Working with Arrays

Session 16

Objectives

- *Define an array*
- *Explain the use of arrays*
- *Explain the process of merging arrays*
- *Explain the use of single and multi-dimensional arrays*
- *Explain the use of array-related functions*

Introduction

- An array is a variable that can store a list of values referred by the same name

- Types of arrays are as follows:
  - Single-dimensional
  - Multi-dimensional

Array

- Is a variable that can store a set of values of the same data type
- Each element of an array can be referred by an array name and an index
- Array index
  - Is used to access an element
  - Can be a number or a string
- If index is string, then array is an associative array
- If index is number, then array is an indexed array
- By default, the index value in an array starts at zero

Initializing an Array

- Two ways of initializing an array are as follows:
  - `array()` function - assigns value to all the elements of an array
  - `array` identifier - assigns value to a specific element of an array

array() Function

- Uses key-value pairs separated by a comma to create an array
- The number of key-value pairs in the `array()` function determines the number of elements in an array

Syntax

```
$array_name = array([key => ] value, [key => ] value)
```

Where,

- **array_name** - specifies the array name
- **key** - specifies the index value of the array element
- **value** - specifies the value of the element

- Using the `array()` function, both the indexed and the associative arrays can be initialized

Indexed Arrays

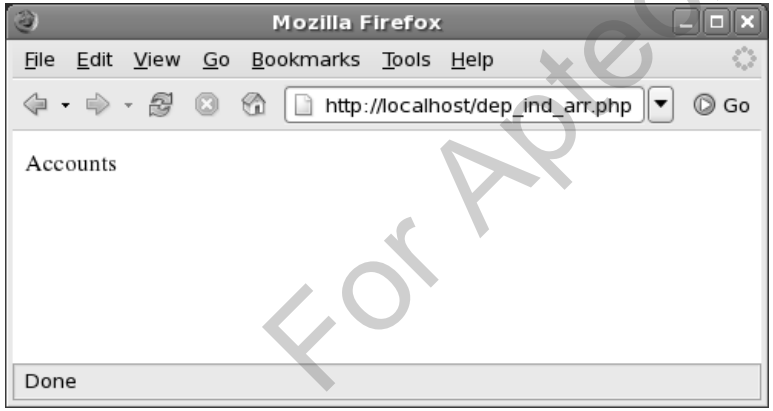
- Includes an integer as the index type
- By default, PHP creates an indexed array, if the index type is not specified at the time of creating an array
- The index value can start with any integer, such as 1, 20, or 123

- Creating an indexed array named **department**

Snippet

```
<?php
// Creating an array and storing values
$department = array (1 => 'Accounts', 2 => 'Economics',
3 => 'Computers', 4 => 'Marketing');
// Displaying the element of the array
echo $department [1];
?>
```

Displays the following output:



In the code, the array index type is an `integer`.

The department array contains four values, Accounts, Economics, Computers, and Marketing.

When the first element of the array is called, the output returned is **Accounts.**

Associative Arrays

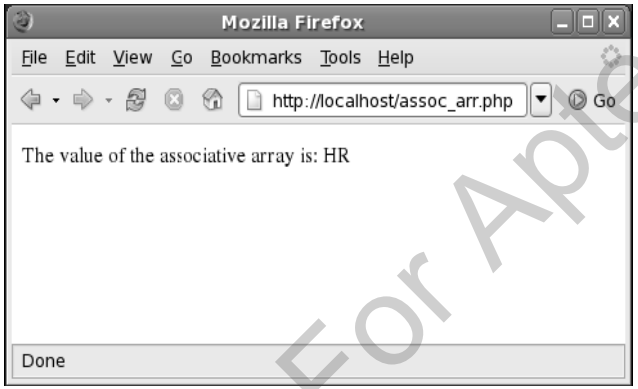
- Is an array where the index type is a string
- The index value must be specified within double quotes

- Creating an associative array named **associate**

Snippet

```
<?php
$associate = array("a" => 'Finance', "b" => 'Sales', "c" => 'HR',
                   "d" => 'Purchase');
echo "The value of the associative array is: ";
echo $associate["c"];
?>
```

Displays the following output:



In the code, the array index type is a `string`.

The index value starts from **a**. and are specified within double quotes.

The department array contains four values Finance, Sales, HR, and Purchase. The statement `echo $associate["c"];` displays the value associated with the index `"c"`.

