UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 5

Implementation of Arrays

Submitted by:
Poliño, Justine

Instructor:
Engr. Maria Rizette H. Sayo

August 18, 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:

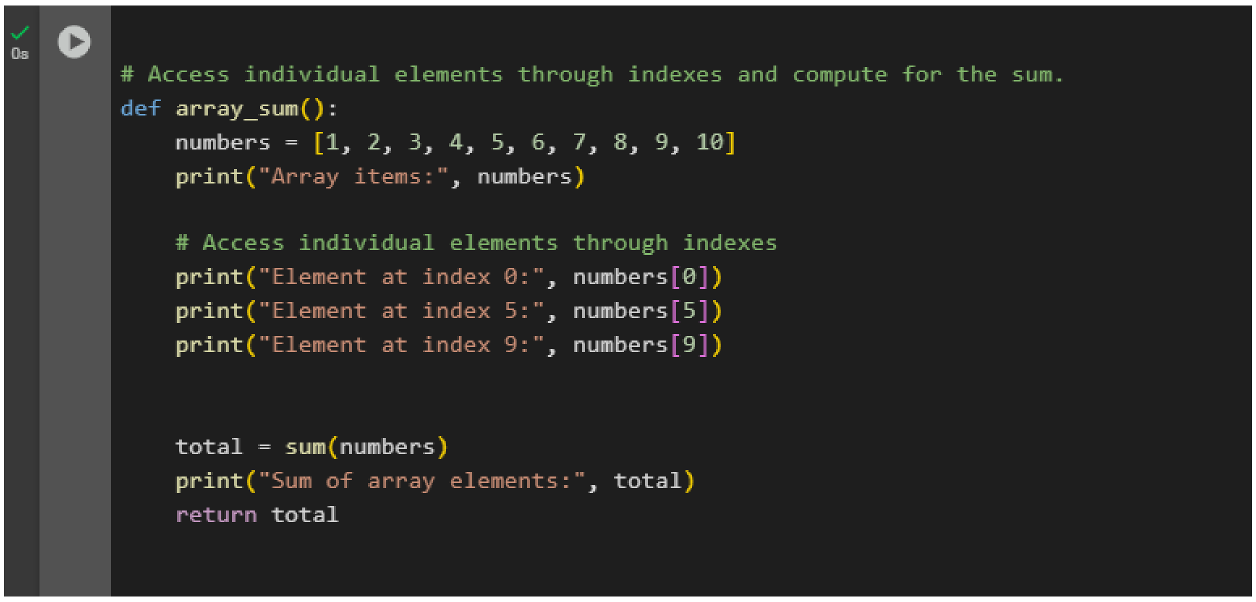
- Writing algorithms using Array data structure
- Writing a python program that can implement Array data structure

II. Methods

- Write a Python program to create an array of 10 integers and display the array items. Access individual elements through indexes and compute for the sum.
- Write a Python program to append a new item to the end of the array. Original array: numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- Write a Python program to insert a new item before the second element in an existing array. Original array: numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
- Write a Python program to reverse the order of the items in the array. Original array: numbers = [5, 4, 3, 2, 1]
- Write a Python program to get the length of the array. Original array: numbers = [5, 4, 3, 2, 1]

III. Results

In this laboratory, numerous Python programs were written to demonstrate how to build the Array data structure. An array is a data structure that arranges elements of the same kind in a succession. The elements are stored in continuous memory locations, resulting in a linear structure. Because of this, an array is also known as a linear and homogenous data structure.



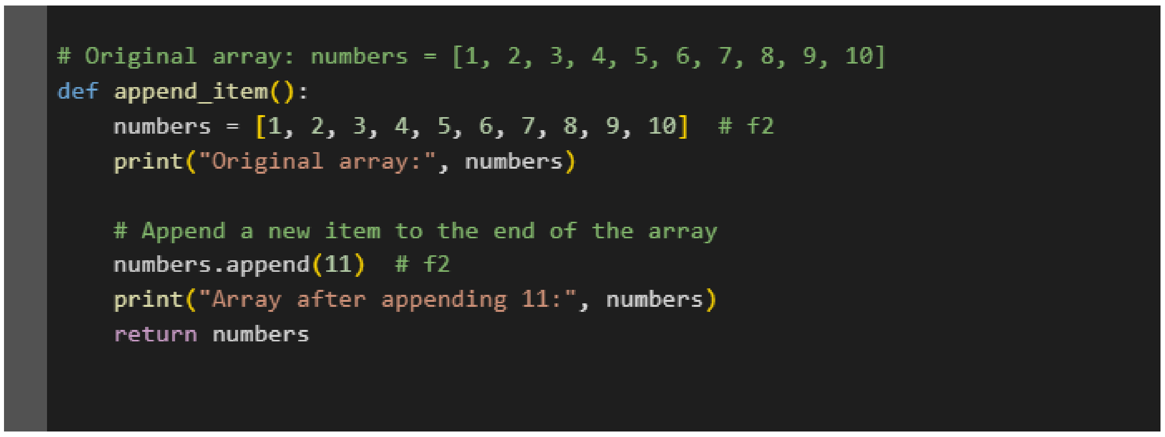
```python
# Access individual elements through indexes and compute for the sum.
def array_sum():
    numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    print("Array items:", numbers)

    # Access individual elements through indexes
    print("Element at index 0:", numbers[0])
    print("Element at index 5:", numbers[5])
    print("Element at index 9:", numbers[9])


    total = sum(numbers)
    print("Sum of array elements:", total)
    return total
```

