purchases stock, it is added to the total amount of that item sold. I wondered if restocking would have any effect on this variable but it did not affect it at all. This worked perfectly and required no fixes.