

### Array

An array stores multiple values in one single variable. An array is a sequence of data item of same type or not same type value. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index. In PHP, the **array ()** function is used to create an array.

In PHP, there are three types of arrays:

1. Indexed arrays - Arrays with a numeric index
2. Associative arrays - Arrays with named keys
3. Multidimensional arrays - Arrays containing one or more arrays

#### 1. Indexed or One dimensional array:

##### Declaration and Initialization of Index or one-dimensional array:

To declare an array in PHP, a programmer specifies the items of the elements and the number of elements required by an **array ()** function as follows;

**Syntax:** *\$ArrayName* = **array** ( "item1", "item2", ..);

**Example:**

```
$age = array (25, 40, 23, 45, 76);
```

Here, the name of array is **age**. The size of array is 5, i.e., there are 5 items (elements) of array **age**. All elements in an array are of the same type (integer, in this case).

##### Array elements

Size of array defines the number of elements in an array. Each element of array can be accessed and used by user according to the need of program. For example: `int age[5];`

Age[0]		1 element of array
Age[1]		2 element of array
Age[2]		3 element of array
Age[3]		4 element of array
Age[4]		5 element of array

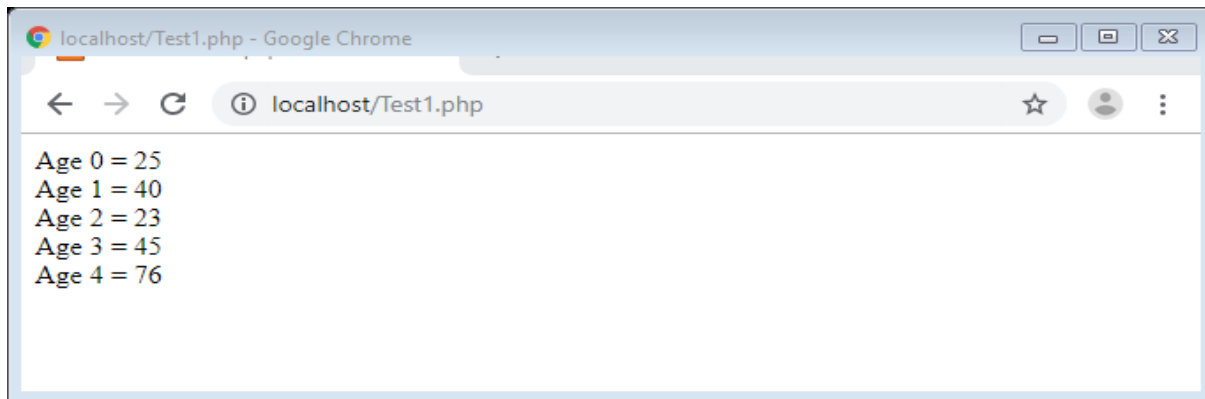
Note that, the first element is numbered 0, second element 1 and so on.

### Accessing array elements

Traversing: We can traverse an indexed array using loops in PHP. We can loop through the indexed array in two ways. First by using for loop and secondly by using **foreach**. You can refer to PHP | Loops for the syntax and basic use.

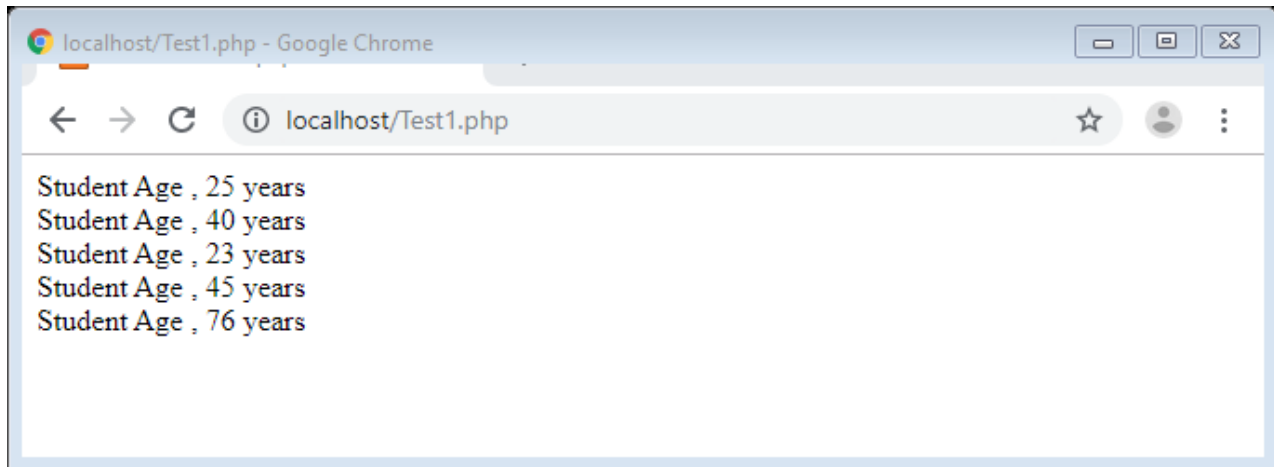
#### Example:

```
<?php
    $age = array (25, 40, 23, 45, 76);
    for ( $i=0; $i<=4; $i++)
    {
        echo " Age $i = $age[$i] <br>";
    }
?>
```



