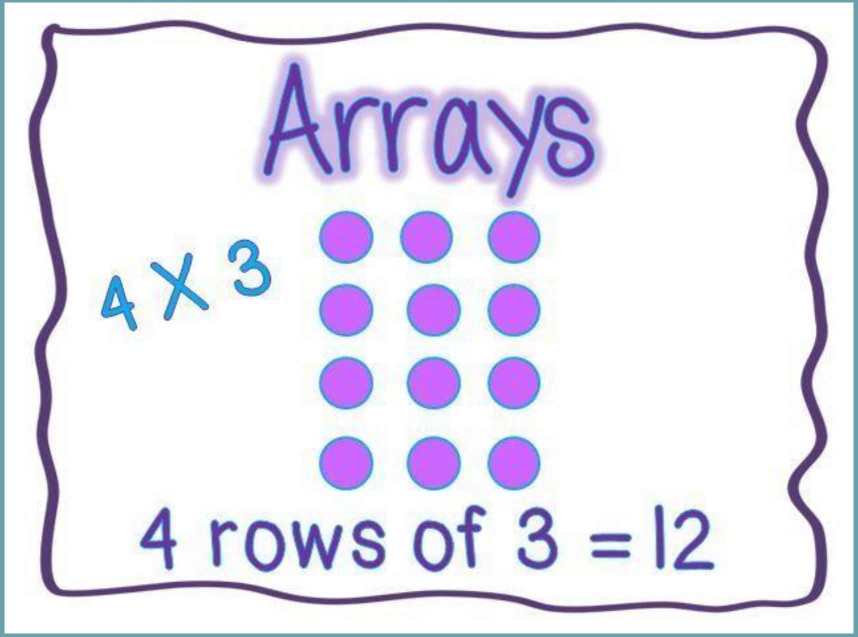
PHP ARRAYS



Department of ICT
Faculty of Technology

Arrays

- An array is a special variable, which can hold more than one value at a time
- It is a collection of data values organized as an ordered collection of key-value pairs
- Arrays store group of related data called '**Elements**'
- An array can hold many values under a single name, and you can access the values by referring to an index number
- Can store heterogeneous data in an array
  - *An array is not limited to one type of data. It can hold strings, integers, Booleans, and so on*

Examples

```
<?php
        $colors = array(“Red", “Green", “Blue");
        echo “Colors are” . $colors[0] . ", " . $colors [1] . " and " . $colors [2] ;
      ?>
```

2.

```
<?php
        $array[0] = "abc";
        $array[1] = 'a’;
        $array[2] =100;
        $array[3] =200;
        print "first ".$array[0]." second ".$array[1]." third ".$array[2]." forth
".$array[3];
?>
```

Associative Arrays

- The associative arrays are very similar to numeric arrays in term of functionality, but they are different in terms of their index.
- Associative array will have their index as string so that you can establish a strong association between key and values.

Associative Arrays

```
<?php
      $array = array( "foo" => "bar", "bar" => "foo",  100   => -100,
   -100 => 100);
      print "first ".$array['foo']." second ".$array['bar']." third“
      .$array[100]." forth ".$array[-100];
?>
```

Out put = ????

Associative Arrays

```
<?php

$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
echo "Peter is". $age['Peter’].
"old" ;
?>
```

Out put = ????

Multidimensional Array

- Multidimensional array is an array containing one or more arrays.
- In a multidimensional array, The values in an array can themselves be arrays.
- The dimension of an array indicates the number of indices need to select an element.
    - *For a two-dimensional array need two indices to select an element*
    - *For a three-dimensional array need three indices to select an element*

```
$row0 = array(1, 2, 3);
$row1 = array(4, 5, 6);
$row2 = array(7, 8, 9);
$multi = array($row0, $row1, $row2);
```

Arrays of Arrays

- The elements of an array can be many things other than a string or integer
- You can even have objects or other arrays as array elements

```
$products = array(
    'paper' =>    array(
        'copier' => "Copier & Multipurpose",
        'inkjet' => "Inkjet Printer",
        'laser' => "Laser Printer",
        'photo' => "Photographic Paper"),
    'pens' => array(
        'ball' => "Ball Point",
        'hilite' => "Highlighters",
        'marker' => "Markers"),
    'misc' => array(
        'tape'      => "Sticky Tape",
        'glue'      => "Adhesives",
        'clips' => "Paperclips")
);
echo $products["pens"]["marker"];
```

???

