Array in Data Structures: What is, Concept, Insert/Delete Operations Example

What are Arrays?

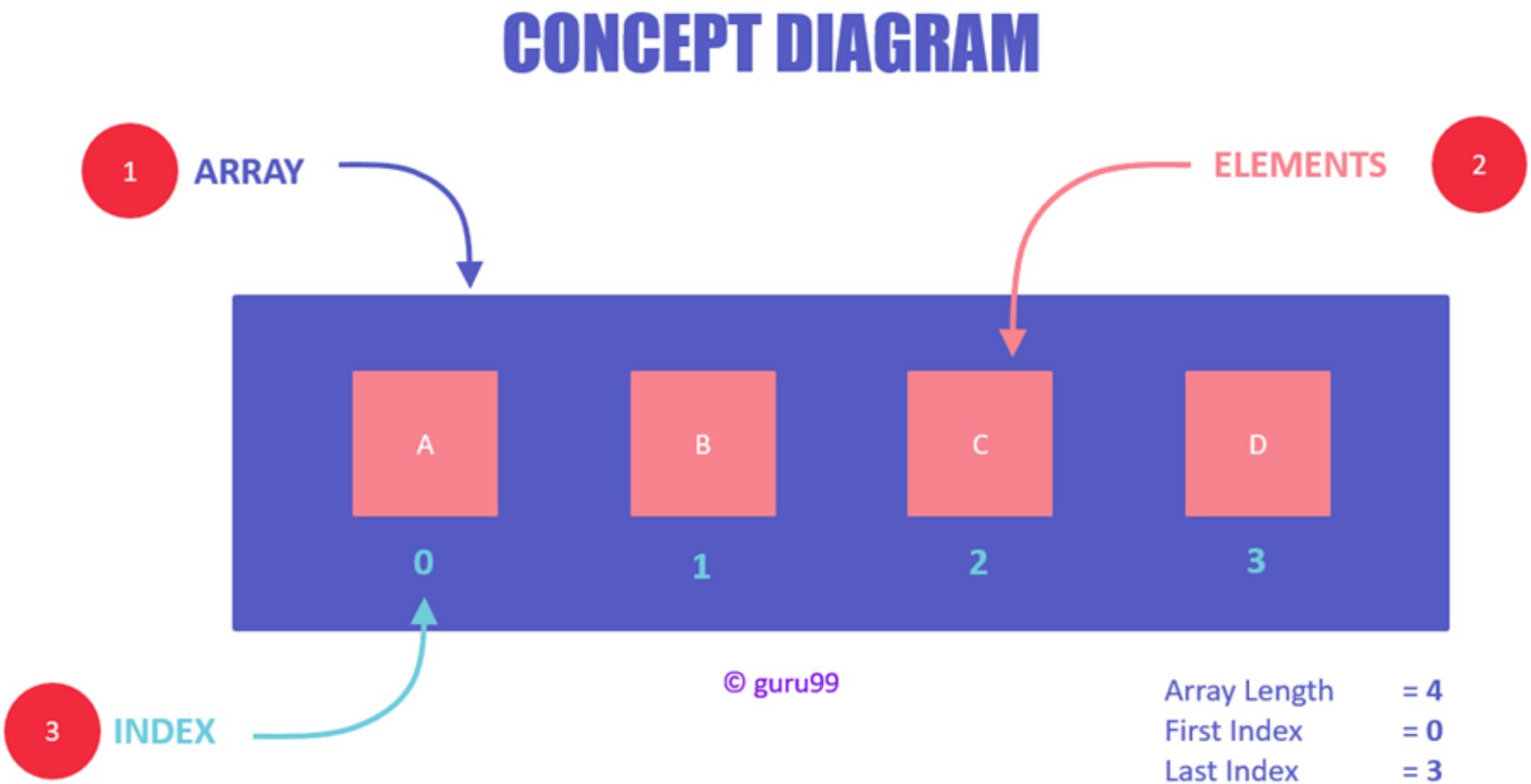
An array is a data structure for storing more than one data item that has a similar data type. The items of an array are allocated at adjacent memory locations. These memory locations are called **elements** of that array.

The total number of elements in an array is called **length**. The details of an array are accessed about its position. This reference is called **index** or **subscript**.