### Using foreach loop

```
<?php
    $age = array (25, 40, 23, 45, 76);
    foreach ($age as $val)
    {
        echo "$val.,br>";
    }
?>
```



If you have a list of items (a list of Fruits names, for example), storing the Fruits in single variables. The index can be assigned automatically (index always starts at 0), like this:

```
<?php
```

```
$Fruits = array ("Mango", "Banana", "Apple");
```

```
?>
```

or the index can be assigned manually like this:

```
<?php
```

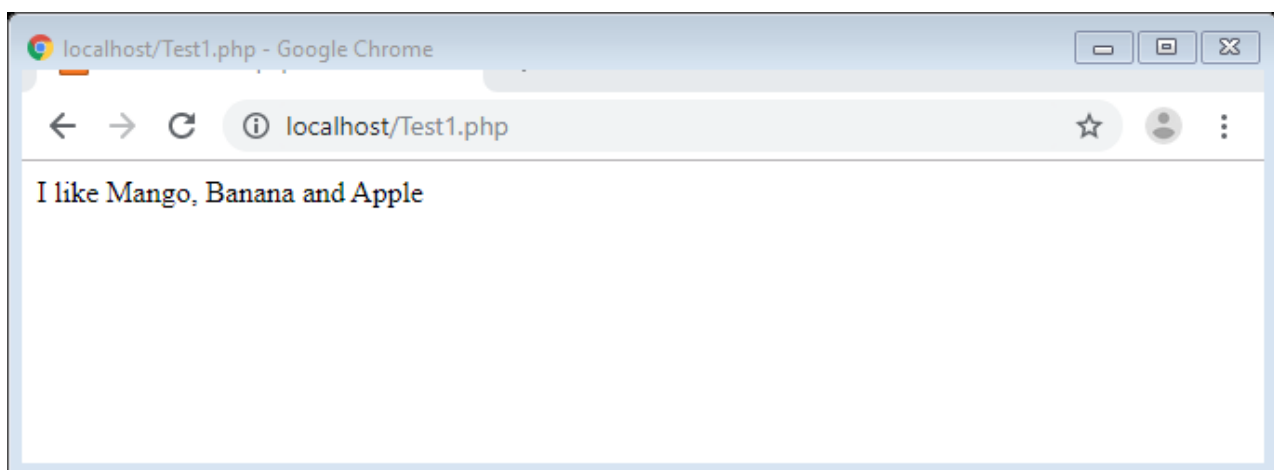
```
    $Fruits [0] = "Mango";
```

```
    $Fruits [1] = "Banana";
```

```
    $Fruits [2] = "Apple";
```

```
    echo "I like " . $Fruits[0] . ", " . $Fruits[1] . " and " . $Fruits[2] ;
```

```
?>
```