Array Identifier

- Enables to initialize the value of a specific element in an array

Syntax

```
$array_name[key] = "element_value";
```

where,

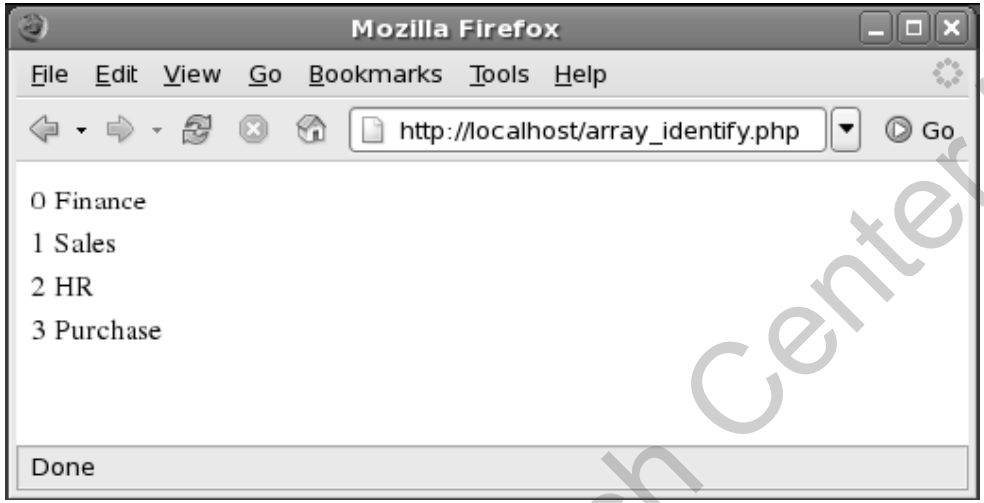
- **array_name** - specifies the name of the array
- **key** - Specifies the index value of the array element
- **element_value** - specifies the value assigned to the element of the array

- Using array identifiers to create an array named **department**

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
   {
    $rec = each($department);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
   }
?>
```

Displays the following output:



In the code, the `count()` function calculates the number of elements in the array and the result is stored in `$no_of_element` variable.

The `each()` function retrieves each key value pair of an array and stores the result in the `$rec` variable.

The `for` statement will continue to retrieve values of the array, till it reaches the last key value pair.

Merging Arrays

- The `array_merge()` function is used to combine the element values of two or more arrays

Syntax

```
$merged_array_name = array_merge($first_array, $second_array);
```

Where,

- **$merged_array_name** - specifies the name of the new array that will contain the merged element values
- **$first_array** and **$second_array** - specifies the names of the arrays whose elements are to be merged

Merging Arrays

- Demonstrating the merging of the arrays

Snippet

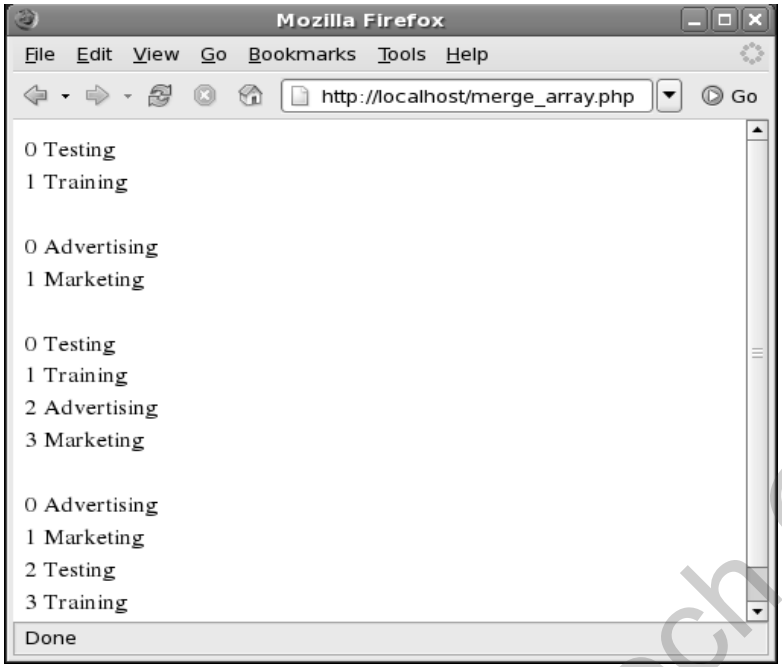
```
<?php
$ITdept = array(0 => "Testing", 1 => "Training");
$Salesdept = array(0 => "Advertising", 1 => "Marketing");
$no_of_element = count($ITdept);
for ($i=0; $i< $no_of_element; $i++)
   { $rec = each($ITdept);
     echo "$rec[key] $rec[value] ";
     echo "<br>";
   }
   echo "<br>";
   $num_of_element = count($Salesdept);
   for ($i=0; $i< $num_of_element; $i++)
    {
      $rec = each($Salesdept);
      echo "$rec[key] $rec[value] ";
     echo "<br>";
    }
    echo "<br>";
    echo "$rec[key] $rec[value] ";
```

Merging Arrays

