Data Structure and Algorithm

Laboratory Activity No. 4

Arrays

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Hermosura, Leigh B. | Engr. Maria Rizette H. Sayo |

August 19, 2025

# Objectives

Introduction

Array, in general, refers to an orderly arrangement of data elements. Array is a type of data structure that stores data elements in adjacent locations. Array is considered as linear data structure that stores elements of same data types. Hence, it is also called as a linear homogenous data structure.

This laboratory activity aims to implement the principles and techniques in:

* Writing algorithms using Array data structure
* Solve programming problems using dynamic memory allocation, arrays and pointers

# Methods

Jenna’s Grocery

A list of grocery items

AI-generated content may be incorrect.

Jenna wants to buy the following fruits and vegetables for her daily consumption. However, she needs to distinguish between fruit and vegetable, as well as calculate the sum of prices that she has to pay in total.

Problem 1: Create a class for the fruit and the vegetable classes. Each class must have a constructor, deconstructor, copy constructor and copy assignment operator. They must also have all relevant attributes (such as name, price and quantity) and functions (such as calculate sum) as presented in the problem description above.

Problem 2: Create an array GroceryList in the driver code that will contain all items in Jenna’s Grocery List. You must then access each saved instance and display all details about the items.

Problem 3: Create a function TotalSum that will calculate the sum of all objects listed in Jenna’s Grocery List.

Problem 4: Delete the Lettuce from Jenna’s GroceryList list and de-allocate the memory assigned.

# Results

The results from the laboratory activity are presented below, along with visual representations and explanations for each output.

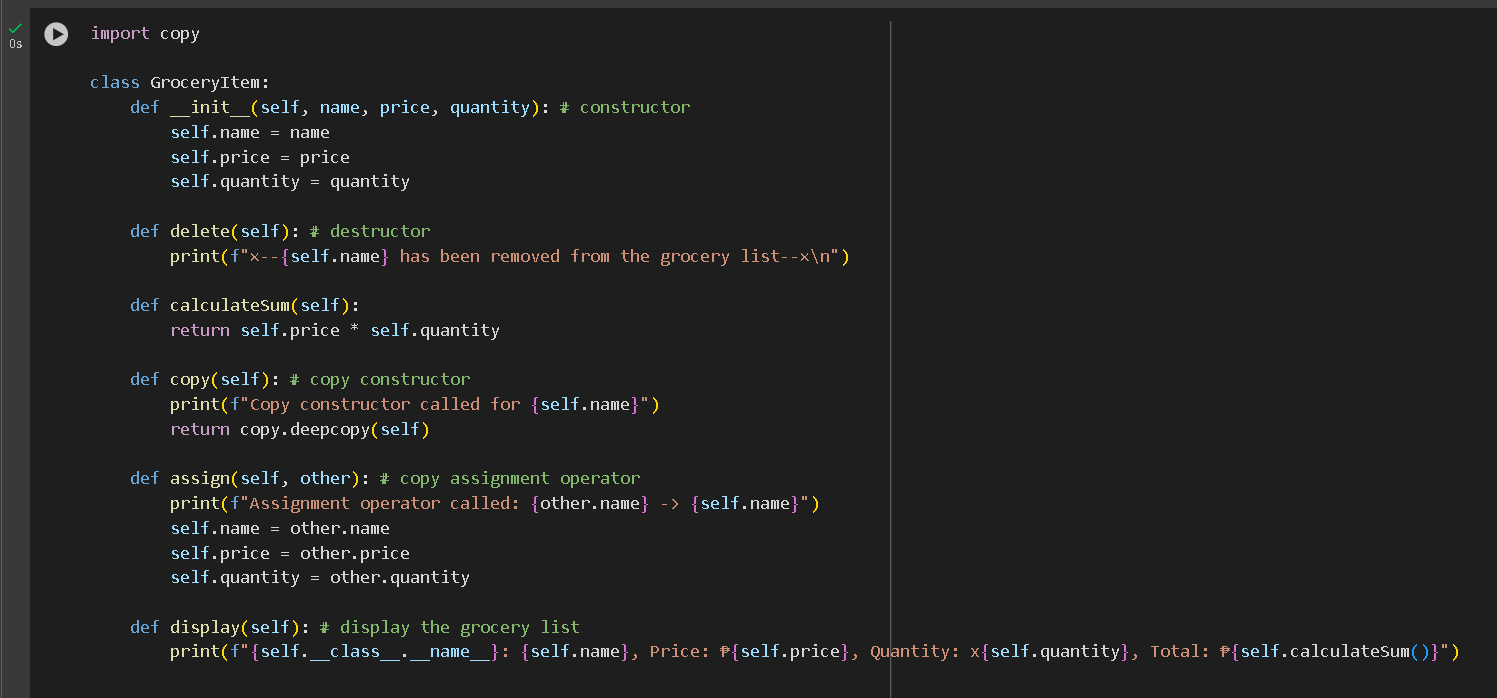


Figure 1 Inside the *GroceryItem* class

The code above shows the *GroceryItem* class with six functions inside. The first function is the constructor class where it initializes the values *name*, *price*, and *quantity*. The destructor class *delete* is called when the user wishes to remove an item to the list. The *calculateSum* function returns the result of multiplying the item’s price to its quantity, computing its total price. The *copy constructor* creates a duplicate item using the imported copy function. The *assign* function serves as a copy assignment operator, copying attributes from another *GroceryItem* object. Lastly, the *display* function prints all the items in the *GroceryList*.

