Namal University, Mianwali
Department of Electrical Engineering
Course Title: EE-253 Data Structures and Algorithms

LAB MANUAL

Lab 4 Arrays in Python

In previous lab, we have studied how to define a class, constructing object, accessing members of objects, and self-parameter. We have also studied the concept of files (read, write, append, etc.) in python.

1. Lab Objectives

The objective of this lab is to introduce students to the concept of array in python. In this lab students will enable the concepts of defining an array, types of arrays, accessing the elements of array and operations which can be performed on array.
Students will be provided with examples for each objective, followed by performing lab tasks.

2. Lab Outcomes

- CLO:1 Recognize the usage of fundamental Data structure using Python Programming Language.
- CLO:3 Demonstrate solution to real life problems using appropriate data structures.

3. Equipment

- Software
    - IDLE (Python 3.11)

4. Instructions

1. This is an individual lab. You will perform the tasks individually and submit a report.
2. Some of these tasks (marked as ‘Example’) are for practice purposes only while others (marked as ‘Task’) have to be answered in the report.
3. When asked to display an output in the task, either save it as jpeg or take a screenshot, in order to insert it in the report.
4. The report should be submitted on the given template, including:
    a. Code (copy and pasted, NOT a screenshot)
    b. Output figure (as instructed in 3)
    c. Explanation where required
5. The report should be properly formatted, with easy to read code and easy to see figures.
6. Plagiarism or any hint thereof will be dealt with strictly. Any incident where plagiarism is caught, both (or all) students involved will be given zero marks, regardless of who copied whom. Multiple such incidents will result in disciplinary action being taken.
7. Late submission of report is allowed within 03 days after lab with 20% deduction of marks every day.
8. You have to submit report in pdf format (Reg.X_DSA_LabReportX.pdf).

5. Background

Arrays

Array is a container which can hold a fix number of items and these items should be of the same type. Most of the data structures make use of arrays to implement their algorithms. Following are the important terms to understand the concept of Array.

- **Element**− Each item stored in an array is called an element.
- **Index** − Each location of an element in an array has a numerical index, which is used to identify the element.

Following are the basic operations supported by an array.

- **Traverse** − print all the array elements one by one.
- **Insertion** − Adds an element at the given index.
- **Deletion** − Deletes an element at the given index.
- **Search** − Searches an element using the given index or by the value.
- **Update** − Updates an element at the given index.

Array is created in Python by importing array module to the python program. Then the array is declared as shown below.

import array
from array import*
ArrayName = array('Type Code', [Initializers])

Type codes are the codes that are used to define the type of value the array will hold. Some common type codes used are:

Code	C Data Type	Typical Number of Bytes
'b'	signed char	1
'B'	unsigned char	1
'u'	Unicode char	2 or 4
'h'	signed short int	2
'H'	unsigned short int	2
'i'	signed int	2 or 4
'I'	unsigned int	2 or 4
'l'	signed long int	4
'L'	unsigned long int	4
'f'	float	4
'd'	float	8

Before going in detail of different operations of array, let's take a simple example of array using python.

Source Code:

```
import array
from array  import*
new1 = array('i', [4,5,6])
for a in new1:
   print(a)
```

Output:

```
4
5
6
```

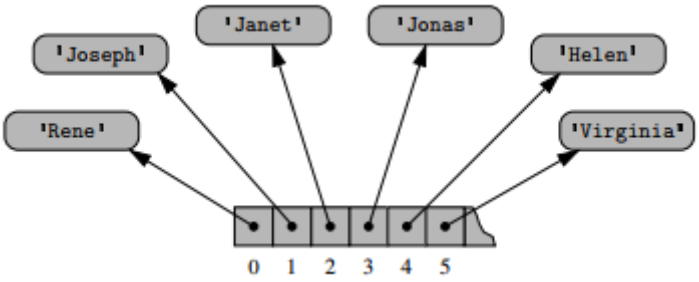
Types of Array:

- Referential Array
- Compact Array
- Dynamic Array

Referential Array:

Python represents a list or tuple instance using an internal storage mechanism of an array of object references that is **Referential array**.