```
$AdminDept = array_merge($ITdept, $Salesdept);
$num1_of_element = count($AdminDept);
for ($i=0; $i< $num1_of_element; $i++)
{
     $rec = each($AdminDept);
     echo "$rec[key] $rec[value] ";
     echo "<br>";
 }
echo "<br>";
$AdminDept = array_merge($Salesdept, $ITdept);
$num2_of_element = count($AdminDept);
for ($i=0; $i< $num2_of_element; $i++)
{
    $rec = each($AdminDept);
    echo "$rec[key] $rec[value] ";
    echo "<br>";
}
echo "<br>";
?>
```

Merging Arrays

Displays the following output:



In the code, the array **AdminDept, will include values,** such as **Testing**, **Training**, **Advertising**, and **Marketing**.

The element of the array that is mentioned first in the `array_merge()` function gets the first index number.

By default, PHP allots zero as the index number to the first array element.

The first element value of the next array, **Salesdept,** is allotted an index number after the first array mentioned in the function has been allotted an index number.

Multi-dimensional Arrays

- Contains one array stored within another
- In a multi-dimensional array, each element is an array
- Each element requires an array name and multiple set of indices

Syntax

```
$array_name = array(array(key => value), array(key => value));
```

Where,

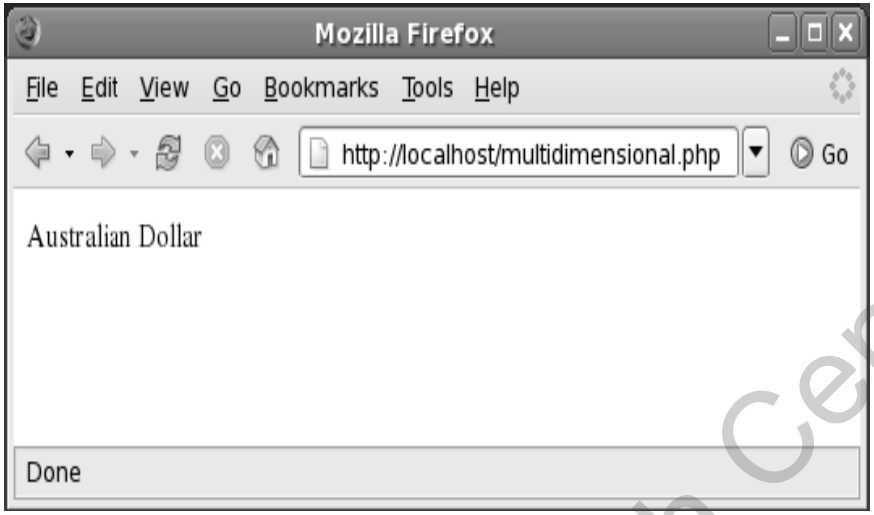
- **$array_name** - specifies the name of the multi-dimensional array
- **key** - specifies the index number of the array element
- **value** - specifies the value of the array element

- Illustrating the use of multi-dimensional arrays

Snippet

```
<?php
$country_mdlist = array(
        "USA" => array(
            "Capital" => "Washington D.C.",
            "Currency" => "US Dollar"),
        "England" => array(
            "Capital" => "London",
            "Currency" => "Pound Sterling"),
        "Australia" => array(
            "Capital" => "Canberra",
            "Currency" => "Australian Dollar"),
        "New Zealand" => array(
            "Capital" => "Wellington",
            "Currency" => "NZ Dollar"));
        echo $country_mdlist["Australia"]["Currency"];
        ?>
```

Displays the following output:



In the code, **`country_mdlist`** is a multi-dimensional associative array.

It contains key indices such as **`USA`**, **`England`**, **`Australia`**, and **`New Zealand`**.

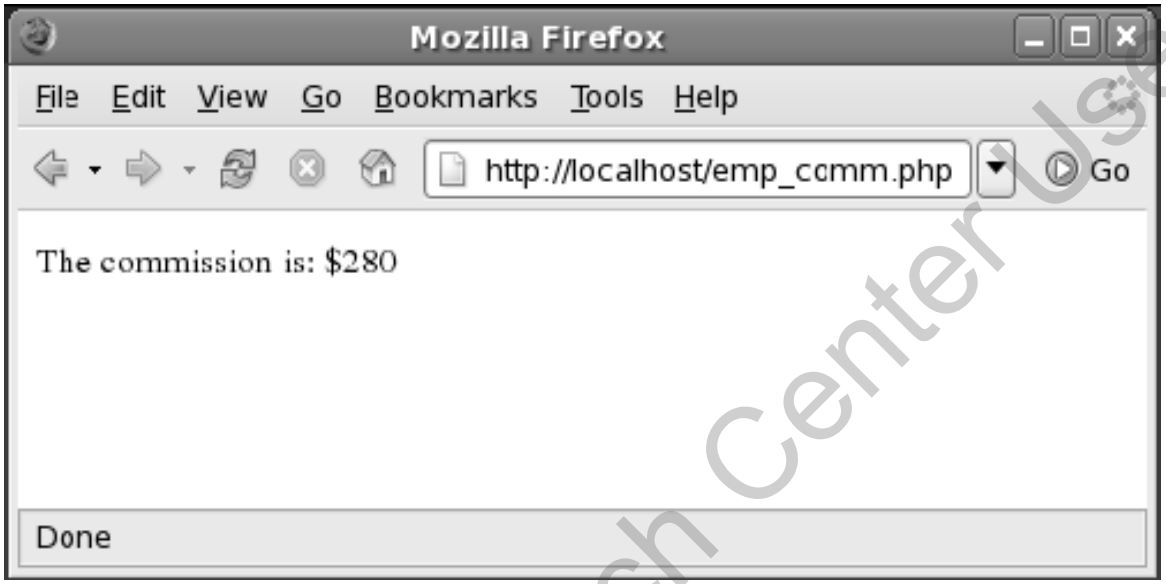
Each array element of a multi-dimensional array includes another array within it containing key indices such as **`Capital`** and **`Currency`**.

- Creating an array that stores the employee commission details

Snippet