Traversing Arrays

- The most common task with arrays is to do something with every element
- Ex:
  - *Sending mail to each element of an array of addresses*
  - *Updating each file in an array of filenames*
- The way of traversing through an array, depends on the data and the task you’re performing

Loop Through Indexed Arrays

- You can use a for loop to count through the indices
- The for loop operates on the array itself and processes elements in key order regardless of their internal order

```
<?php
    $color=array("Red","Blue","Green");
    $arrlength=count($color);
    for($x=0;$x<$arrlength;$x++)
    {
        echo $color[$x];
        echo "<br>";
    }
?>
```

Loop Through an Associative Array

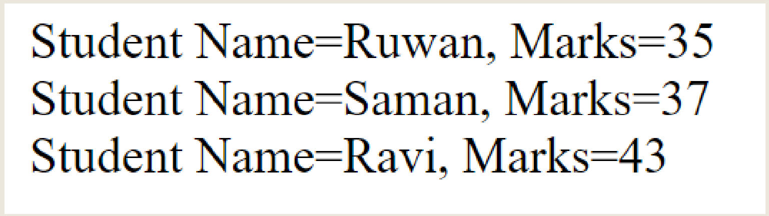
- The most common way to loop over elements of an array is to use the **foreach** construct
- Elements are processed by their internal order

```
<?php
        $marks=array("Ruwan"=>35,"Saman"=>37,"Ravi"=>43);
        foreach($marks as $x=>$x_value)
        {
                echo "Student Name=" . $x . ", Marks=" .$x_value;
                echo "<br>";
        }
?>
```

Loop Through an Associative Array

- The most common way to loop over elements of an array is to use the **foreach** construct
- Elements are processed by their internal order

```
<?php
        $marks=array("Ruwan"=>35,"Saman"=>37,"Ravi"=>43);
        foreach($marks as $x=>$x_value)
        {
                echo "Student Name=" . $x . ", Marks=" .$x_value;
                echo "<br>";
        }
?>
```

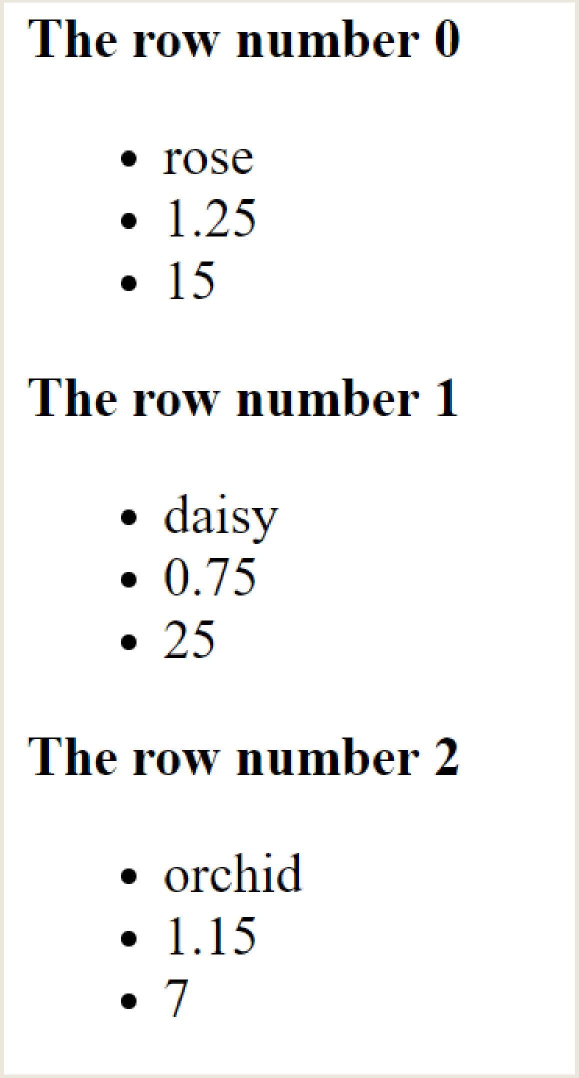


```
Student Name=Ruwan, Marks=35
Student Name=Saman, Marks=37
Student Name=Ravi, Marks=43
```

