UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 5

Implementation of Arrays

Submitted by:
Disomnong, Jalilah M.

Instructor:
Engr. Maria Rizette H. Sayo

8-16-2025

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:

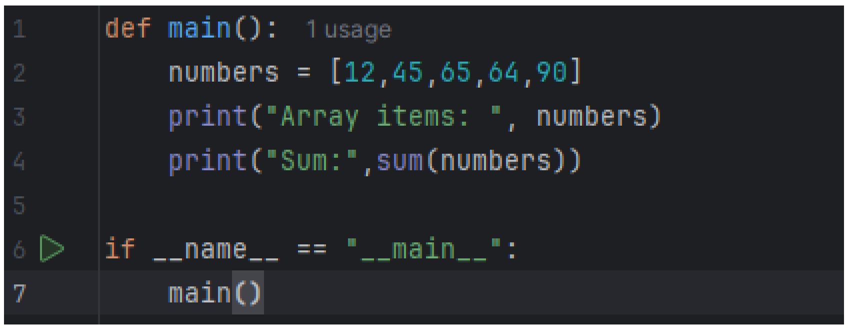
- Writing algorithms using Array data structure
- Writing a python program that can implement Array data structure

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]

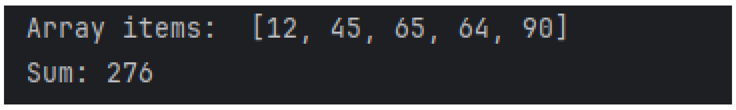
III. Results



```python
def main():  1 usage
    numbers = [12,45,65,64,90]
    print("Array items: ", numbers)
    print("Sum:",sum(numbers))

if __name__ == "__main__":
    main()
```

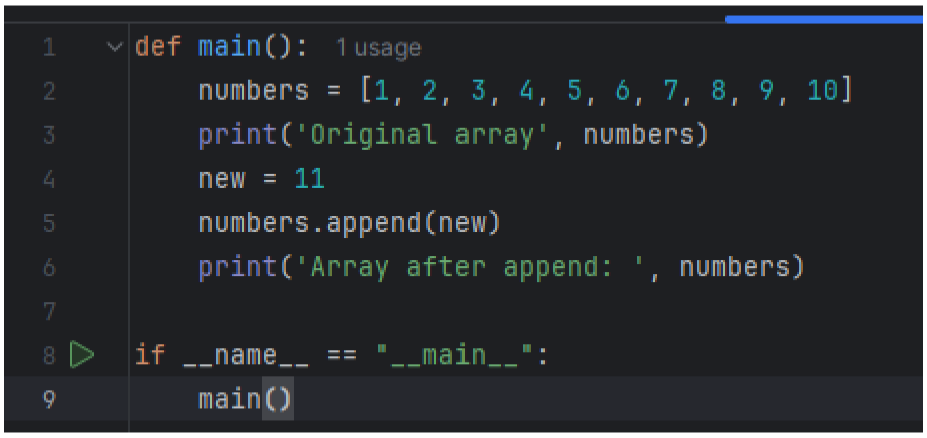
Figure 1 Screenshot of source code refers to this: create.py



```
Array items:  [12, 45, 65, 64, 90]
Sum: 276
```

Figure 2 Output refers to this: create.py

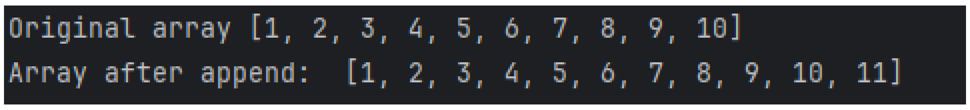
In figure 1, a simple Python program is written to demonstrate how to work with lists and perform basic operations such as printing elements and calculating their sum. The main() function initializes a list named numbers containing five integers: 12, 45, 65, 64, and 90. It then uses the print() function to display the elements of the list with the label "Array items:". After that, the built-in sum() function is used to compute the total of the list's elements, which is then printed.



```
def main():  1 usage
    numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    print('Original array', numbers)
    new = 11
    numbers.append(new)
    print('Array after append: ', numbers)

if __name__ == "__main__":
    main()
```

