Contents

UNIT 7– PHP Basics

7.0 Structure of PHP Basics

7.1 Learning Outcomes:

After the successful completion of this unit, the student will be able to:

- Explain the basic concepts of PHP.[L2]
- Explain the output and control statements used in PHP with examples.[L2]
- Describe various string and array methods used in PHP with examples.[L2]
- Describe the use of functions and develop server-side scripts in PHP.[L5]

7.2 Introduction and Basic Syntax of PHP

PHP, which stands for "Hypertext Preprocessor," is an open-source and widely used server-side scripting language designed for web development. PHP was developed by Rasmus Lerdorf a member of the Apache Group. In 1995 Lerdorf originally named it Personal Home Page (PHP) but currently, it is known as "Hypertext Preprocessor".

It is particularly well-suited for creating dynamic and interactive web applications. PHP scripts are executed on the server, generating HTML code that is then sent to the client's web browser for rendering. PHP supports a wide range of databases and is widely used in

developing web applications using databases. PHP also supports the use of cookies and session variables in web applications.

PHP is an open source and can be downloaded from the official PHP resource: www.php.net. It runs on various platforms (Windows, Linux, Unix, Mac OS, etc) and is compatible with almost all servers like Apache and IIS.

PHP's Modes of Operation:

PHP processor has two modes of operation: Copy mode and Interpret mode

Copy mode: In this mode the PHP processor takes the PHP document as an input file, finds the XHTML code in it, and simply copies it to the output file.

Interpret mode: In this mode, the PHP processor takes the PHP document as an input file, finds the PHP script in it, interprets it, and then sends the output in XHTML form to the requesting browser which then renders it.

Basic Syntax:

PHP code is embedded within XHTML, and it is enclosed in special tags: <?php and ?>. A PHP script starts with <?php and ends with ?>. A PHP script can be placed anywhere in the document. PHP statements end with a semicolon (;).The default file extension for PHP files is ".php".

Syntax:

```
<?php
      // PHP code to be executed
?>
```

Example: Embedding PHP in XHTML document ["phpexample.php"]

```
<html>
  <head><title> Example of Embedding PHP in XHTML </title></head>
<body>
<?php
    echo "Example of Embedding PHP in XHTML <br/><br/>";
    echo "<b> Welcome to learning PHP </b> <br/><br/>" ;
    echo "<b> at </b> <br/><br/> <b> Graphic Era University <b><br/>";
?>
</body>
</html>
```

Figure 7.1: Example of embedding PHP in an XHTML document

Output:

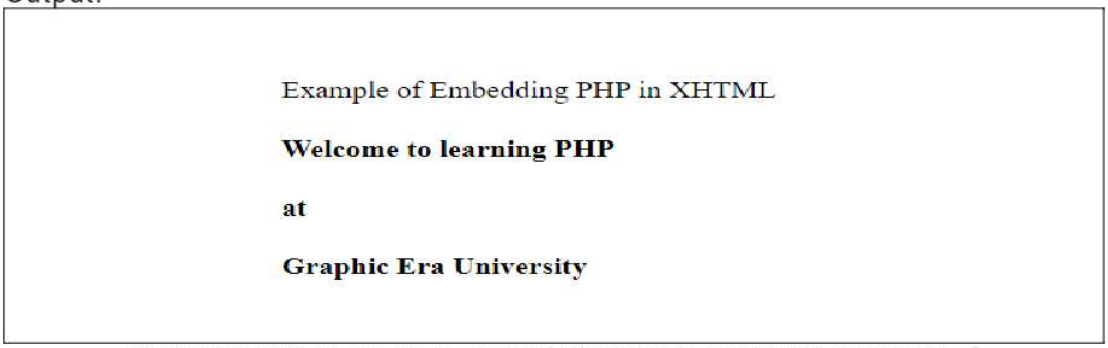

Figure 7.2: Output - Example of embedding PHP in an XHTML document

Variables, Data Types, and Operators in PHP

Variable names in PHP start with a $ sign and are case-sensitive. PHP is dynamically typed so it does not declare the type of the variable. The type of the variable is set when it is assigned a value.

For example -

```
$x = 10;          // integer type
$s = "Hello";     // string type
```

PHP has basically 3 data types

- Scalar - Boolean, Integer, Double and String
- Compound – Array and Objects
- Special – resource and NULL

An unassigned (unbound) variable has the value, NULL. The unbound variable is converted to 0 or an empty string based on the context where it is used. The `unset()` function sets a variable to NULL and the `isSet()` function is used to determine whether a variable is NULL. It returns either true or false

For example -

```
$x = 25;
$b = isset($x);      // returns true since $x has a value
unset($x);           // sets the variable $x to null
```

Arithmetic operators, and relational operators in PHP are the same as the usual operators used in other languages like – 'C', JavaScript, etc. (+, -, *, /, %, ++, --, **).

Logical operators used are – (and, or, &&, ||, ! , xor).

7.3 Output Statements

In PHP, the output statements are used to display the content on the browser. The output statements used in PHP are – echo, print, and printf().

echo statement:

The echo statement can be used with or without parenthesis: echo, echo(). It can output strings, numbers, values of variables, and results of expressions. Single or Multiple arguments can be passed to the echo statement separated by a comma. It does not return any value. It is faster than a print statement.