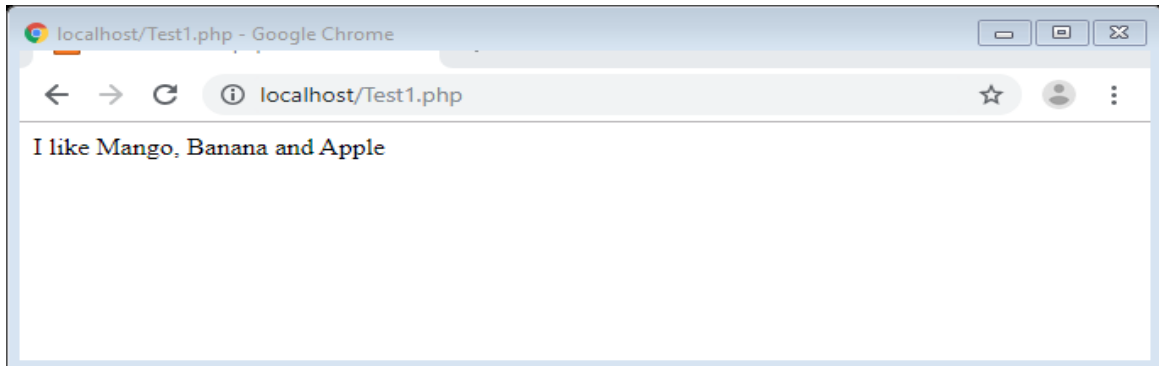


The following example creates an indexed array named \$Fruits, assigns three elements to it, and then prints a text containing the array values:

**<?php**

```
$Fruits = array ("Mango", "Banana", "Apple");  
echo "I like " . $Fruits[0] . ", " . $Fruits[1] . " and " . $Fruits[2] ;
```

**?>**



## 2. Associative arrays

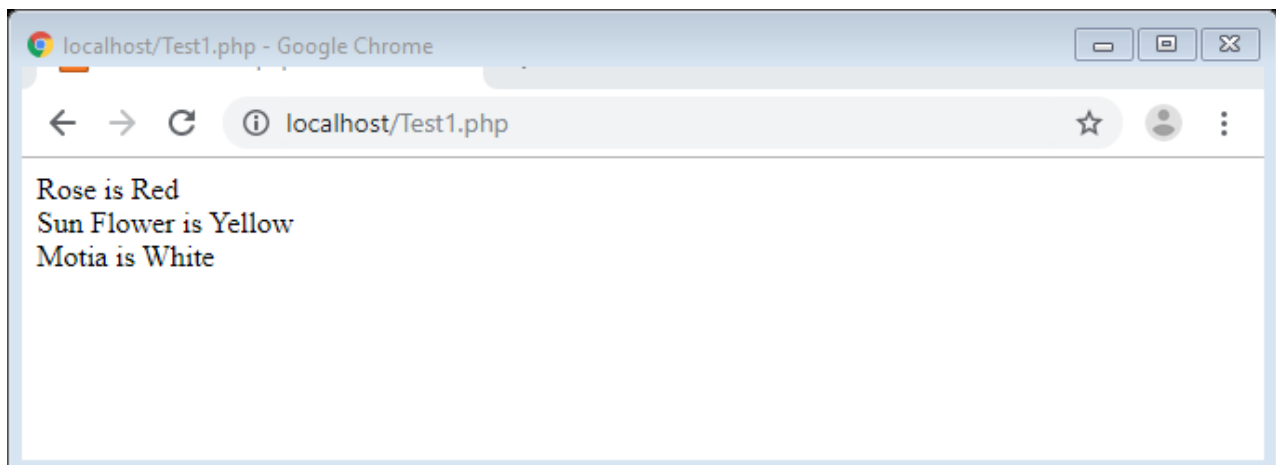
These types of arrays are similar to the indexed arrays but instead of linear storage, every value can be assigned with a user-defined the key can either be an integer or a string. The value can be of any type.

**Syntax:** *\$ArrayName = array ( key1 => value, key2 => value2, key3 => value3, ...)*

**<?php**

```
$Flowers = array ("Rose"=>"Red", "Sun Flower"=>"Yellow", "Motia"=>"White");  
foreach ($Flowers as $flower => $color)  
{  
    echo "$flower is ".$color . " <br> ";  
}
```

**?>**



The key can either be an integer or a string. The value can be of any type.

### Example

```
$Flowers = array ("Rose"=>"Red", 22=>"Yellow", true=>"White");
```

### 3. Multidimensional arrays

Multi-dimensional arrays are such type of arrays which stores an another array at each index instead of single element. It can be created using nested array. These type of arrays can be used to store any type of elements, but the index is always a number.

