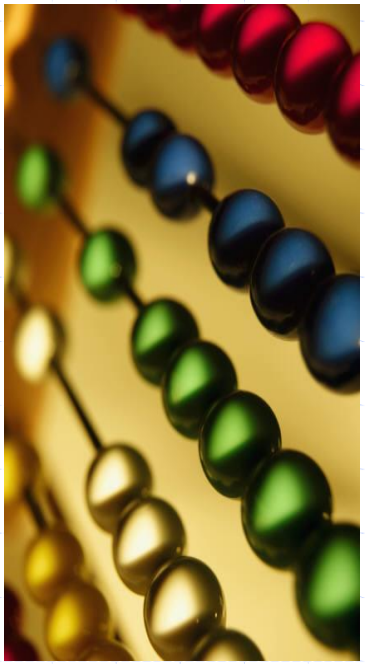
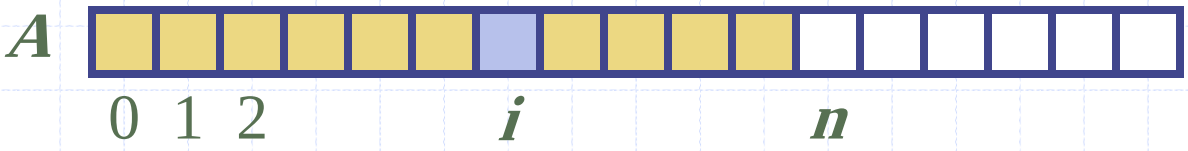
Arrays
Dr.laheeb Alzubaidy



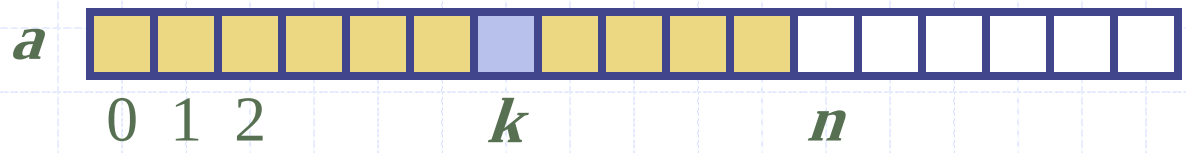
Array Definition

- An ***array*** is a sequenced collection of variables all of the same type. Each variable, or ***cell***, in an array has an ***index***, which uniquely refers to the value stored in that cell. The cells of an array, A, are numbered 0, 1, 2, and so on.
- Each value stored in an array is often called an ***element*** of that array.



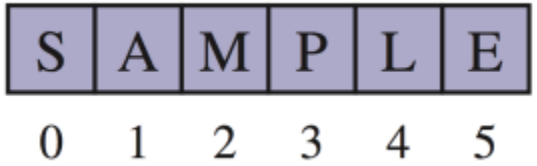
Array Length and Capacity

- Since the length of an array determines the maximum number of things that can be stored in the array ---***capacity***.
  - numbered 0, 1, 2, and so on, up through *a*.length−1,
  - the cell with index *k* can be accessed with syntax *a*[*k*].

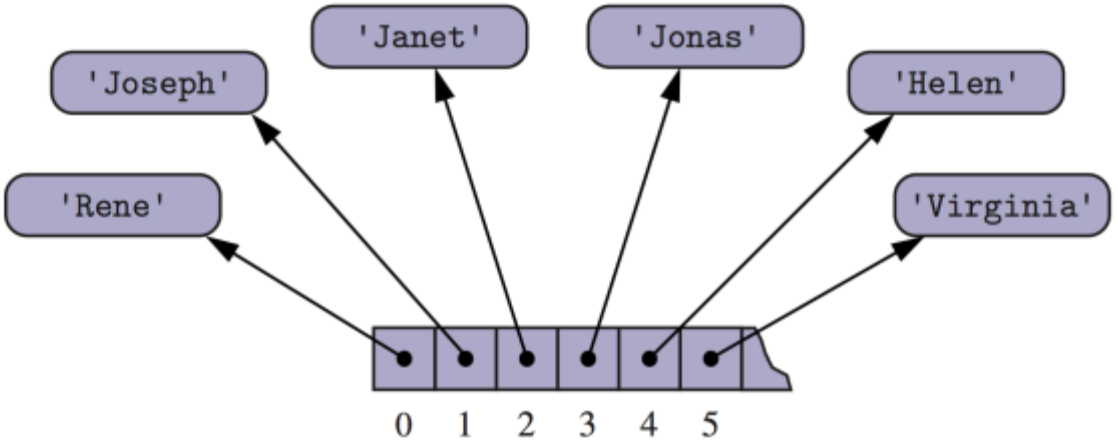


Arrays of Characters or Object References

- An array can store primitive elements, such as characters.