```
<?php $employee_det = array(
      "Employee 1" => array(
            1 => "$100",
            2 => "$150",
            3 => "$100",
            4 => "$160",
            5 => "$250",
            6 => "$148"),
      "Employee 2" => array(
            1 => "$180",
            2 => "$195",
            3 => "$200",
            4 => "$130",
            5 => "$280",
            6 => "$218"));
            echo "The commission is: ";
            echo $employee_det["Employee 2"][5];?>
```

Displays the following output:



In the code, both associative and indexed indices are used to create a multi-dimensional array.

Array-Related Functions

- Some of the array-related functions supported by PHP are as follows:
  - `sort()` Function
  - `rsort()` Function
  - `arsort()` Function

sort() Function

- Arranges the element values in alphabetical order

Syntax

```
sort(ArrayName)
```

Where,

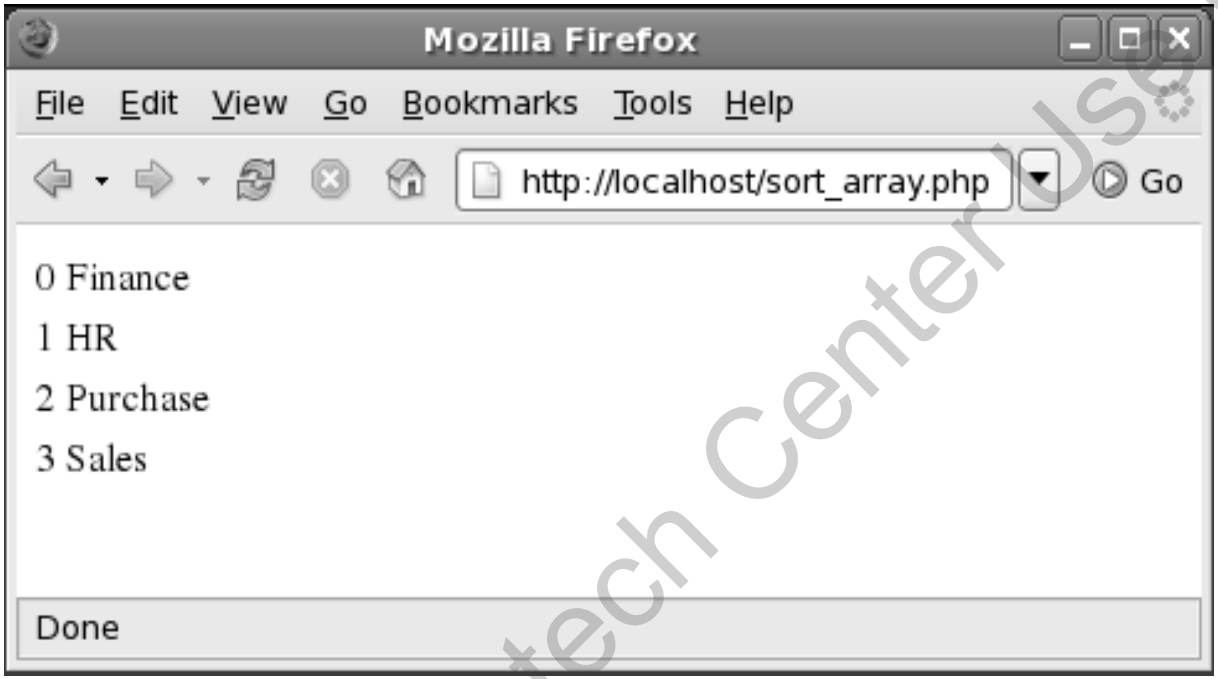
- **sort** - arranges the element values in alphabetical order
- **ArrayName** - specifies the name of the array whose elements are to be sorted

- Illustrating the sorting of the department array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
sort($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
   $rec = each($department);
   echo "$rec[key] $rec[value] ";
   echo "<br>";
   }
   ?>
```

Displays the following output:



The code displays the elements of the array in alphabetical order.