A multidimensional array is an array containing one or more arrays. PHP understands multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Initializing t

Multidimensional arrays may be initialized by specifying bracketed values for each row. Following is an array with 4 rows and each row has 3 columns.

**Example:**

**<?php**

```
$records = array ( array ("asif","Assitant",25000), // initializers for row indexed by 0
                  array ("Nadeem","Accountant",3000), // initializers for row indexed by 1
                  array ("Farhan","Programmer",455000), // initializers for row indexed by 2
                  array ("Rashid","Operator",1000)      // initializers for row indexed by 3
                );
```

**?>**

**Accessing Two-Dimensional Array Elements:**

An element in a two-dimensional array is accessed by using the subscripts, i.e., row index and column index of the array.

**Example:**

**<?php**

```
echo $record[1][2];
```

**?>**

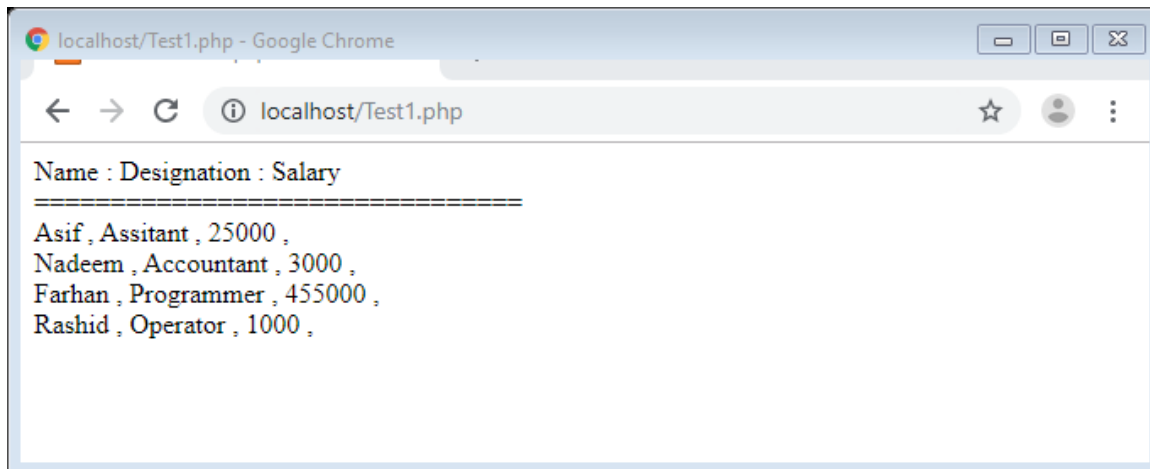
The above statement will take the 3<sup>rd</sup> column element from the 2<sup>nd</sup> row of the array.

**Example of PHP program (Multi-Dimensional Array):**

**<?php**

```
$records = array ( array ("Asif","Assitant",25000),
                  array ("Nadeem","Accountant",3000),
```

```
        array ("Farhan","Programmer",455000),
        array ("Rashid","Operator",1000) );
echo "Name   :  Designation   : Salary <br>";
echo "=====<br>";
for($row=0; $row<4; $row++){
    for($col=0; $col<3; $col++){
        echo $records[$row][$col]. " , " ;
    }
    echo "<br>";
}
?>
```



## Multidimensional Arrays with associative array

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

### Example

In this example an associative array, we create a two dimensional array to store marks of three students in three subjects. You can create numeric array in the same fashion.