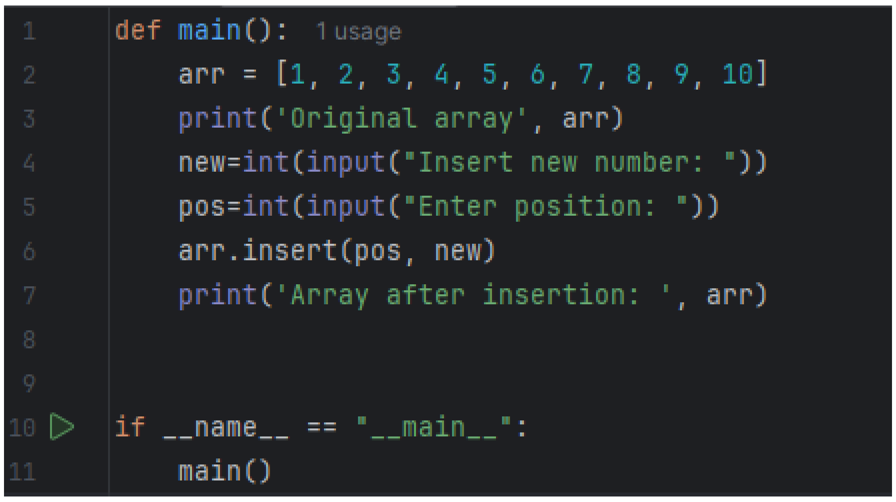
Figure 3 Screenshot of source code refers to this: append.py



```
Original array [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Array after append:  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

Figure 4 Output

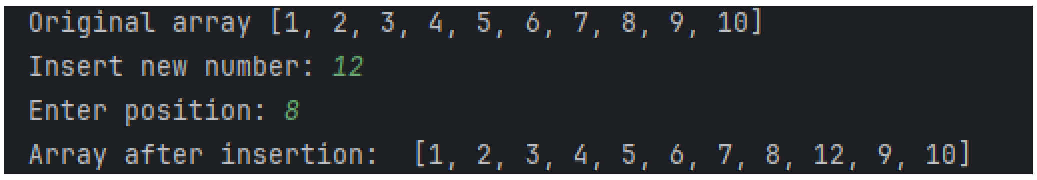
In figure 3, a Python program is designed to demonstrate how to append a new element to an existing list. Inside the main() function, a list named numbers is initialized with the integers from 1 to 10. The program first prints this original list using the print() function with the label "Original array". Next, a new integer value, 11, is stored in the variable new, and the append() method is used to add this value to the end of the numbers list. After appending, the updated list is printed with the label "Array after append" to show the change.



```
def main():  1 usage
    arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    print('Original array', arr)
    new=int(input("Insert new number: "))
    pos=int(input("Enter position: "))
    arr.insert(pos, new)
    print('Array after insertion: ', arr)


if __name__ == "__main__":
    main()
```

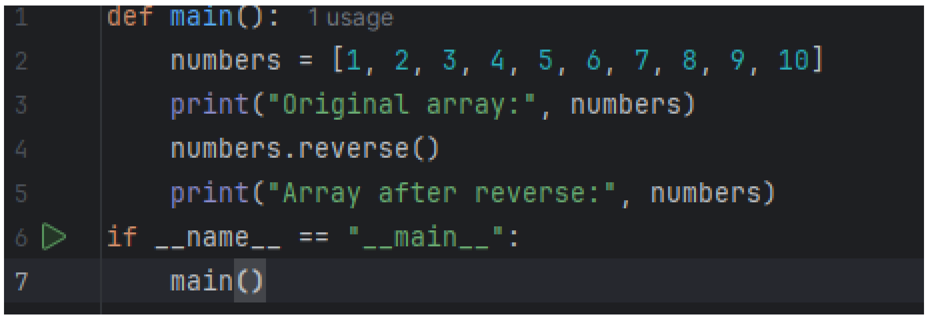
Figure 5 Screenshot of source code refer to this: insert.py