In this Data structure tutorial you will learn,

- What are Arrays?
- Concept of Array
- Why do we need arrays?
- Creating an Array in Python
- Ways to Declare an Array in Python
- Array Operations
- Creating an Array in C++
- Array Operations in C++
- Array Operations in Java

Concept of Array



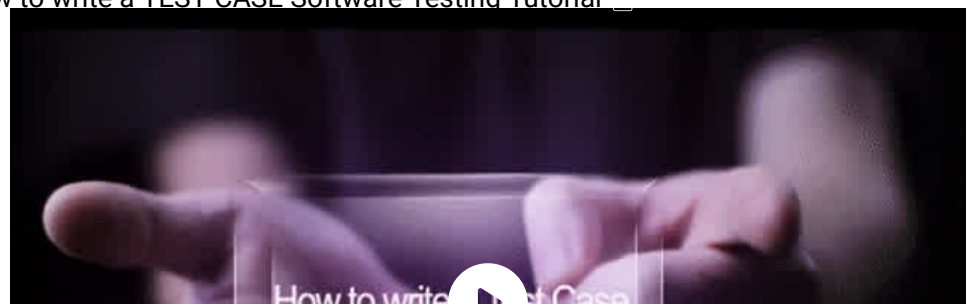
(/images/1/102319_0559_ArrayinData1.png)

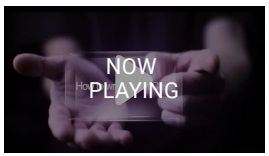
Concept Diagram of Arrays

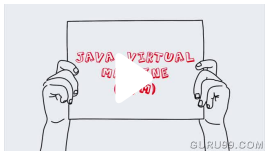
The above diagram illustrates that:

FEATURED VIDEOS

How to write a TEST CASE Software Testing Tutorial _





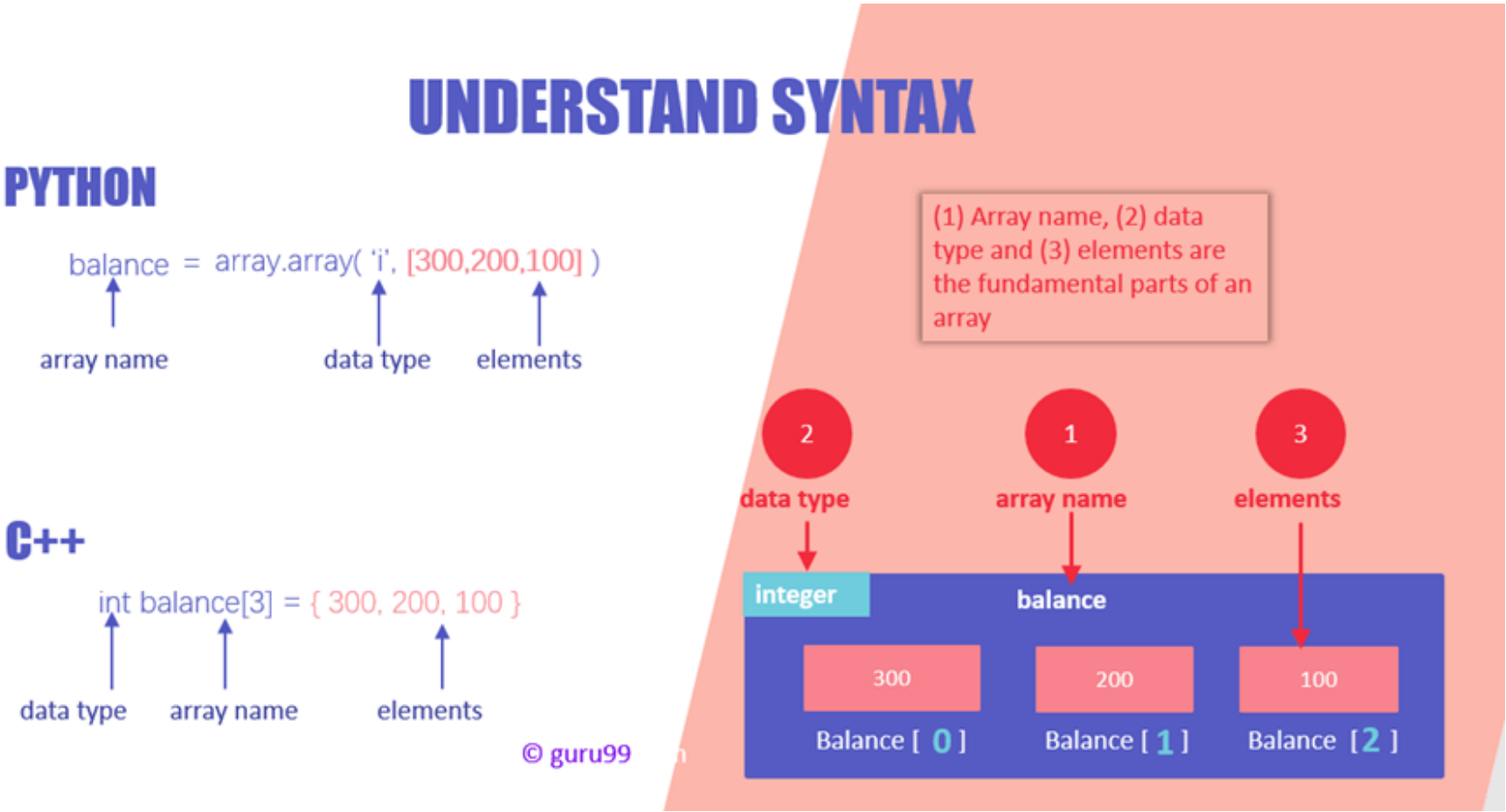


1. An array is a container of elements.
2. Elements have a specific value and data type, like "ABC", TRUE or FALSE, etc.
3. Each element also has its own index, which is used to access the element.

Note:

- Elements are stored at contiguous memory locations.
- An index is always less than the total number of array items.
- In terms of syntax, any variable that is declared as an array can store multiple values.
- Almost all languages have the same comprehension of arrays but have different ways of declaring and initializing them.
- However, three parts will always remain common in all the initializations, i.e., array name, elements, and the data type of elements.

The following diagram illustrates the syntax of declaring an array in Python and C++ to present that the comprehension remains the same though the syntax may slightly vary in different languages.



(/images/1/102319_0559_ArrayinData2.png)

Understand Syntax of Arrays