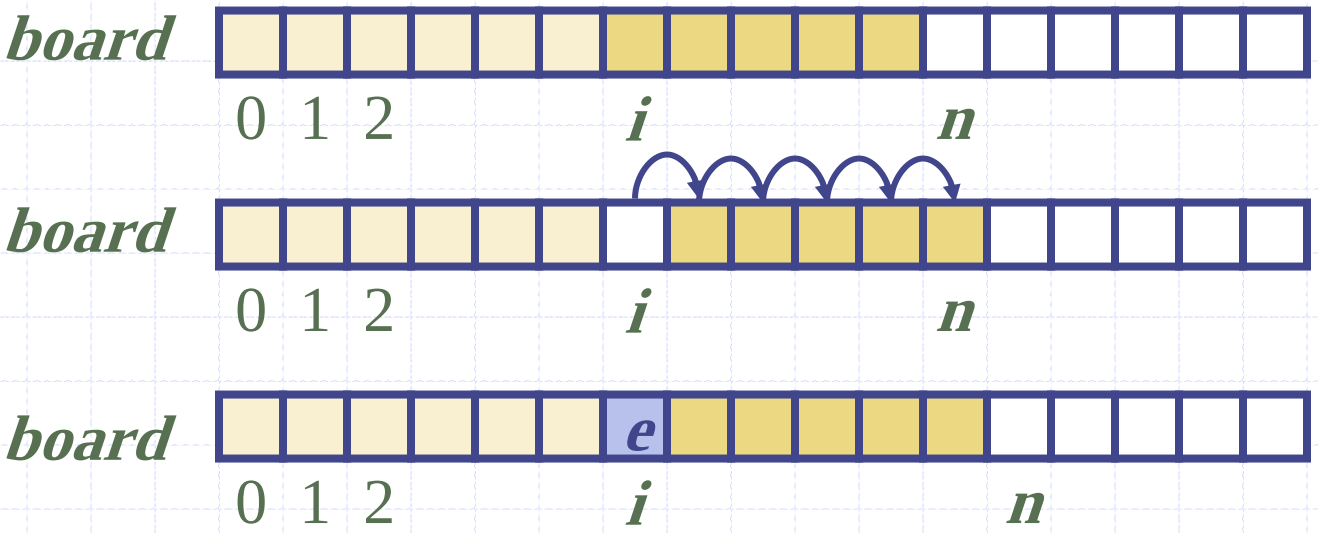


- An array can also store references to objects.