The order of the element values have changed.

However, the order of the **index values** is constant.

rsort() Function

- Sorts the element values in descending alphabetical order

Syntax

```
rsort(ArrayName)
```

Where,

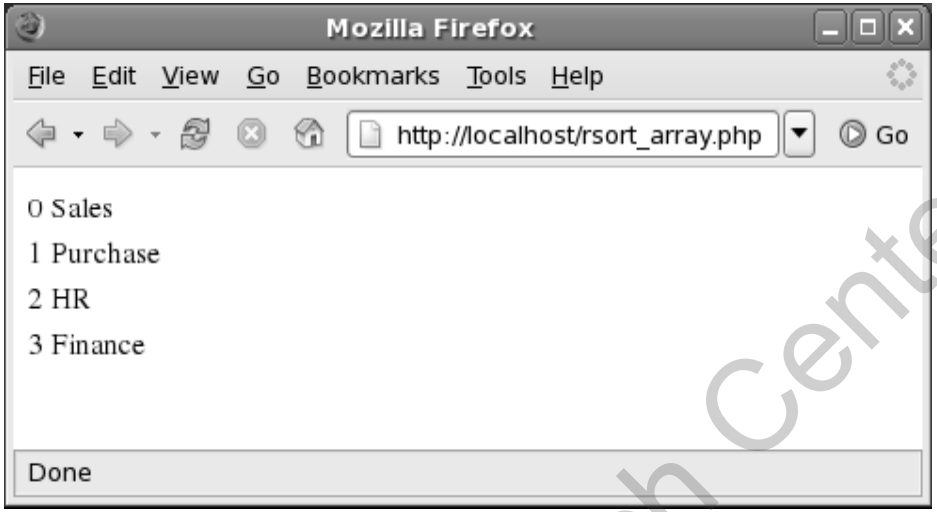
- **rsort** - arranges the element values in descending alphabetical order
- **ArrayName** - specifies the name of the array whose elements are to be sorted

- Illustrating the use of the `rsort()` function to display the values of the department array in the descending alphabetical

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
rsort($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
    $rec = each($department);
         echo "$rec[key] $rec[value] ";
         echo "<br>";
    }
    ?>
```

Displays the following output:



The code displays the elements of the array in descending alphabetical order.

The order of the element values have changed.

However, the order of the index values is constant.

arsort() Function

- Similar to `rsort()` function
- The only difference between `rsort()` and `arsort()` function is that the `arsort()` function can sort both associative and indexed arrays

Syntax

```
arsort(ArrayName)
```

- Demonstrating the use of the `arsort()` function on the array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
arsort($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
   $rec = each($department);
   echo "$rec[key] $rec[value] ";
   echo "<br>";
   }
   ?>
```

Displays the following output:

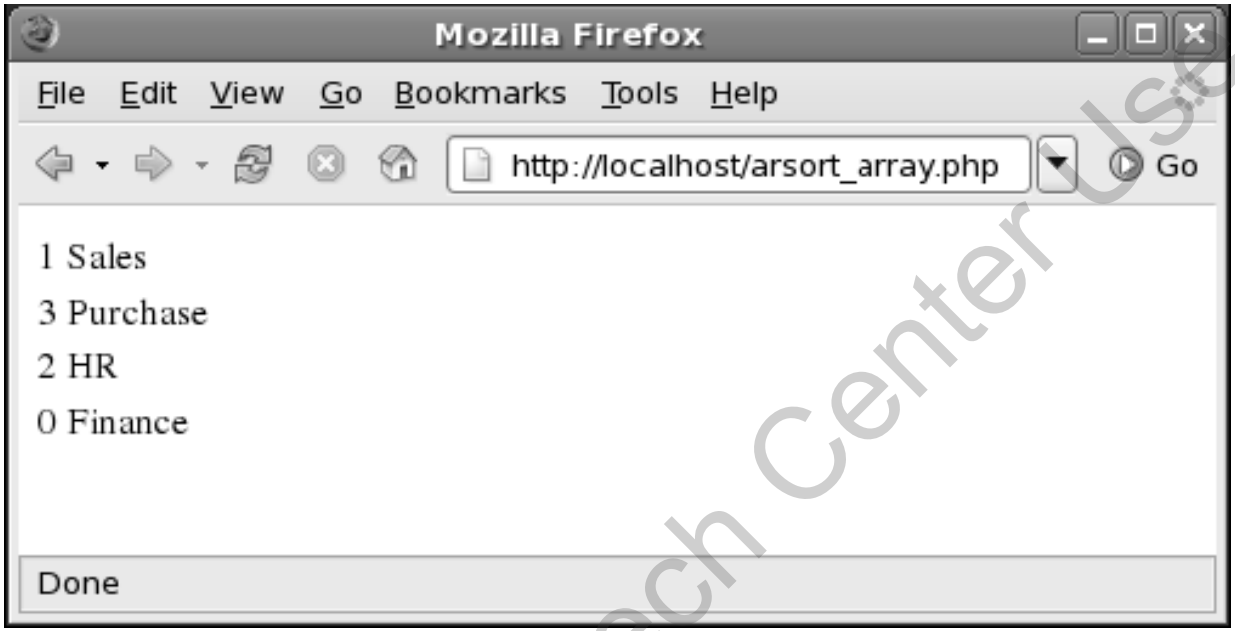


Table lists different functions to manipulate arrays

Function Name	Description
`count()`	Returns the number of elements in an array
`sizeof()`	Returns the number of elements in an array. You can use this function instead of count()
`array_count_values()`	Maintains a count of the occurrences of same element values in an array. It returns the number of occurrences of same element values
`array_flip()`	Converts the element values to index values and vice versa
`array_intersect()`	Identifies and returns the common element value among a group of arrays
`array_keys()`	Displays all the key indices of the specified array
`array_reverse()`	Reverses the order of the array elements
`array_shift()`	Returns and removes the first element of an array
`array_key_exists()`	Identifies whether or not a given key or index exists in an array
`array_push()`	Adds one or more elements to the end of an array
`array_pop()`	Pops and returns the last value of an array