- **Array name:** necessary for easy reference to the collection of elements
- **Data Type:** necessary for type checking and data integrity
- **Elements:** these are the data values present in an array

Why do we need arrays?

Here, are some reasons for using arrays:

- Arrays are best for storing multiple values in a single variable

- Arrays are better at processing many values easily and quickly
- Sorting and searching the values is easier in arrays

Creating an Array in Python

In Python, arrays are different from lists; lists can have array items of data types, whereas arrays can only have items of the same data type.

Python has a separate module for handling arrays called array, which you need to import before you start working on them.

Note: The array must contain real numbers like integers and floats, no strings allowed.

The following code illustrates how you can create an integer array in python to store account balance:

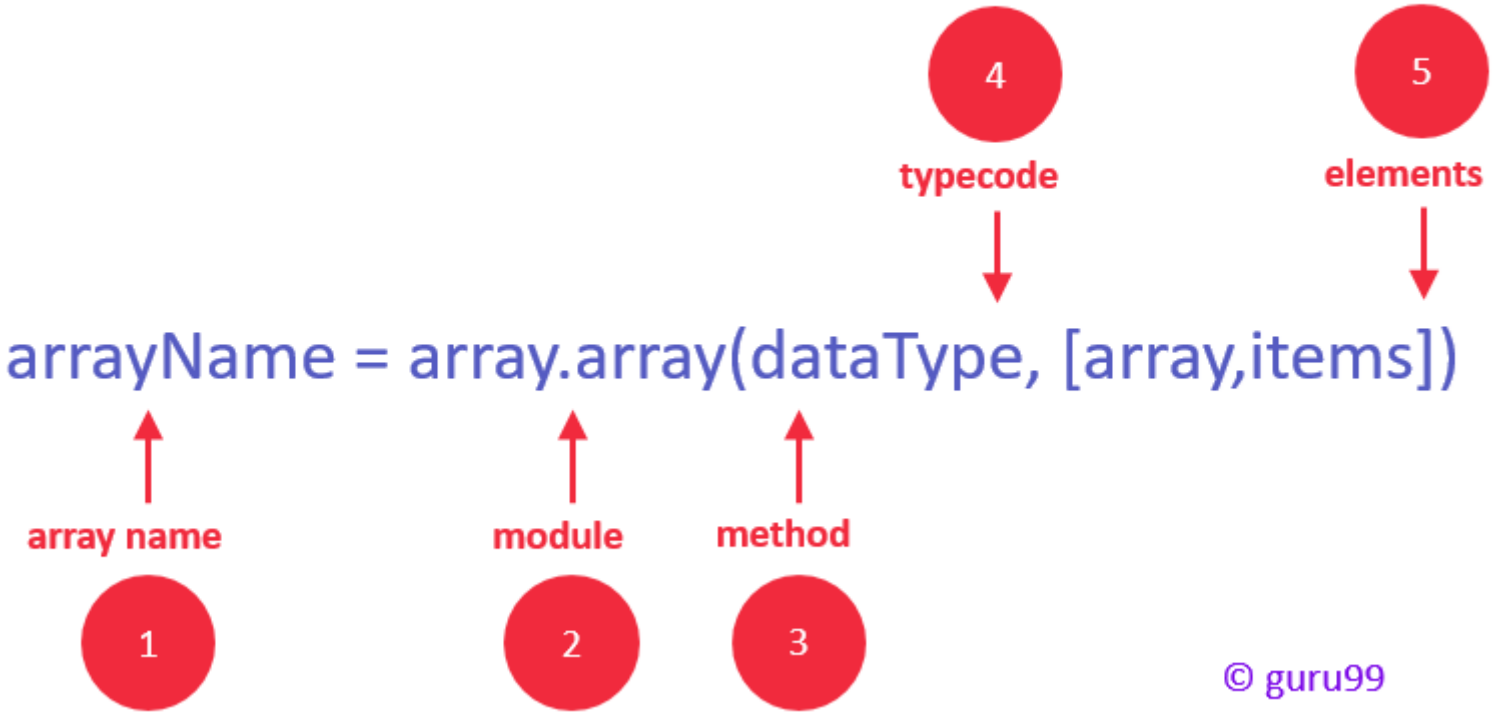
```
import array
balance = array.array('i', [300,200,100])
print(balance)
```

