UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 4

# Arrays

*Submitted by:*
Delinia, Filjohn B.

*Instructor:*
Engr. Maria Rizette H. Sayo

08, 09, 2025

## I. Objectives

Introduction

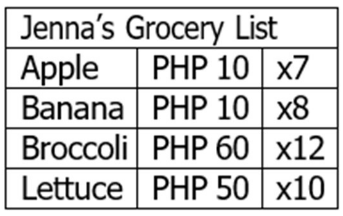
Array, in general, refers to an orderly arrangement of data elements. Array is a type of data structure that stores data elements in adjacent locations. Array is considered as linear data structure that stores elements of same data types. Hence, it is also called as a linear homogenous data structure.

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

## II. Methods

Jenna’s Grocery



| Jenna’s Grocery List | | |
|---|---|---|
| Apple | PHP 10 | x7 |
| Banana | PHP 10 | x8 |
| Broccoli | PHP 60 | x12 |
| Lettuce | PHP 50 | x10 |

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.

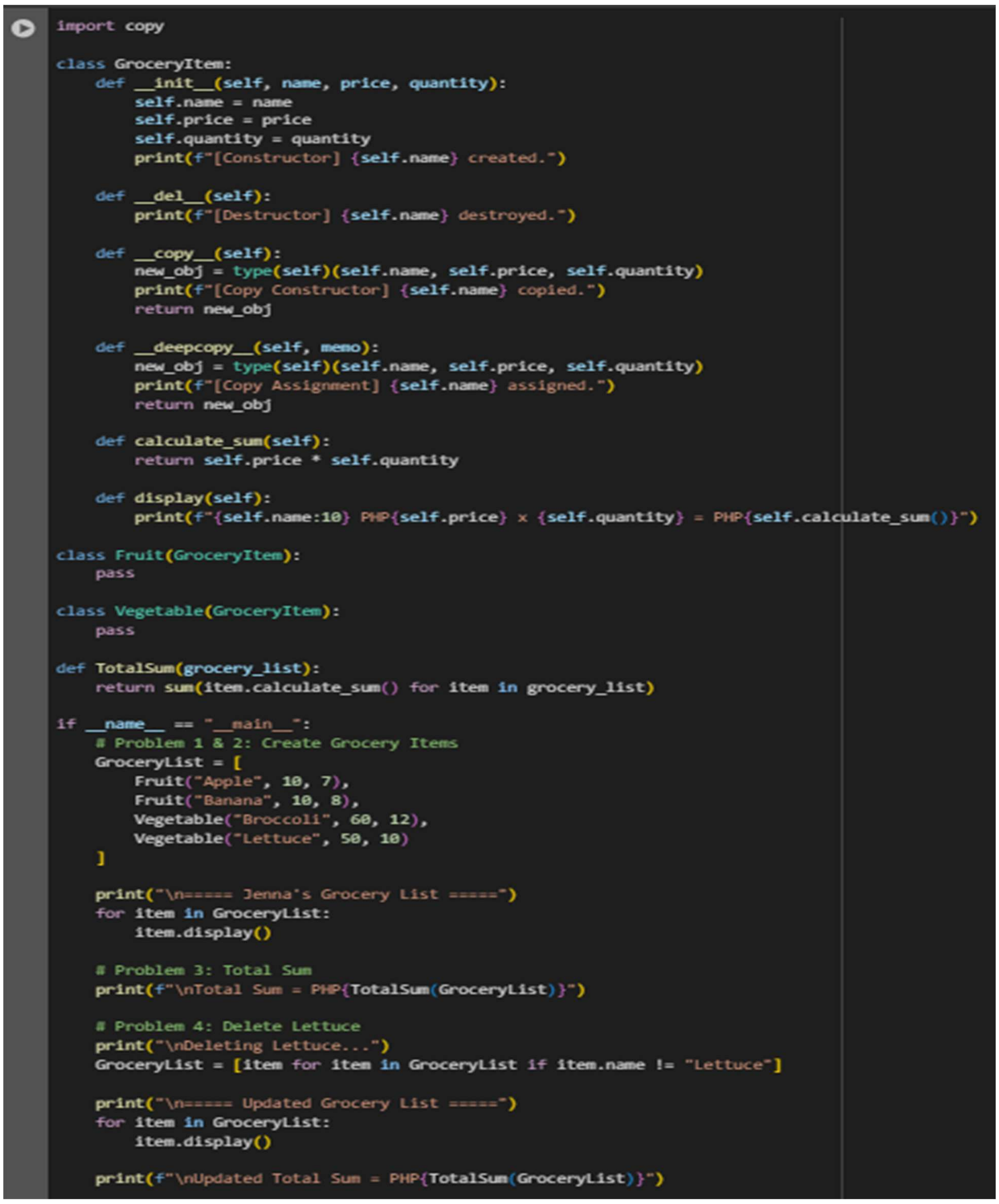
Problem 1: Create a class for the fruit and the vegetable classes. Each class must have a constructor, deconstructor, copy constructor and copy assignment operator. They must also have all relevant attributes (such as name, price and quantity) and functions (such as calculate sum) as presented in the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