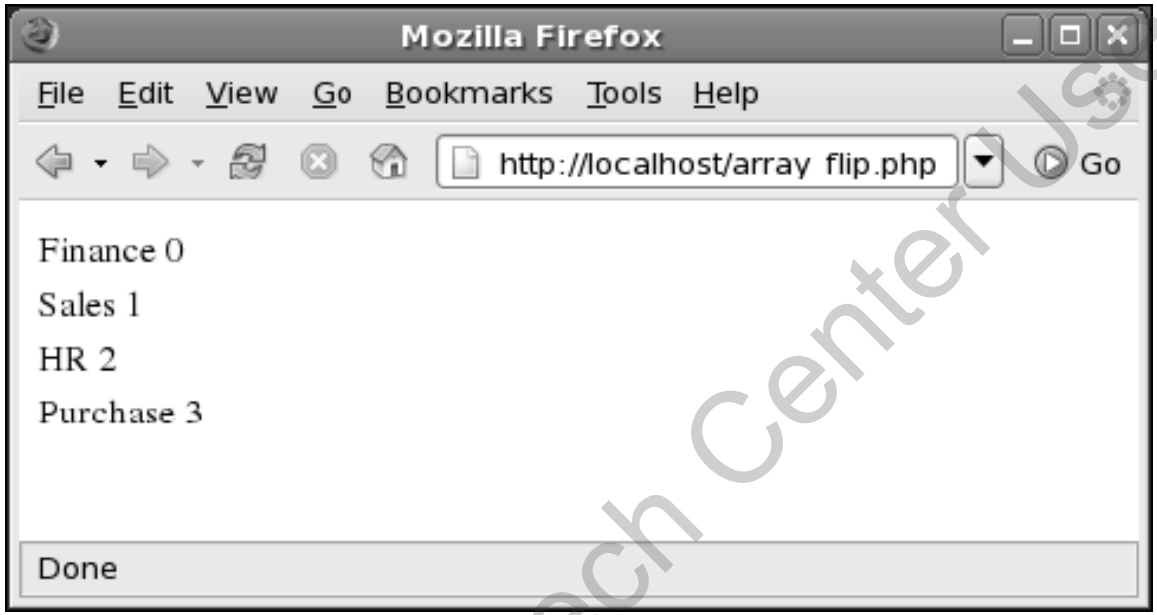
Other Array-Related Functions

- To flip the element values to the index values and the index values to the element values of the department array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$dept = array_flip($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
   $rec = each($dept);
   echo "$rec[key] $rec[value] ";
   echo "<br>";
   }
   ?>
```

Displays the following output:

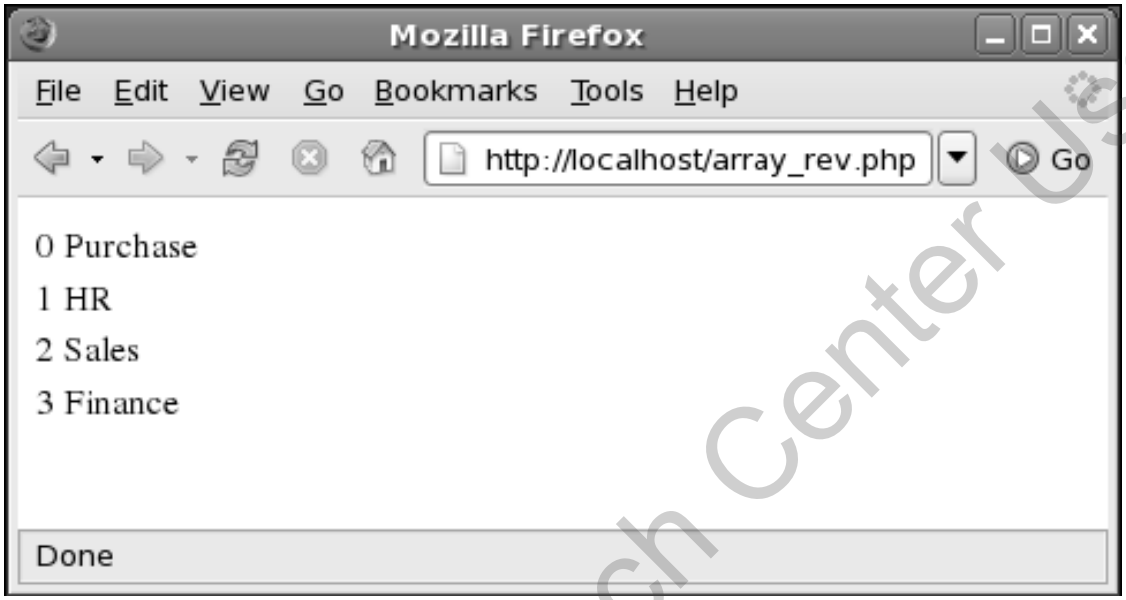


- Reversing the order of elements of the department array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$dept = array_reverse($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
   $rec = each($dept);
   echo "$rec[key] $rec[value] ";
   echo "<br>";
   }
   ?>
```

Displays the following output:

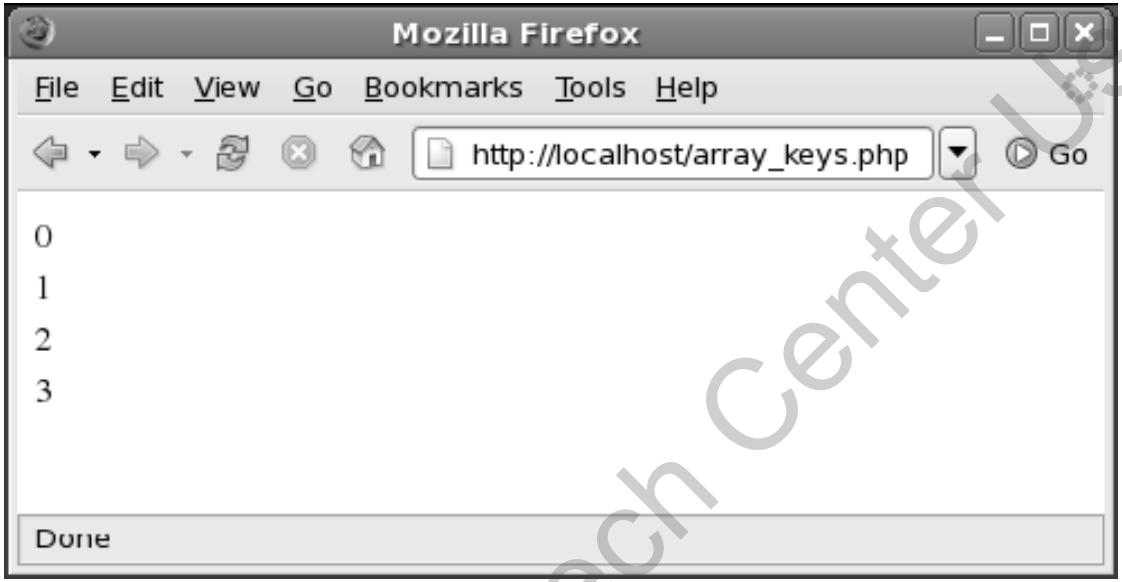


- Viewing all the key values of the department array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
$dept = array_keys($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
   $rec = each($dept);
   echo "$rec[value] ";
   echo "<br>";
   }
   ?>
```

Displays the following output:

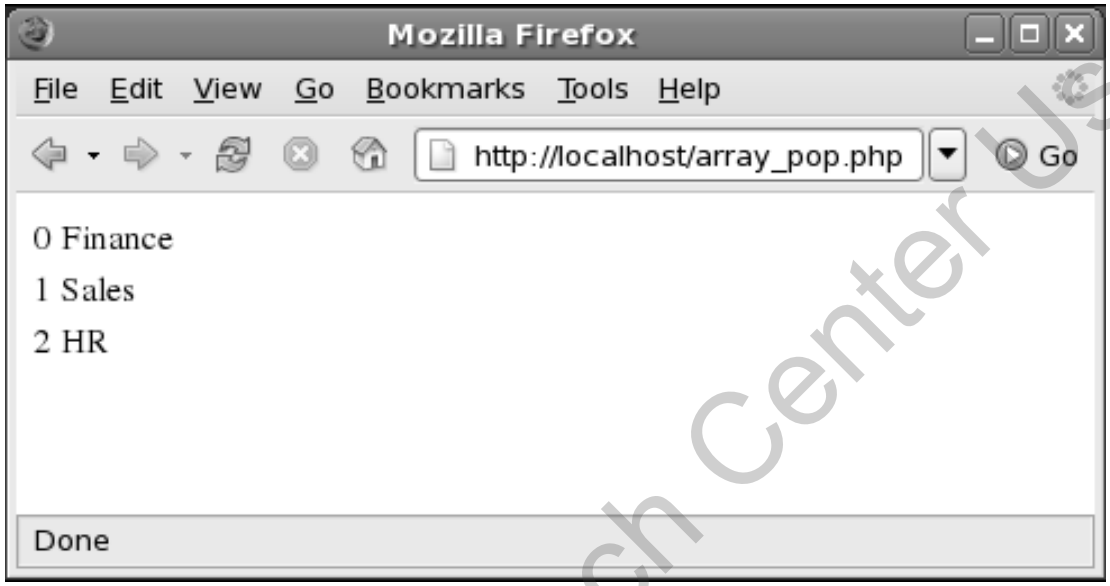


- Removing an element value from the department array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
array_pop($department);
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
   $rec = each($department);
   echo "$rec[key] $rec[value] ";
   echo "<br>";
   }
   ?>
```