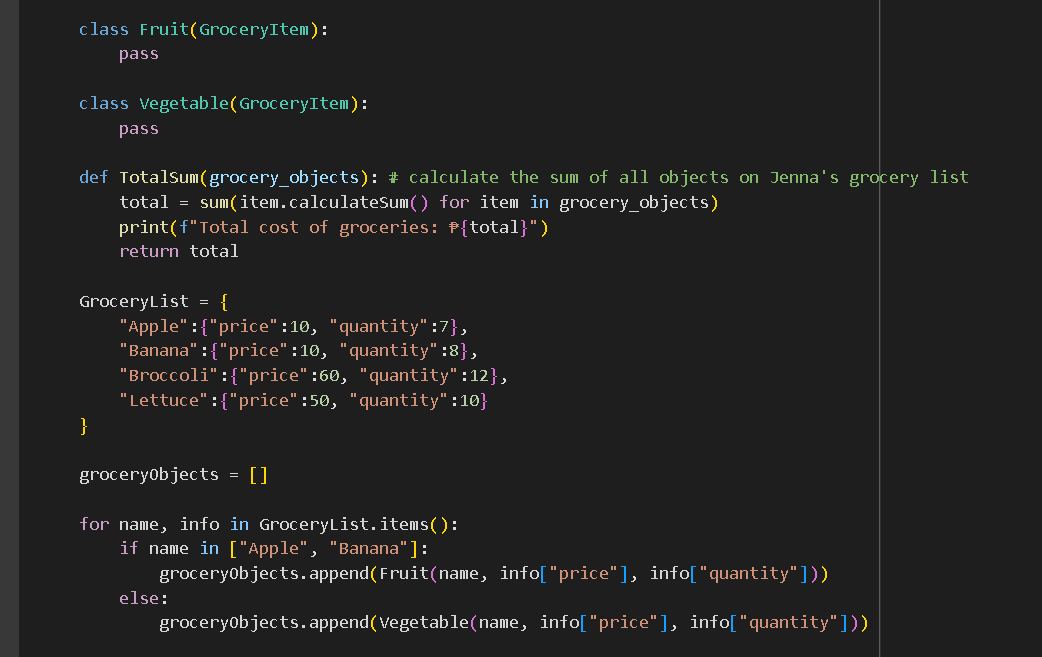


Figure 2 Continuation of the code

In Figure 2, two classes called *Fruit* and Vegetable are defined, which inherits from the *GroceryItem* class. It only contains a *pass* statement, but allows the differentiation of fruits and vegetables in the list. Outside the classes is the function *TotalSum* which computes the total cost of all items in the *GroceryList*. Below the function is a dictionary called *GroceryList* which stores the items from Jenna’s grocery list, including each item’s name, price, and quantity. An empty list called *groceryObjects* is defined to store the categorized fruit and vegetable objects.

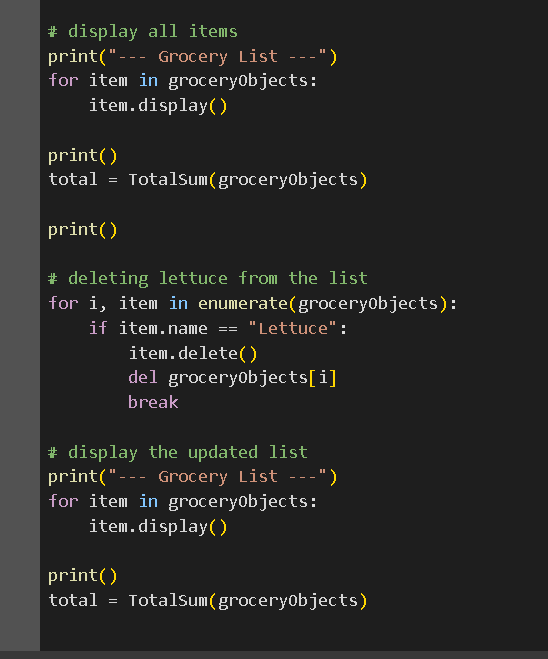


Figure 3 Displaying the output of the program

The first part displays all of the items inside the *groceryObjects* and prints their total costs. A *for* loop interates over every item, and the destructor function is called to delete the *lettuce* from the list. The updated list is then displayed, showing that the lettuce has been successfully removed.

# Conclusion

The laboratory activity successfully demonstrates the use of object-oriented programming concepts such as class inheritance, constructors, destructors, and method overloading through the GroceryItem class and its derived classes, Fruit and Vegetable. By organizing grocery items into categorized objects and manipulating them using defined functions, the program showcases how real-world problems can be modeled effectively in code. The implementation of the TotalSum function and object removal via the destructor confirms the correct functionality of each component.