Ways to Declare an Array in Python

You can declare an array in Python while initializing it using the following syntax.

```
arrayName = array.array(type code for data type, [array,items])
```

The following image explains the syntax.



(/images/1/102319_0559_ArrayinData3.png)

Syntax of Array in Python

1. **Identifier**: specify a name like usually, you do for variables
2. **Module**: Python has a special module for creating arrays, called "array" – you must import it before using it
3. **Method**: the array module has a method for initializing the array. It takes two arguments, typecode, and elements.
4. **Type Code**: specify the data type using the typecodes available (see list below)
5. **Elements**: specify the array elements within the square brackets, for example [130,450,103]

The following table illustrates the typecodes available for supported data types:

Type code	C Type	Python Type	Minimum size in bytes

'c'	char	character	1
'B'	unsigned char	int	1
'b'	signed char	int	1
'u'	Py_UNICODE	Unicode character	2
'h'	signed short	int	2
'H'	unsigned short	int	2
'i'	signed int	int	2
'I'	unsigned int	long	2
'l'	signed long	int	4
'L'	unsigned long	long	4
'f'	float	float	4
'd'	double	float	8

How to access a specific array value?

You can access any array item by using its index.

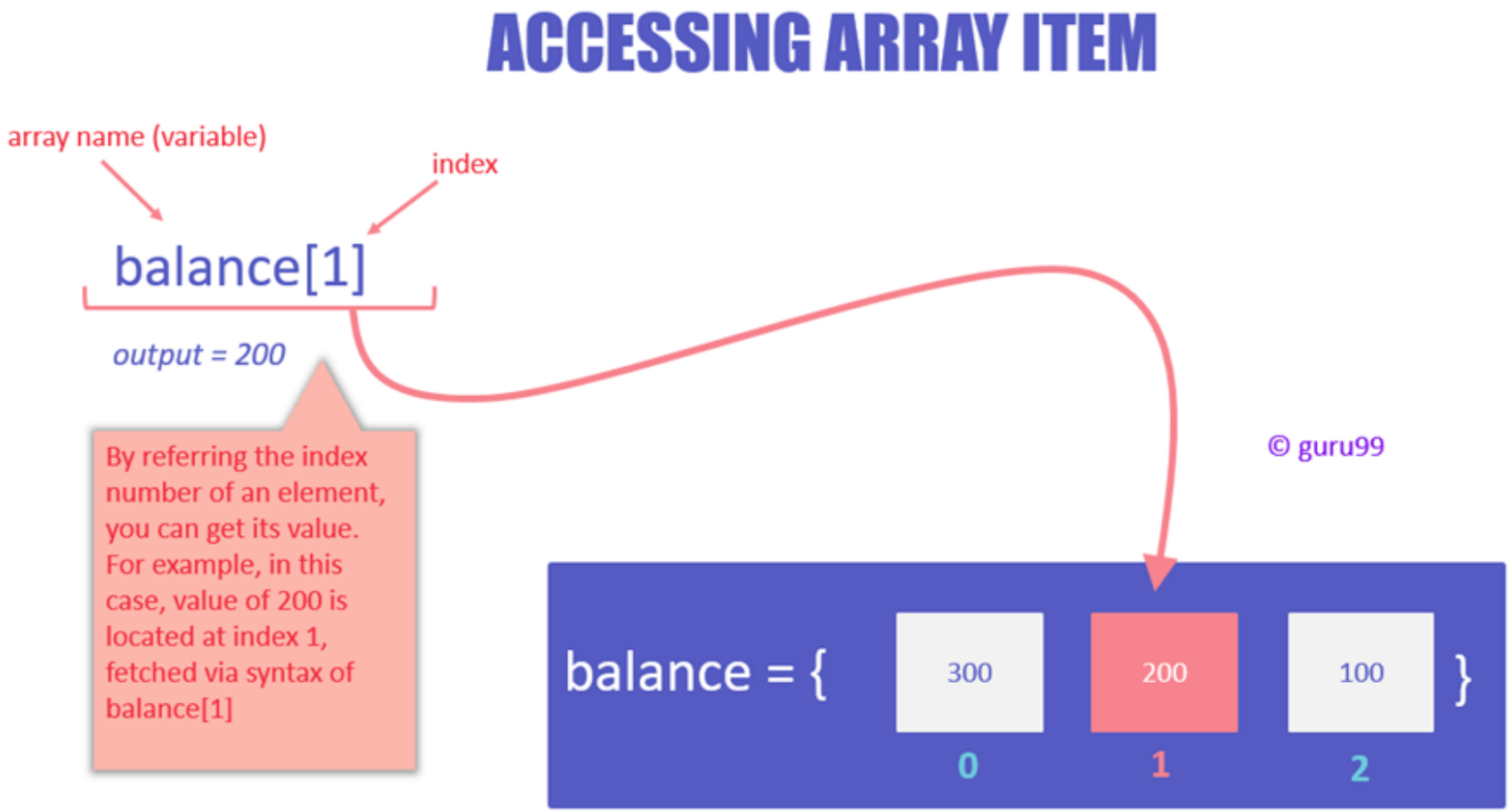
SYNTAX

```
arrayName[indexNum]
```

EXAMPLE

```
balance[1]
```

The following image illustrates the basic concept of accessing arrays items by their index.



(/images/1/102319_0559_ArrayinData4.png)

Access an Array Element

Here, we have accessed the second value of the array using its index, which is 1. The output of this will be 200, which is basically the second value of the balanced array.

```
import array
balance = array.array('i', [300,200,100])
print(balance[1])
```

OUTPUT

```
200
```

Array Operations

The array module of Python has separate functions for performing array operations. This is a destructive method of operating with arrays, which means the modification will be saved in the array variable.

Insert

With this operation, you can insert one or more items into an array at the beginning, end, or any given index of the array. This method expects two arguments index and value.

SYNTAX

```
arrayName.insert(index, value)
```

Example:

Let's add a new value right after the second item of the array. Currently, our balance array has three items 300, 200, and 100. So what's the index of the second array item with a value of 200 if you said 1.

In order to insert the new value right "after" index 1, you need to reference index 2 in your insert method, like this:

```
import array
balance = array.array('i', [300,200,100])
balance.insert(2, 150)
```

Now, to verify if the new value has been inserted, enter the array name and press Enter on the keyboard:

```
import array
balance = array.array('i', [300,200,100])
balance.insert(2, 150)
print(balance)
```

OUTPUT

```
array('i', [300,200,150,100])
```

Delete

With this operation, you can delete one item from an array by value. This method accepts only one argument, value. After running this method, the array items are re-arranged, and indices are re-assigned.

SYNTAX

```
arrayName.remove(value)
```

Example

Let's remove the value of 150 from the array. Currently, our balance array has four items 300, 200, 150, and 100. So, in order to remove 150 from the array, we only have to type 150 inside the method argument. Simple, right?

```
import array
balance = array.array('i', [300,200,100])
balance.insert(2, 150)
print(balance)
balance.remove(150)
```

Now, to verify if the value has been deleted, enter the array name and press Enter on the keyboard:

```
import array
balance = array.array('i', [300,200,100])
balance.insert(2, 150)
print(balance)
balance.remove(150)
print(balance)
```

OUTPUT

```
 array('i', [300,200,100])
```

SEARCH

With this operation, you can search for an item in an array based on its value. This method accepts only one argument, value. This is a non-destructive method, which means it does not affect the array values.