Figure 1 Screenshot of 1st program

Figure 1 demonstrates a program that creates an array with 10 integers and displays all its elements. It also shows how to access each item using its index and then calculates the total sum of all the numbers in the array.

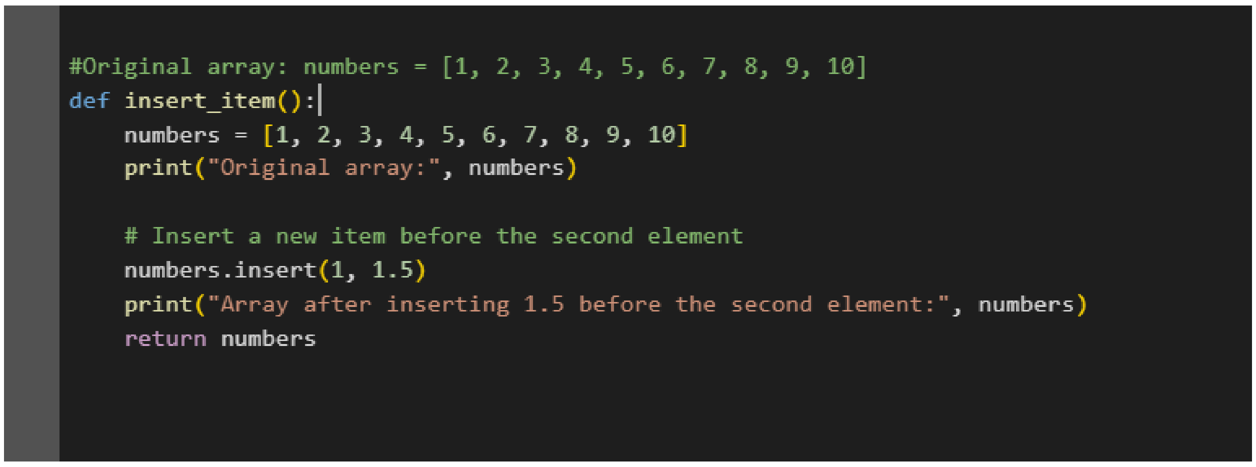


```python
# Original array: numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
def append_item():
    numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  # f2
    print("Original array:", numbers)

    # Append a new item to the end of the array
    numbers.append(11)  # f2
    print("Array after appending 11:", numbers)
    return numbers
```

Figure 2 Screenshot of 2nd program

Figure 2 demonstrates a program that appends a new item (11) to the end of an existing array. It shows the original array and the modified array after the append operation.

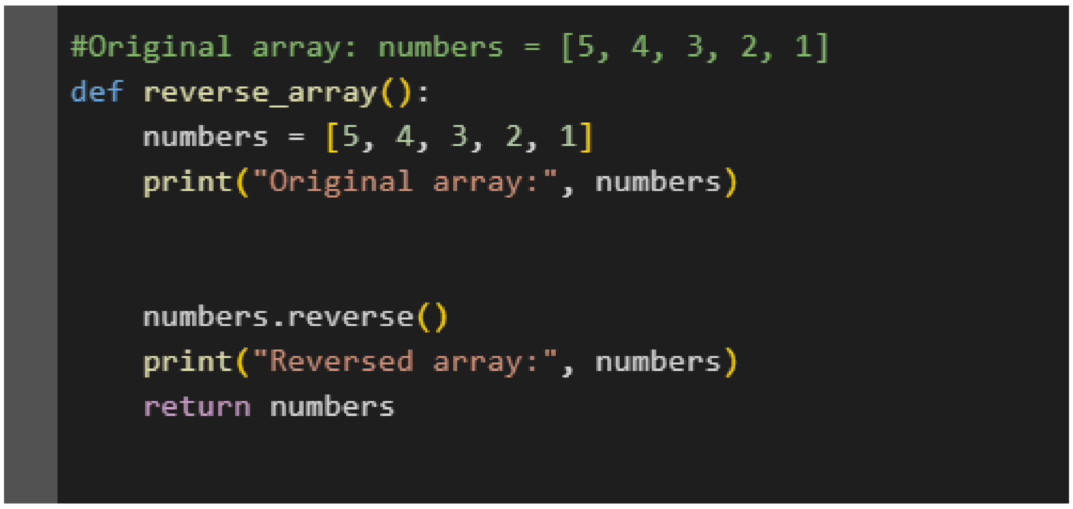


```python
#Original array: numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
def insert_item():
    numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    print("Original array:", numbers)

    # Insert a new item before the second element
    numbers.insert(1, 1.5)
    print("Array after inserting 1.5 before the second element:", numbers)
    return numbers
```