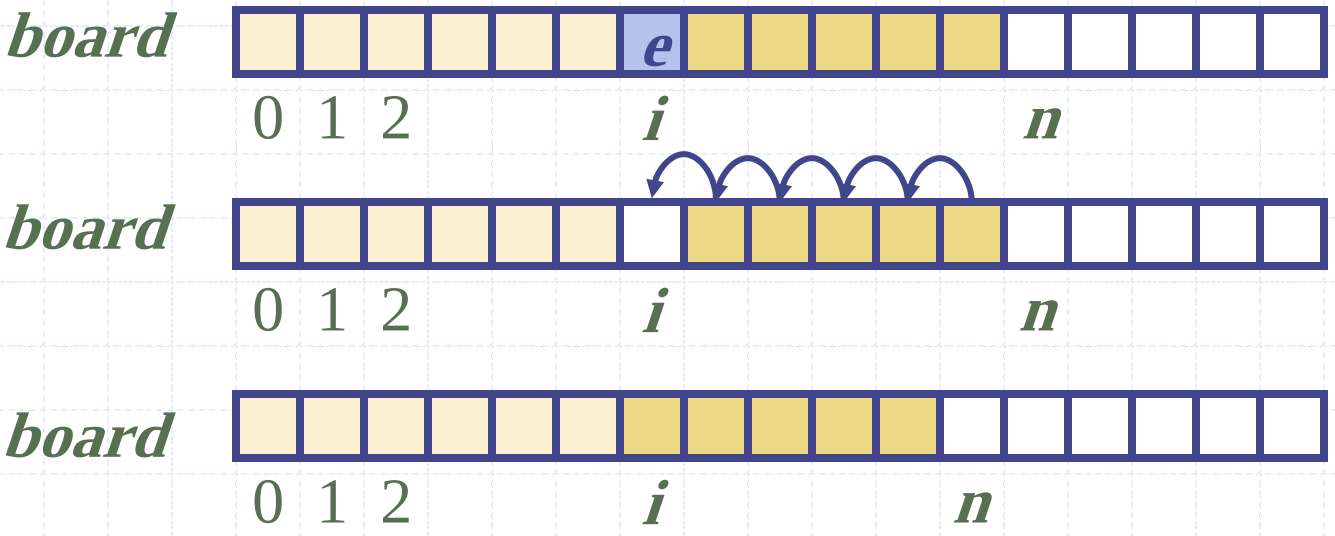
Adding an Entry

- To add an entry e into array board at index i, we need to make room for it by shifting forward the $n - i$ entries $board[i], \ldots, board[n-1]$



Removing an Entry

- To remove the entry e at index i, we need to fill the hole left by e by shifting backward the $n - i - 1$ elements $board[i+1], \ldots, board[n-1]$



Disadvantages of arrays?

- Need to know the maximum size before hand
- If elements need to be kept in order
  - Inserting/deleting an element requires moving (lots of) elements

Represented One Dim. Array in memory

- X : Array [1.. N] of integer {any other type}
- Location {X[I]} = Base Address + (I – 1)

- Example :calculate the address of element No. 4 in one dim. Array z: Array [1..6] of integer, where base address = 500
- Location {z(4)} = 500 + (4-1)
- =503

Represented Two Dim. Array in memory Row Wise method

- A : Array [1.. N, 1..M] of integer {any other type}
- Location {A[I,J]} = Base Address + N * (I – 1) + (J-1)
- Where N:total number of column
- I: row number
- J:column Number
- Example :calculate the address of element T[4,6] in one dim. Array T: Array [1..5, 1..7] of integer, where base address = 900
- Location {T(4,6)} = 900 +7 * (4-1) + (6-1)
- =926

Represented Two Dim. Array in memory Column Wise method

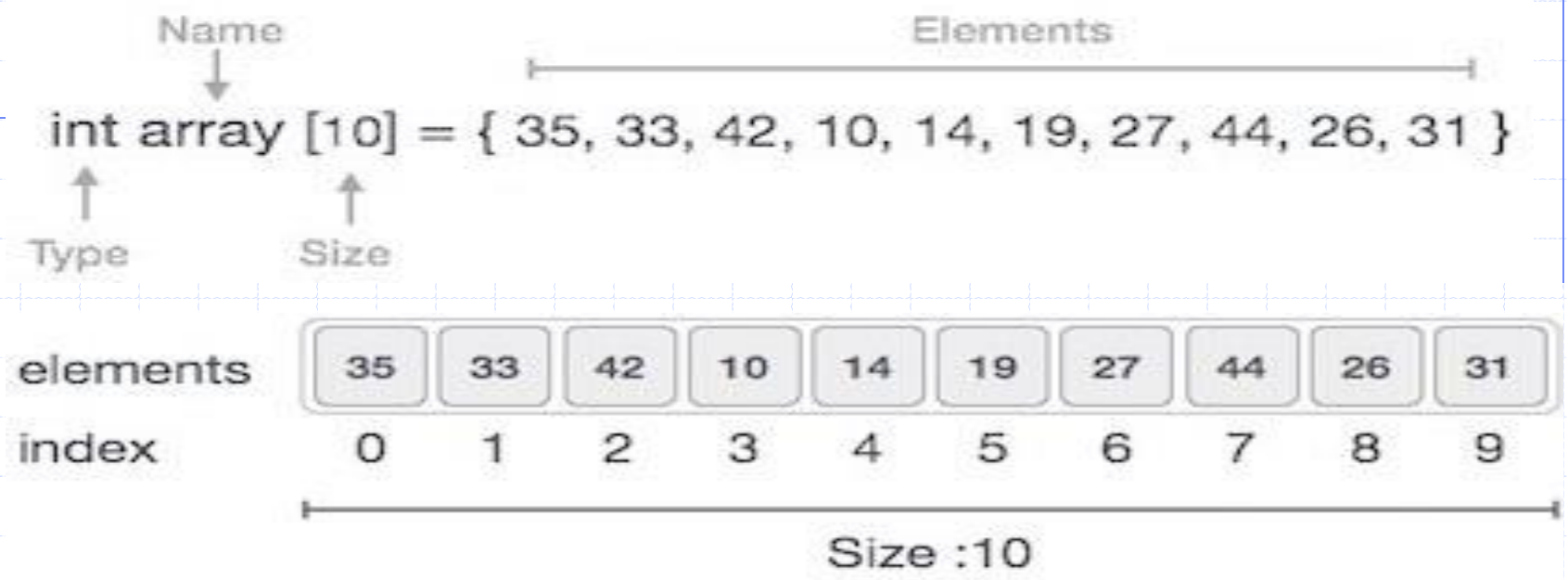
- A : Array [1.. N, 1..M] of integer {any other type}
- Location {A[I,J]} = Base Address + M * (J – 1) + (I-1)
- Where M:total number of row
- I: row number
- J:column Number
- Example :calculate the address of element T[5,7] in two dim. Array T: Array [1..6, 1..8] of integer, where base address = 300
- Location {T(5,7)} = 300 + 6 * (7-1) + (5-1)
- =340