SYNTAX

```
arrayName.index(value)
```

Example:

Let's search for the value of 150 in the array. Currently, our balance array has four items 300, 200, 150, and 100. So, in order to search 150 in the array, we only have to type 150 inside the method argument. That's quite easy. This method returns the index of the searched value.

```
import array
balance = array.array('i', [300,200,150,100])
print(balance.index(150))
```

OUTPUT

```
2
```

UPDATE

This operation is quite similar to the insert method, except that it will replace the existing value at the given index. This means will simply assign a new value at the given index. This method expects two arguments index and value.

Syntax

```
arrayName.udpate(index, value)
```

Example

Assume that our array has four items 300, 200, 150, and 100, and we want to replace 150 with 145. So what's the index 150?

Kudos, if you said 2.

In order to replace 150 that has index 2, you need to reference index 2 by using simple assignment operator, like this one:

```
import array
balance = array.array('i', [300,200,150,100])
balance[2] = 145
```

Now, to verify if the value has been updated, enter the array name and press Enter on the keyboard:

```
import array
balance = array.array('i', [300,200,150,100])
balance[2] = 145
print(balance)
```

OUTPUT

```
array('i', [300,200,145,100])
```

Traverse

You can traverse a python array by using loops, like this one:

```
import array
balance = array.array('i', [300,200,100])
for x in balance:
        print(x)
```

OUTPUT

```
300
200
100
```

Creating an Array in C++

C++ language is more flexible than Python when it comes to creating arrays. You can create arrays in three ways mentioned ahead.

The following code illustrates how you can create an integer array in C++ to store account balance:

```
#include <iostream>
using namespace std;

int main()
{
  int balance[3] = { 300, 200, 100 };
  for (int i = 0; i < 3; i++)
    {
      cout << "value of i: " << balance[i] << endl;
    }
  return 0;
}
```

Ways to Declare an Array in C++

You can declare an array in three syntax variants. Which one suits your program; this choice is based on your program requirements.

Declaration by Size

Syntax

```
dataType arrayName[arraySize];
```

Example

```
int balance[3];
```

Declaration Initialization Array Items Only

Syntax

```
dataType arrayName[] = {array, items};
```

Example

```
int balance[] = { 300, 200, 100 };
```

Declaration by Size and Initialization Array Items

Syntax

```
dataType arrayName[arraySize] = {array, items};
```

Example

```
int balance[3] = { 300, 200, 100 };
```

How to access a specific array value?

You can access any array item by using its index.