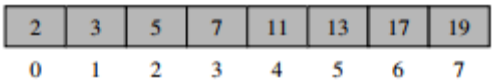
For example, to store all the employees name in an array what we will do is we will initialize an array but instead of the values or char in it we will reference object of the employee's names in each cell as show in below figure.



Compact Array:

Compact arrays over referential ones is no extra overhead from storing memory references, we directly store the primary data. Primary support for **compact arrays** is in a module named array. That module defines a class, also named array, providing compact storage for arrays of primitive data types.

Here is a simple example of compact array as shown in figure:



Dynamic Array:

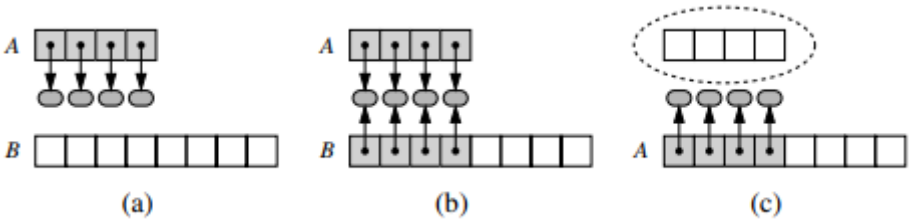
When creating a compact array, we specify the precise size of that array so that the computer knows what memory to allocate. Because it allocates neighboring memory area to other arrays.

In practice we can't always know the size of our arrays, they might expand over time. As a remedy to this problem, python provides as abstraction called **dynamic array**.

The key is to provide means to grow the array A that stores the elements of a list. Of course, we cannot actually grow that array, as its capacity is fixed. If an element is appended to a list at a time when the underlying array is full, we perform the following steps:

1. Allocate a new array B with larger capacity.
2. Set B[i] = A[i], for i = 0,...,n−1, where n denotes current number of items.
3. Set A = B, that is, we henceforth use B as the array supporting the list.
4. Insert the new element in the new array.

This can be seen from the below figure.



Example 1: Write a python program which take an array of three different models of car. Print the array, add the new element, remove existing one, update any element.

Source code:

```
cars = ["Ford", "Volvo", "BMW"]
print(cars)
cars.append("Honda")
print(cars)
cars.pop(1)
print(cars)
cars[0] = "Toyota"
print(cars)
```

Output:

['Ford', 'Volvo', 'BMW']
['Ford', 'Volvo', 'BMW', 'Honda']
['Ford', 'BMW', 'Honda']
['Toyota' , 'BMW', 'Honda']

Example 2: Write a python program which create an array of five integers. Print the array with the help of traversing, add the new element, remove existing one, update any element.

Source code:

```
import array
from array  import*
new1 = array('i', [4,5,6,7,8,9])
#print the array by traversing
print("Array before insertion")
for a in new1:
   print(a)
#Add new element
new1.append(10)
print("Array after insertion")
for a in new1:
   print(a)
#Remove an element
new1.pop(3)
print("Array after removing")
for a in new1:
   print(a)
#Update an element
new1[3]=11
print("Array after updating")
for a in new1:
   print(a)
```

Output:

Array before insertion
4
5
6
7
8
9
Array after insertion
4
5
6
7
8
9
10
Array after removing
4
5
6
8
9
10
Array after updating
4
5
6
11
9
10

Example 3: Write a python program which create a dynamic array using ctypes library. (Book example page 196)