Displays the following output:

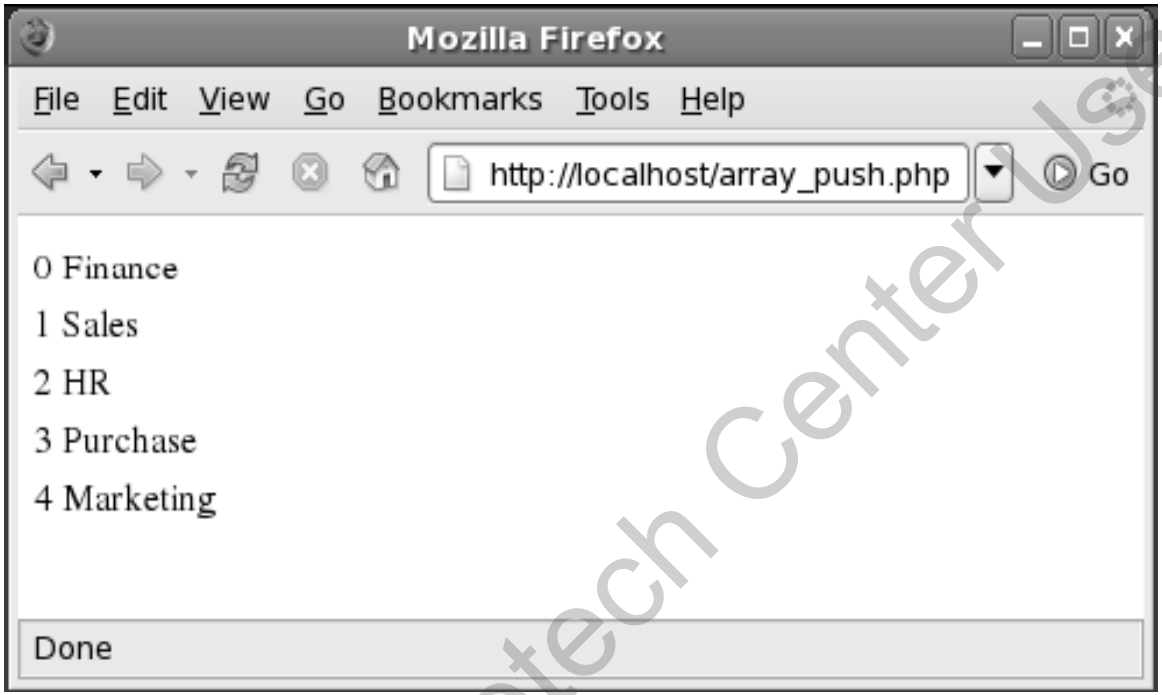


- Adding an element value to the department array

Snippet

```
<?php
$department[0]= "Finance";
$department[1]= "Sales";
$department[2]= "HR";
$department[3]= "Purchase";
array_push($department, "Marketing");
$no_of_element = count($department);
for ($i=0; $i< $no_of_element; $i++)
{
$rec = each($department);
echo "$rec[key] $rec[value] ";
        echo "<br>";
}
?>
```

Displays the following output:



Summary

- An array is a variable that can store a set of values of the same data type
- All the elements in an array are referenced by a common name
- An array index is used to access an element
- Merging arrays is the process of combining element values of two or more arrays
- In a single-dimensional array, the element includes only one level of key value pairs

Summary

- In a multi-dimensional array, each element is an array. Each element requires an array name and multiple set of indices
- An indexed array is an array where the index type is integer and an associative array is an array where the index type is string
- The sort() function arranges the element values in alphabetical order, the rsort() function sorts the element values in descending alphabetical order, and the arsort() function sorts both associative and indexed arrays