```
Original array [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Insert new number: 12
Enter position: 8
Array after insertion:  [1, 2, 3, 4, 5, 6, 7, 8, 12, 9, 10]
```

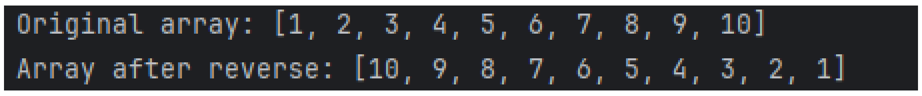
Figure 6 Output

In figure 5, the code demonstrates how to insert an element into a specific position in a list using user input. It begins with a predefined list of integers from 1 to 10. The user is then prompted to enter a new number and the position where that number should be inserted. The insert() method is used for this purpose, which places the new element at the specified index while shifting the other elements to the right. This allows the list to grow without overwriting any existing values. The program also includes a print statement before and after the insertion to clearly show how the list has changed.



```python
def main():  1 usage
    numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    print("Original array:", numbers)
    numbers.reverse()
    print("Array after reverse:", numbers)
if __name__ == "__main__":
    main()
```

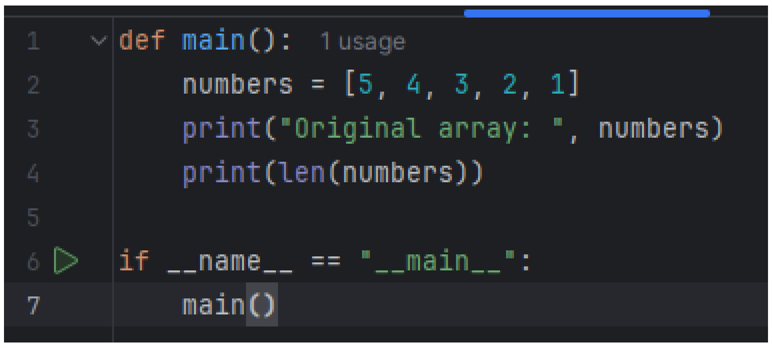
Figure 8 Screenshot of source code refer to this: reverse.py



```
Original array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Array after reverse: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Figure 7 Output

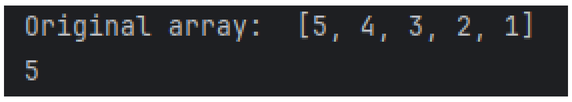
In figure 7, the Python program demonstrates the use of the reverse() method to manipulate the order of elements within a list. Initially, a list named numbers is defined with sequential integers from 1 to 10, representing a basic data structure commonly used in programming. The original list is printed to provide a reference point for comparison. The core operation of the program involves reversing the list in place using the reverse() method, which efficiently alters the sequence of elements without creating a new list object.



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

if __name__ == "__main__":
    main()
```

Figure 9 Screenshot of source code refer to this: length.py



```
Original array:  [5, 4, 3, 2, 1]
5
```

Figure 10 Output

In the figure 9, the Python program demonstrates how to determine the number of elements in a list using the built-in len() function. Within the main() function, a list named numbers is initialized with integers arranged in descending order from 5 to 1. The original list is first printed to provide context for the user. Following this, the len() function is applied to the list, which returns the total count of elements it contains. In this case, the output will be 5, reflecting the five elements present in the list.

Conclusion

In this laboratory activity, we demonstrated the fundamental concepts and practical implementation of arrays using Python programming. Through a series of carefully structured exercises, we explored the creation of arrays, accessing individual elements, appending new items, inserting elements at specified positions, reversing the order of elements, and determining the length of the array. These tasks highlighted the application of Python’s built-in list methods in manipulating linear, homogeneous data structures. The activity provided clearer insight into how arrays function both conceptually and programmatically, further reinforcing our understanding of array behavior and manipulation techniques.

References

[1] “PyCharm: The only Python IDE you need,” JetBrains, Jun. 02, 2021.
https://www.jetbrains.com/pycharm/#