**<?php**

```
$marks = array(
    "Muhammad" => array (
        "physics" => 35,
        "maths" => 30,
        "chemistry" => 39
    ),
);
```

```
"Nadeem" => array (
    "physics" => 30,
    "maths" => 32,
    "chemistry" => 29
),

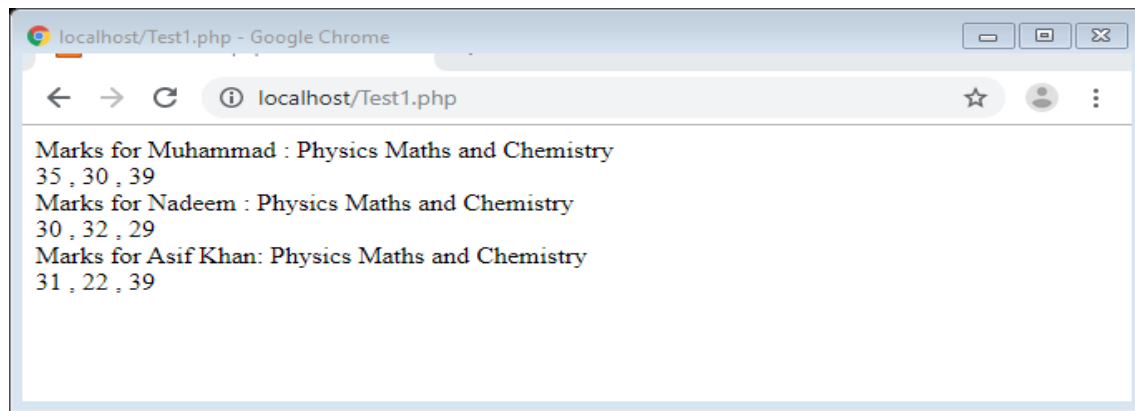
"Asif Khan" => array (
    "physics" => 31,
    "maths" => 22,
    "chemistry" => 39
)
);

/* Accessing multi-dimensional array values */
echo "Marks for Muhammad : Physics Maths and Chemistry <br> ";
echo $marks['Muhammad']['physics'] . " , " ;
echo $marks['Muhammad']['maths'] . " , " ;
echo $marks['Muhammad']['chemistry'] . " <br>";

echo "Marks for Nadeem : Physics Maths and Chemistry <br> " ;
echo $marks['Nadeem']['physics'] . " , " ;
echo $marks['Nadeem']['maths'] . " , " ;
echo $marks['Nadeem']['chemistry'] . "<br>";

echo "Marks for Asif Khan: Physics Maths and Chemistry <br> " ;
echo $marks['Asif Khan']['physics'] . " , " ;
echo $marks['Asif Khan']['maths'] . " , " ;
echo $marks['Asif Khan']['chemistry'] . "<br>";
```

**?>**



### Sorting the Array

The **sort ()** function, the elements in an array can be sorted in alphabetical or numerical in ascending order.

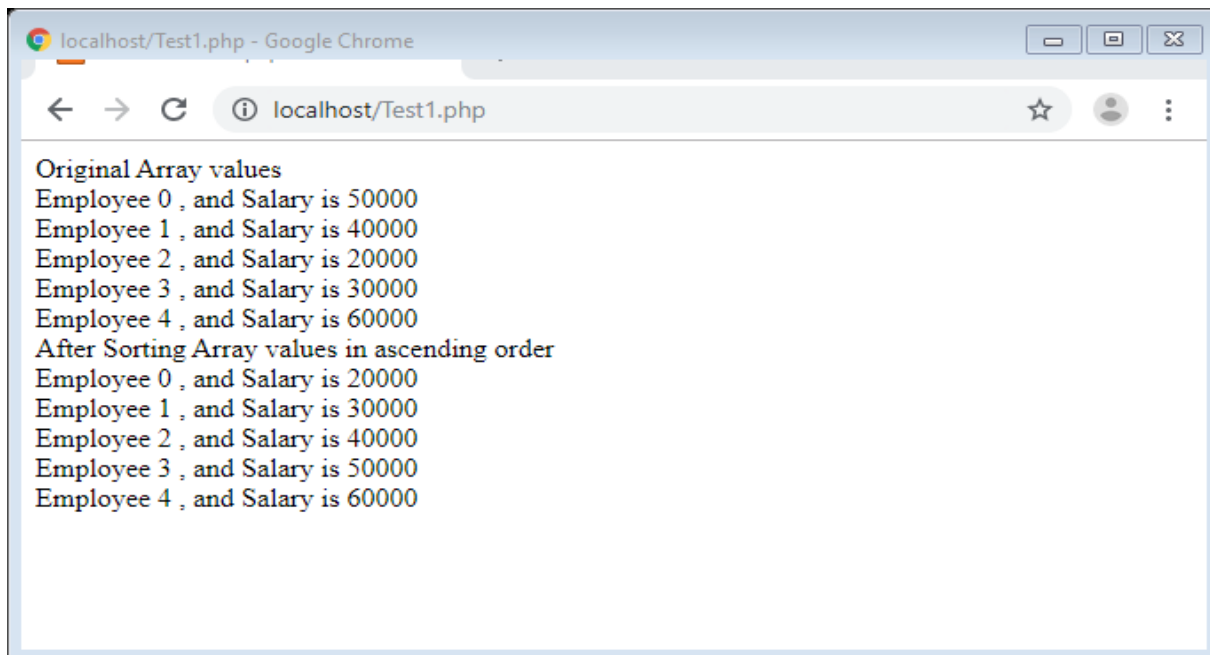
The **rsort ()** function, the elements in an array can be sorted in alphabetical or numerical, descending order.