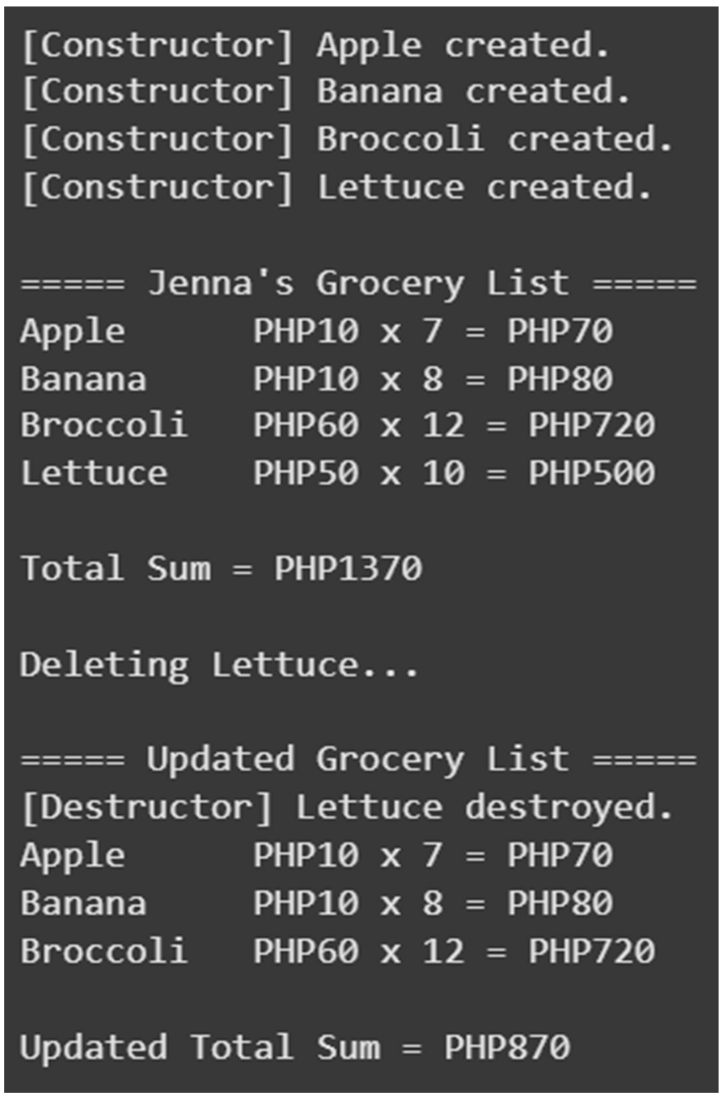
Problem 4: Delete the Lettuce from Jenna’s GroceryList list and de-allocate the memory assigned.