Array Representation in Python

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.

Arrays can be declared in various ways in different languages. Below is an illustration.



As per the above illustration, following are the important points to be considered.

- Index starts with 0.
- Array length is 10 which means it can store 10 elements.
- Each element can be accessed via its index. For example, we can fetch an element at index 6 as 19.

Basic Operations

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.

from array import *arrayName = array(typecode, [Initializers])

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

Typecode	Value
b	Represents signed integer of size 1 byte/td>
B	Represents unsigned integer of size 1 byte
c	Represents character of size 1 byte
i	Represents signed integer of size 2 bytes
I	Represents unsigned integer of size 2 bytes
f	Represents floating point of size 4 bytes
d	Represents floating point of size 8 bytes

creates an array

- Before looking at various array operations lets create and print an array using python.
- The below code creates an array named array1.

```
from array import *
array1 = array('i', [10,20,30,40,50])
for x in array1:
print(x)
```

When we compile and execute the above program, it produces the following result –

- **Output**

```
10
20
30
40
50
```

Accessing Array Element

- We can access each element of an array using the index of the element. The below code shows how

```
from array import *
array1 = array('i', [10,20,30,40,50])
print (array1[0])
print (array1[2])
```

- When we compile and execute the above program, it produces the following result – which shows the element is inserted at index position 1.
- **Output**

```
10
30
```

Insertion Operation

Insert operation is to insert one or more data elements into an array. Based on the requirement, a new element can be added at the beginning, end, or any given index of array.

Here, we add a data element at the middle of the array using the python in-built insert() method.

```
from array import *
array1 = array('i', [10,20,30,40,50])
array1.insert(1,60)
for x in array1:
print(x)
```

- When we compile and execute the above program, it produces the following result which shows the element is inserted at index position 1.
- **Output**

```
10
60
20
30
40
50
```

Deletion Operation

- Deletion refers to removing an existing element from the array and re-organizing all elements of an array.
- Here, we remove a data element at the middle of the array using the python in-built remove() method.

```
from array import *
array1 = array('i', [10,20,30,40,50])
array1.remove(40)
for x in array1:
print(x)
```

- When we compile and execute the above program, it produces the following result which shows the element is removed form the array.
- **Output**

```
10
20
30
50
```

Search Operation

- You can perform a search for an array element based on its value or its index.
- Here, we search a data element using the python in-built index() method.

```
from array import *
array1 = array('i', [10,20,30,40,50])
print (array1.index(40))
```

When we compile and execute the above program, it produces the following result which shows the index of the element. If the value is not present in the array then the program returns an error.

- **Output**

3

Update Operation

Update operation refers to updating an existing element from the array at a given index. Here, we simply reassign a new value to the desired index we want to update.

```
 from array import *
array1 = array('i', [10,20,30,40,50])
array1[2] = 80 for x in array1: print(x)
```

When we compile and execute the above program, it produces the following result which shows the new value at the index position 2.

Output

```
10
20
80
40
50
```