### Example

**<?php**

```
$salary = array (50000,40000,20000,30000,60000);  
echo "Original Array values <br>";  
for ($i=0; $i<=4; $i++)  
{  
    echo " Employee $i , and Salary is $salary[$i] <br> " ;  
}  
echo "After Sorting Array values <br>";  
sort($salary);  
for ($i=0; $i<=4; $i++)  
{  
    echo " Employee $i , and Salary is $salary[$i] <br> " ;  
}
```

**?>**



A screenshot of a Google Chrome browser window displaying the output of a PHP script. The browser's address bar shows 'localhost/Test1.php'. The page content lists 'Original Array values' and 'After Sorting Array values in ascending order'. The original array contains five employee-salary pairs. The sorted array lists the same pairs in ascending order of salary.

```
Original Array values
Employee 0 , and Salary is 50000
Employee 1 , and Salary is 40000
Employee 2 , and Salary is 20000
Employee 3 , and Salary is 30000
Employee 4 , and Salary is 60000
After Sorting Array values in ascending order
Employee 0 , and Salary is 20000
Employee 1 , and Salary is 30000
Employee 2 , and Salary is 40000
Employee 3 , and Salary is 50000
Employee 4 , and Salary is 60000
```

### Remove Element from Array

The **`array_splice()`** function removes selected elements from an array and replaces it with new elements. The function also returns an array with the removed elements.

#### Syntax:

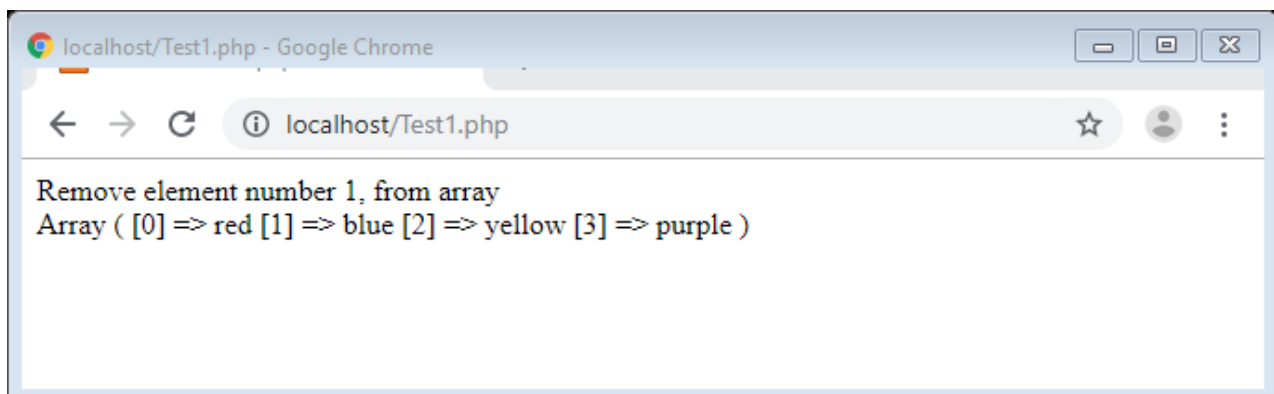
**`array_splice(array, Which_Element_To_Be_Remove, How_Many_Element_To_be_Remove)`**

#### Example

**`<?php`**

```
echo "Remove element number 1, from array <br>";
$ColorNames = array("0"=>"red", "1"=>"green", "2"=>"blue", "3"=>"yellow", "4"=>"purple");
array_splice($ColorNames, 1, 1);
print_r($ColorNames);
```