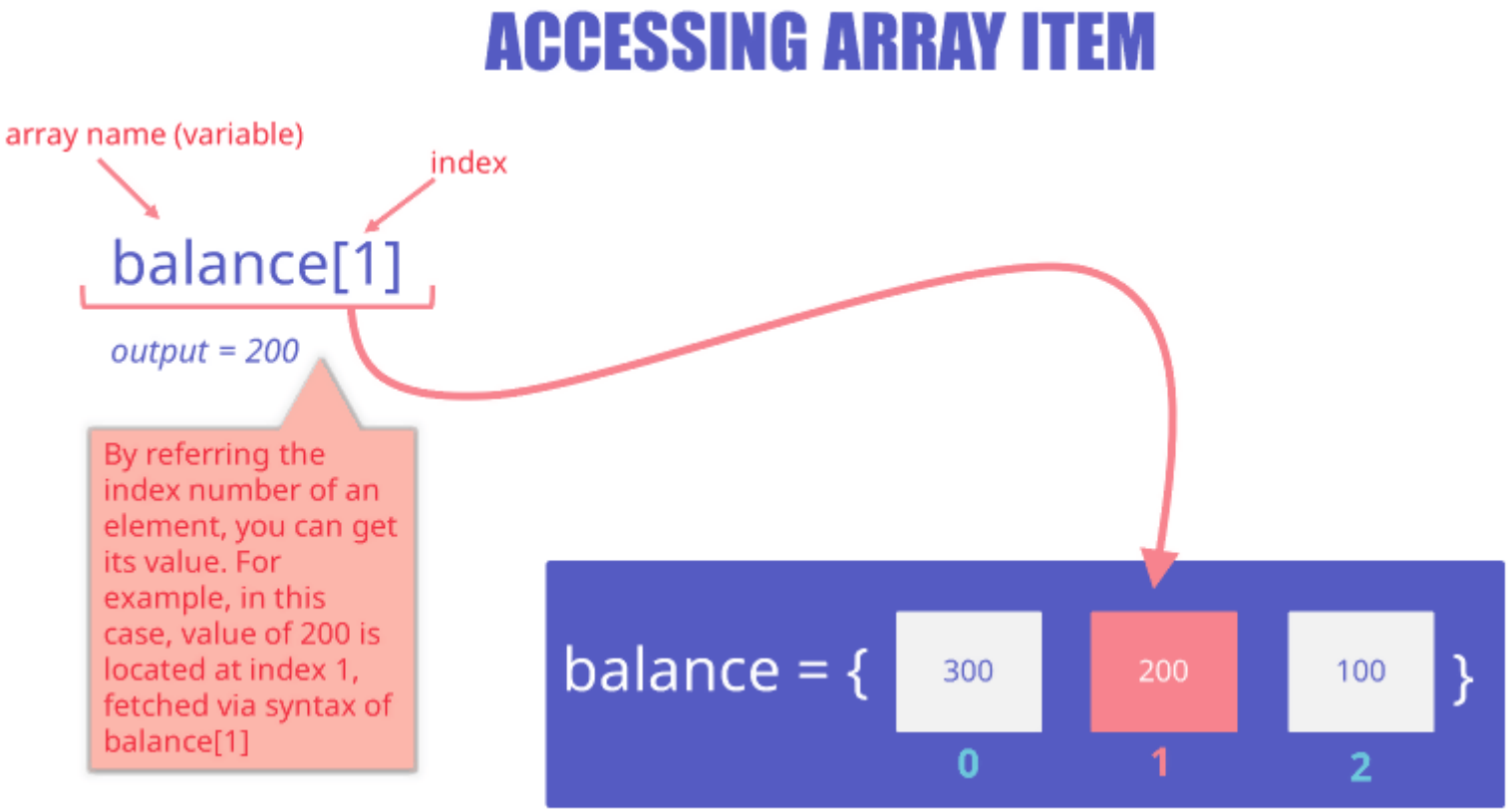
Syntax

```
arrayName[indexNum]
```

Example

```
balance[1]
```

The following image illustrates the basic concept of accessing arrays items by their index.



(/images/1/102319_0559_ArrayinData5.png)

Accessing an Array Element

Here, we have accessed the second value of the array using its index, which is 1. The output of this will be 200, which is basically the second value of the balance array.

```
#include <iostream>
using namespace std;

int main()
{
  int balance[3] = { 300, 200, 100 };
  cout << balance[1];

  return 0;
}
```

Output

```
200
```

Array Operations in C++

Unlike Python, in C++ you have to program the logic yourself for performing the insert, delete, search update and traverse operations on C++ arrays.

Insert

The logic for insertion operation goes as follows:

- loop through the array items
- shift them to a greater index
- add a new array item at a given index

In the following example, we have 5 items in the balance array, and we want to add a new item just after the value of 200. This means we have to shift all the items after 200 to a greater index, and then insert our new value of 150.

```
#include <iostream>
#include <stdio.h>

main() {
   int pos = 2;
   int size = 4;
   int balance[] = {300,200,100,50,0};

   printf("BEFORE INCREMENT: \n");
   for(int i = 0; i<5; i++) {
      printf("%d\n",balance[i]);
   }

   /* FOR SHIFTING ITEMS TO A GREATER INDEX */
   for(int i = size; i >= pos; i--) {
       balance[i+1]=balance[i];
   }

   /* FOR INSERTING VALUE AT OUR DESIRED INDEX */
   balance[pos] = 150;

   printf("AFTER INCREMENT: \n");


   /* FOR PRINTING THE NEW ARRAY */
   for(int i = 0; i<6; i++) {
      printf("%d\n",balance[i]);
   }
}
```

Output

```
BEFORE INCREMENT
300
200
100
50
0
AFTERINCREMENT
300
200
150
100
50
0
```

Array Operations in Java

Lets create a programming in Java, in this program we will accept the size and the value of the array elements from the user.

```
import java.util.Scanner;

public class AddElements {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the size of the array");
        int n=sc.nextInt();

        int arr[]=new int[n];

        System.out.println("Enter Elements in the array");
        for(int i=0;i<n;i++)
        {
            arr[i]=sc.nextInt();
        }
        System.out.println("Elements in the array");
        for(int j=0;j<n;j++)
        {
            System.out.print(arr[j]+" ");
        }
    }
}
```

Output:-

```
Enter the size of the array

5
Enter Elements in the array

1
2
3
4
5
Elements in the array
1 2 3 4 5
```

Modify Element in Array:-

Update an element by the given index.