Loop Through a Multidimensional Array

```
<?php
$shop = array( array("rose", 1.25 , 15), array("daisy", 0.75 , 25),
array("orchid", 1.15 , 7) );
for ($row = 0; $row < 3; $row++) {
        echo "<b>The row number $row</b>";
        echo "<ul>";
        for ($col = 0; $col < 3; $col++) {
                echo "<li>".$shop[$row][$col]."</li>";
         }
        echo "</ul>";
}
?>
```

Output



```
The row number 0

  • rose
  • 1.25
  • 15

The row number 1

  • daisy
  • 0.75
  • 25

The row number 2

  • orchid
  • 1.15
  • 7
```

Array Functions

Function	Task
count()	Get the length of the array
current()	Returns the current element in an array
next()	Advance the internal array pointer of an array
reset()	Sets the internal pointer of an array to its first element
sort()	Sorts an array

Get the Length of an Array

- The **count()** function is used to return the length (the number of elements) of an array

```
<?php
    $cars = array("Volvo", "BMW", "Toyota");
    echo count($cars);
?>
```

Calculating the Sum of an Array

- The array_sum() function adds up the values in an indexed or associative array
  - ***$sum = array_sum(array);***
  - ***Ex:***

```
$scores = array(98, 76, 56, 80);
$total = array_sum($scores); // $total = 310
```

Inserting an Element Into the End of an Array

- The **array_push()** function inserts one or more elements to the end of an array

```
<?php
    $a=array("red","green");
    array_push($a,"blue","yellow");
    print_r($a);
?>
```

Deleting From an Array

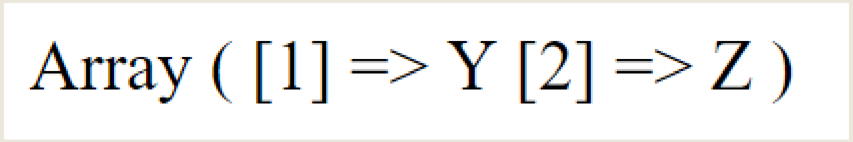
- Deleting an element from an array is just like getting rid of an assigned variable, by calling the **unset()** construct
  - *unset($array_1[2]);*
  - *unset($array_2['yellow’]);*
    - *Ex:*

```
<?php
        $anArray = array("X", "Y", "Z");
        unset($anArray[0]);
        print_r($anArray);
?>
```

Deleting From an Array

- Deleting an element from an array is just like getting rid of an assigned variable, by calling the **unset()** construct
  - *unset($array_1[2]);*
  - *unset($array_2['yellow’]);*
    - *Ex:*

```
<?php
        $anArray = array("X", "Y", "Z");
        unset($anArray[0]);
        print_r($anArray);
?>
```



```
Array ( [1] => Y [2] => Z )
```

Inspecting Arrays

Function	*Behavior*
is_array()	Takes a single argument of any type and returns a true value if the argument is an array, false otherwise
in_array()	Takes 2 arguments, the **element** you are looking for and the **array** it may be in. If the element is contained as a value in the array, it returns true, otherwise false
IsSet($array[$key])	Takes an array[key] form and returns true if the key portion is a valid key for the array

is_array()

```
<?php
$a = "Hello";
echo "a is " . is_array($a) . "<br>";

$b = array("red", "green", "blue");
echo "b is " . is_array($b) . "<br>";

$c= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43"
);
echo "c is " . is_array($c) . "<br>";

$d = "red, green, blue";
echo "d is " . is_array($d) . "<br>";
?>
```

in_array()

```
<?php
$people
= array("Peter", "Joe", "Glenn", "Cleveland");

if (in_array("Glenn", $people))
  {
  echo "Match found";
  }
else
  {
  echo "Match not found";
  }
?>
```

Questions…..