**`?>`**



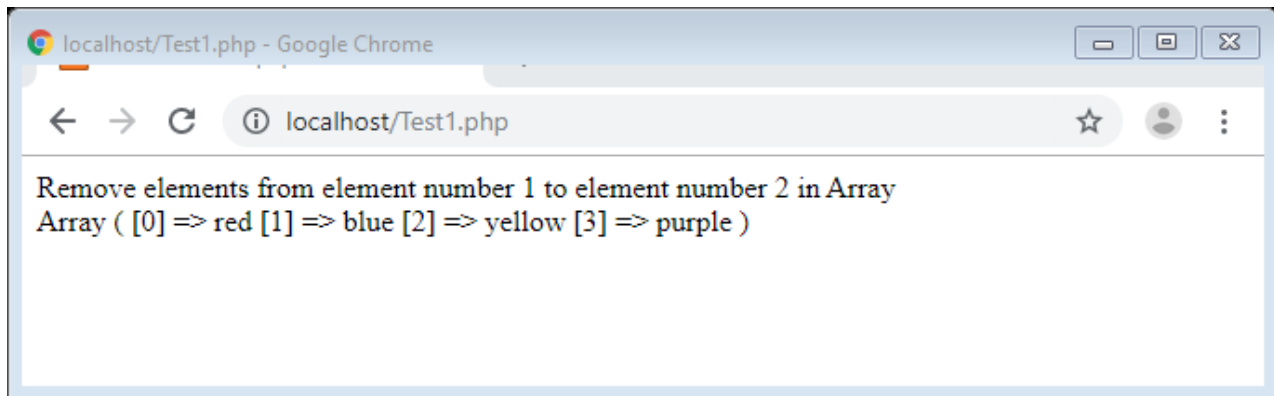
A screenshot of a Google Chrome browser window showing the output of the PHP script. The browser's address bar shows 'localhost/Test1.php'. The page content displays the message 'Remove element number 1, from array' followed by the array structure after removing the element at index 1. The output shows: 'Array ( [0] => red [1] => blue [2] => yellow [3] => purple )'.

```
Remove element number 1, from array
Array ( [0] => red [1] => blue [2] => yellow [3] => purple )
```

### Example

```
<?php
```

```
echo "Remove elements from element number 1 to element number 2 in Array <br>";  
$ColorNames = array("0"=>"red","1"=>"green","2"=>"blue","3"=>"yellow","4"=>"purple");  
array_splice($ColorNames,1,2);  
print_r($ColorNames);  
?>
```



### Exercise

#### Theory Questions

1. What is array?
2. What is the difference between a single dimensional array and a multidimensional array?
3. Describe the Associative arrays.
4. Write list of function name we can use in array.
5. What is difference between the **sort()** and **rsort()** function in PHP.

#### Practical Questions

1. What would be the output of the following Program?

**<?php**

```
$salary = array (50000,40000,20000,30000,60000);  
for ($i=4; $i>=0; $i--)  
{  
    echo " Employee $i , and Salary is $salary[$i] <br> " ;  
}
```

**?>**

2. Write a program that takes 10 integers as input and prints their sum by using array.
3. Create a Mark-Sheet for 10 students using arrays which contains the following;
  - Name
  - Seat No
  - Marks (of 3 subjects)
  - Percentage
4. Write a program to sort an integer array in ascending order.
5. Write a program that adds up two 2x2 arrays A and B stores the sum in third array C.

#### Objective MCQ's

1. The first element number start always from \_\_\_\_ index number.
  - a) 1
  - b) 0
  - c) 2
  - d) NULL
2. What is the correct syntax for declaring and initializing an associative array.
  - a) \$Flowers = array("Rose" . "Red", "Sunflower" . " Yellow");
  - b) \$Flowers = array("Rose" > "Red", "Sunflower" > " Yellow");
  - c) \$Flowers = array("Rose" >= "Red", "Sunflower" >= " Yellow");
  - d) \$Flowers = array("Rose" => "Red", "Sunflower" => " Yellow");

3. Which of the following function perform a sort on an array?
  - a) ksort()
  - b) sort()
  - c) sorting()
  - d) assort()
  
4. Which function is use to remove element from array and re-arrange index number of array.
  - a) unset()
  - b) array\_delete()
  - c) array\_splice()
  - d) delete();