## III. Results



```
import copy

class GroceryItem:
    def __init__(self, name, price, quantity):
        self.name = name
        self.price = price
        self.quantity = quantity
        print(f"[Constructor] {self.name} created.")

    def __del__(self):
        print(f"[Destructor] {self.name} destroyed.")

    def __copy__(self):
        new_obj = type(self)(self.name, self.price, self.quantity)
        print(f"[Copy Constructor] {self.name} copied.")
        return new_obj

    def __deepcopy__(self, memo):
        new_obj = type(self)(self.name, self.price, self.quantity)
        print(f"[Copy Assignment] {self.name} assigned.")
        return new_obj

    def calculate_sum(self):
        return self.price * self.quantity

    def display(self):
        print(f"{self.name:10} PHP{self.price} x {self.quantity} = PHP{self.calculate_sum()}")

class Fruit(GroceryItem):
    pass

class Vegetable(GroceryItem):
    pass

def TotalSum(grocery_list):
    return sum(item.calculate_sum() for item in grocery_list)

if __name__ == "__main__":
    # Problem 1 & 2: Create Grocery Items
    GroceryList = [
        Fruit("Apple", 10, 7),
        Fruit("Banana", 10, 8),
        Vegetable("Broccoli", 60, 12),
        Vegetable("Lettuce", 50, 10)
    ]

    print("\n===== Jenna's Grocery List =====")
    for item in GroceryList:
        item.display()

    # Problem 3: Total Sum
    print(f"\nTotal Sum = PHP{TotalSum(GroceryList)}")

    # Problem 4: Delete Lettuce
    print("\nDeleting Lettuce...")
    GroceryList = [item for item in GroceryList if item.name != "Lettuce"]

    print("\n===== Updated Grocery List =====")
    for item in GroceryList:
        item.display()

    print(f"\nUpdated Total Sum = PHP{TotalSum(GroceryList)}")
```

Figure 1. Source Code



```
[Constructor] Apple created.
[Constructor] Banana created.
[Constructor] Broccoli created.
[Constructor] Lettuce created.

===== Jenna's Grocery List =====
Apple      PHP10 x 7 = PHP70
Banana     PHP10 x 8 = PHP80
Broccoli   PHP60 x 12 = PHP720
Lettuce    PHP50 x 10 = PHP500

Total Sum = PHP1370

Deleting Lettuce...

===== Updated Grocery List =====
[Destructor] Lettuce destroyed.
Apple      PHP10 x 7 = PHP70
Banana     PHP10 x 8 = PHP80
Broccoli   PHP60 x 12 = PHP720

Updated Total Sum = PHP870
```

**Explanation:**

The program uses classes to manage items such as fruits and vegetables, with constructors and destructors showing the lifecycle of each object. Copy methods are included to simulate C++ behavior in Python, allowing for object copying and assignment. All grocery items are stored in the GroceryList array, and the TotalSum function calculates the total bill by summing the cost of all items. Finally, deleting Lettuce from the list updates the grocery collection and recalculates the total amount Jenna needs to pay.

Figure 2. Output

Source Code link:
https://colab.research.google.com/drive/1DuvtfsBW5aYlhSlIiFGt_o53hvOHqcIN#scrollTo=jXt6Ul8-N7P7&line=61&uniqifier=1

## IV. Conclusion

In conclusion, the program demonstrates how data structures and object-oriented programming can be combined to solve practical problems. By using classes to represent fruits and vegetables, the program highlights encapsulation of data and operations, while constructors and destructors emphasize object lifecycle management. The GroceryList serves as a simple array (list) structure for storing and accessing multiple items efficiently. The TotalSum function illustrates algorithmic processing by iterating through the list to compute the total cost, while the deletion of Lettuce shows how elements can be dynamically removed and the structure updated.

## References

[1] Co Arthur O.. "University of Caloocan City Computer Engineering Department Honor Code," UCC-CpE Departmental Policies, 2020.