Program in Java for how to Modify elements in an array

```
import java.util.Scanner;

public class ModifyElement {
    public static void main(String[] args) {
        int arr[]={1,2,3,4,5};
        int length= arr.length;
        Scanner sc=new Scanner(System.in);

        System.out.println("Array Elements Before modify");
        for(int i=0;i<length;i++)
        {
            System.out.print(arr[i]+" ");
        }


        System.out.println("\nEnter the position where you want to change in an array");
        int pos=sc.nextInt();

        System.out.println("Enter the value");
        int val=sc.nextInt();

        arr[pos]=val;

        System.out.println("Array Elements After modify");
        for(int j=0;j<length;j++)
        {
            System.out.print(arr[j]+" ");
        }
    }
}
```

Output:-

```
Array Elements Before modify
1 2 3 4 5
Enter the position where you want to change in an array

2
Enter the value
8
Array Elements After modify
1 2 8 4 5
```

Access Element in Array:-

Print all array elements.

Program in Java for how to Traverse in array

```
public class AccessElements {
    public static void main(String[] args) {
        int arr[]={1,2,3,4,5};
        int length= arr.length;

        System.out.println("Array Elements are:-");
        for(int i=0;i<length;i++)
        {
            System.out.print(arr[i]+" ");
        }

    }
}
```

Output:-

```
Array Elements are:-
1 2 3 4 5
```

Summary:

- An array is a data structure for storing multiple data items that have a similar data type
- Identifier, data type, array length, elements, and index are the major parts of an array
- Use the index for processing the values of array elements
- Arrays have excellent support for keeping data-type intact
- In most languages, an array is created by specifying an identifier, data type, and elements to include
- Arrays are best for processing a large number of values, and for quick sorting and searching
- Python has modules and built-in methods to perform basic array operations like insert, delete, search, update and traverse
- C++ needs defining programs for basic array operations like insert, delete, search update and traverse

Prev

Report a Bug

Next

YOU MIGHT LIKE:

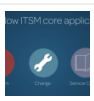
DEVOPS


(/devops-tools.html) (/devops-tools.html)

30 BEST DevOps Automation Tools & Technologies (2020 List)

(/devops-tools.html)

DEVOPS


(/servicenow-tutorial.html) (/servicenow-tutorial.html)

ServiceNow Training Tutorial: What is, Use, Reporting

(/servicenow-tutorial.html)

R PROGRAMMING


(/histogram-vs-bar-chart.html) (/histogram-vs-bar-chart.html)

Histogram vs Bar Graph: Must Know Differences

(/histogram-vs-bar-chart.html)

LINUX

DEVOPS

SDLC

(/unix-virtual-terminal.html)
(/unix-virtual-terminal.html)

Linux/Unix Virtual Terminal

(/unix-virtual-terminal.html)


(/puppet-tutorial.html)
(/puppet-tutorial.html)

Puppet Tutorial for Beginners: Resources, Classes, Manifest, Modules

(/puppet-tutorial.html)

(/front-end-web-development-tools.html) (/front-end-web-development-tools.html)

20 Best Front End Web Development Tools in 2020

(/front-end-web-development-tools.html)

Design & Algorithms Tutorial

1) Greedy Algorithm (/greedy-algorithm.html)

2) Circular Linked List (/circular-linked-list.html)

3) Array in Data Structures (/array-data-structure.html)

4) B TREE in Data Structure (/b-tree-example.html)

5) B+ TREE (/introduction-b-plus-tree.html)

(https://www.facebook.com/guru99com/)
(https://twitter.com/guru99com)
(https://www.linkedin.com/company/guru99/)
(https://www.youtube.com/channel/UC19i1XD6k88KqHlET8atqFQ)
(https://forms.aweber.com/form/46/724807646.htm)

About

About Us (/about-us.html)
Advertise with Us (/advertise-us.html)
Write For Us (/become-an-instructor.html)
Contact Us (/contact-us.html)

Career Suggestion

SAP Career Suggestion Tool (/best-sap-module.html)
Software Testing as a Career (/software-testing-career-complete-guide.html)

Interesting

eBook (/ebook-pdf.html)
Blog (/blog/)
Quiz (/tests.html)
SAP eBook (/sap-ebook-pdf.html)

Execute online

Execute Java Online (/try-java-editor.html)
Execute Javascript (/execute-javascript-online.html)
Execute HTML (/execute-html-online.html)
Execute Python (/execute-python-online.html)

© Copyright - Guru99 2020