Figure 3 Screenshot of 3rd program

Figure 3 demonstrates a program that inserts a new item (1.5) before the second element in an array. It shows how the insert operation shifts existing elements to accommodate the new value.

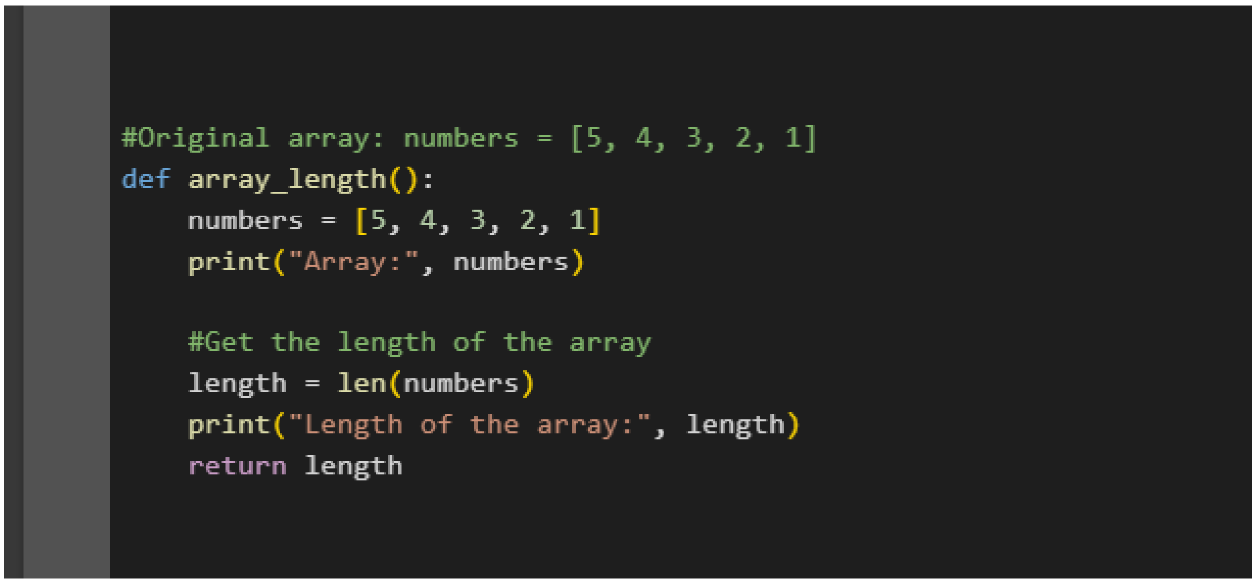


```python
#Original array: numbers = [5, 4, 3, 2, 1]
def reverse_array():
    numbers = [5, 4, 3, 2, 1]
    print("Original array:", numbers)


    numbers.reverse()
    print("Reversed array:", numbers)
    return numbers
```

Figure 4 Screenshot of 4th program

Figure 4 demonstrates a program that reverses the order of elements in an array. It transforms the array [5, 4, 3, 2, 1] into [1, 2, 3, 4, 5] using the reverse() method.



```python
#Original array: numbers = [5, 4, 3, 2, 1]
def array_length():
    numbers = [5, 4, 3, 2, 1]
    print("Array:", numbers)

    #Get the length of the array
    length = len(numbers)
    print("Length of the array:", length)
    return length
```

Figure 5 Screenshot of 5th program

Figure 5 demonstrates a program that calculates and displays the length of an array. It uses the len() function to determine the number of elements in the array [5, 4, 3, 2, 1].

IV. Conclusion

In conclusion, the activities showed different ways of implementing arrays in Python. We created arrays, accessed elements, computed the sum, added new items, inserted elements at specific positions, reversed the order, and found the length of the array. Through these tasks, we learned the basic operations that make arrays an important data structure for storing and managing data efficiently.

References

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