Source code:

```
import ctypes # provides low-level arrays
class DynamicArray:
  '''A dynamic array class akin to a simplified Python list.'''
  def __init__(self):
    '''Create an empty array.'''
    self._n = 0 # count actual elements
    self._capacity = 1 # default array capacity
    self._A = self._make_array(self._capacity) # low-level array
  def __len__(self):
    '''Return number of elements stored in the array.'''
    return self._n
  def __getitem__(self, k):
    '''Return element at index k.'''
    if not 0 <= k < self._n:
      raise IndexError( 'invalid index ')
```

```
        return self._A[k] # retrieve from array
    def append(self, obj):
        '''Add object to end of the array.'''
        if self._n == self. capacity: # not enough room
            self._resize(2*self._capacity) # so double capacity
            self._A[self._n] = obj
            self._n += 1
    def _resize(self, c): # nonpublic utitity
        '''Resize internal array to capacity c.'''
        B = self._make_array(c) # new (bigger) array
        for k in range(self._n): # for each existing value
            B[k] = self._A[k]
            self._A = B # use the bigger array
            self._capacity = c
    def _make_array(self, c): # nonpublic utitity
        '''Return new array with capacity c.'''
        return (c*ctypes.py_object)( ) # see ctypes documentation
```

Output:

5,6,7
(5, 6, 7)
8,9,10,11,14,15
(8, 9, 10, 11, 14, 15)

6. Lab Tasks:

Task 1: Write a Python program to input a referential array from user using for loop and find the sum of all elements and average in an array. You have to print array, sum and average of array.

Task 2: Write a Python program to find the smallest and the largest element in a referential array. You have to print array, smallest and largest elements of array.

Task 3: Write a Python program to take a compact array from user, reverse it and sort it in ascending order. You have to print array, reversed and sorted array.
You can import array library.
You can use built-in functions reverse () to reverse and sort () to sort an array.

Task 4: Write a Python program that input a dynamic array using dynamic array class. Make a function that returns a new array which contains only the unique elements of previous array. You have to print both arrays.

Lab Evaluation Rubrics							
Domain	**CLOs/ Rubric**	**Performance Indicator**	**Unsatisfactory 0-2**	**Marginal 3-5**	**Satisfactory 6-8**	**Exemplary 9-10**	**Allocated Marks**
Psychomotor	**CLO:1 R2**	Implementation with Results **(P)**	Does not try to solve problems. Many mistakes in code and difficult to comprehend for the instructor. There is not result of the problem.	Does not suggests or refine solutions but is willing to try out solutions suggested by others. Few mistakes in code, but done along with comments, and easy to comprehend for the instructor. Few mistake in result.	Refines solutions suggested by others. Complete and error-free code is done. No comments in the code, but easy to comprehend for the instructor. Results are correctly produced.	Actively looks for and suggests solution to problems. Complete and error free code is done, easy to comprehend for the instructor. Results are correctly produced. Student incorporated comments in the code.	
Affective	**CLO:3 R3**	Lab Report **(A)**	Code of the problem is not given. Outputs are not provided. Explanation of the solution is not stated.	Code of the problem is not given. Output is not complete. Explanation of the solution is not satisfactory.	Code of the problem is not given. Output is completely given. Explanation of the solution is not satisfactory.	Code of the problem is not given. Output is completely given. Explanation of the solution is satisfactory.	
	CLO:1 R5	Discipline and Behavior **(A)**	Got and wandered around. More than two incidents of talking non-lab related stuff in laband/or any talk with other groups, voice level exceeding the appropriate level, use of cell phones and involvement in any non-lab activity.	Got out of seat and wander around for some time. No more than two incidents of talking non-lab related stuff in lab Voice level exceeding theappropriate level, use of cell phones and involvement in any non-lab related activity.	Stayed in seat and got up for a specific lab related reason, but took more time than required to do the job. No more than one incidents of talking non-lab related stuff in lab. Voice level exceedingthe appropriate level, use of cell phones and involvementin any non-lab related activity.	Stayed in seat and got up for a specific lab related reason. Tookcare of lab related business and sat down right away. Voice level kept appropriate. Not used cell phones or involved in any non- lab related activity.	