Example 1:

```
<?php
      echo "GEU - Example of echo Statements <br/>";
?>
```

Output:

GEU - Example of echo Statements

Example 2:

```
<?php
      $x = 4;
      $y = 8;
      echo "The sum of $x and $y is : <b> ";
      echo  $x+$y  ;
      echo "</b><br/>";
?>
```

Output:

The sum of 4 and 8 is: 12

print statement:

The print statement can be used with or without parenthesis: print, print (). It can output strings, numbers, variables, and results of expressions. A single argument can be passed to the print statement. It returns a value of 1 if the print is successful otherwise returns nothing. It is slower than an echo statement.

Example 1:

```
<?php
      print "GEU - Example of print Statements <br/>";
?>
```

Output:

GEU - Example of print Statements

Example 2:

```
<?php
        $x = 5;
        $y = 10;
        print "The sum of $x and $y is : <b> ";
        print $x+$y ;
        print "</b><br/>";
?>
```

Output:

The sum of 5 and 10 is: 15

Example 3:

```
<?php
        $p = print "<br/> print returns a value : ";
        print "$p";            //prints 1 since it is successful
 ?>
```

Output:

print returns a value: 1

printf():

The printf() function is used for formatted output. It is used to display output using format specifiers like the 'C' programming language. The format specifiers used in 'C' are supported in PHP. "%s" for strings, "%d" for decimal numbers, "%f" for floating point numbers, etc.

Example 1:

```
<?php
        $x = 15;
        $y = 10;
        $z = $x + $y ;
        printf("<br/> The sum of %d and %d is %d ", $x, $y, $z);
 ?>
```

Output:

The sum of 15 and 10 is: 25

7.4 Control Statements in PHP

PHP provides control structures and loops that allow one to manage the flow of the code and execute specific tasks repeatedly. These are fundamental to programming and are used extensively in a variety of applications.

Control Structures:

The execution of a program is controlled by the control structures based on specific conditions.

Conditional Statements (if, else if, else): Conditional statements are used to execute different code blocks depending on whether a specified condition evaluates to true or false.

if statement: If the condition is true the statements in the block are executed.

Syntax:

```
if (condition){
    // Code to execute if the condition is true
 }
```

Example:

```
<?php
      $x = 100;
      if($x > 0)
      {
           echo "<br/><b>  $x </b> is a positive number";
      }
?>
```

Figure 7.3: Snippet Code - using if statement in PHP

if else statement: If the condition is true the statements in the if block are executed otherwise the statements in the else block are executed.

Syntax :

```
if (condition) {
   // Code to execute if the condition is true
}
else {
   // Code to execute if the condition is not true
}
```

Example:

```
<?php
     $x = 99;
     if($x%2!=0){
          echo "<br/><b>  $x </b> is an odd number";
     }
     else {
          echo "<br/><b>  $x </b> is an even number";
      }
 ?>
```

Figure 7.4: Snippet Code - using if else statement in PHP

if else if statement: If the first condition is true the statements in the if block are executed otherwise the else if second condition is checked. If the (else if) second condition is true the statements in the (else if) block are executed otherwise the else block statements are executed.

Syntax:

```
if (condition) {
    // Code to execute if the condition is true
} else if (another condition) {
    // Code to execute if the first condition is false and the second is true
} else {
    // Code to execute if no conditions are true
}
```

Example:

```
<?php
       $marks = 69;
       if($marks>=90){
          echo "<br/> Grade is A ";
       }
      else if ($marks >=70) {
          echo "<br/> Grade is B ";
       }
       else { echo "<br/> Grade is C ";
       }
?>
```

Figure 7.5: Snippet of using if else if statement in PHP

Switch Statement: A switch statement allows the execution of different code blocks based on different values of a single expression. The type of control expression and case expression can be integer or string.

Syntax :

```
switch (expression)
{
    case value1:
       // Code to execute if the expression matches the value1
        break;
    case value2:
      // Code to execute if the expression matches the value2
        break;
    default:
     // Code to execute if the expression doesn't match any case
}
```

Example:

```
<?php
      $option = 3;
      switch ($option)
      {
       case 1:
              echo "Option is 1";
              break;
       case 2:
              echo "Option is 2";
              break;
      case 3:
             echo "Option is 3";
             break;
       default:
              echo "Invalid Option";
       }
?>
```

Figure 7.6: Snippet of using the switch statement in PHP

Loops:

The loops – while, for loop, do...while, in PHP are similar to other languages like 'C', JavaScript, etc.

for Loop: The for loop is used to repeatedly execute a block of code as long as a specified condition is true. The for is an entry-controlled loop.

Syntax :

```
for (initialization; condition; increment) {
    // Code to be executed in each iteration
}
```

Example:

```
<?php
    for($i=1;$i<=5;$i++) {
       echo "<br/> $i";
    }
?>
```

Figure 7.7: Snippet code - using for..loop in PHP

while Loop: The while loop executes a block of code as long as a specified condition remains true. The while loop is called an entry-controlled loop as the condition is checked first and then the statements within the loop are executed.

Syntax:

```
while (condition) {
    // Code - statements to be executed in each iteration
}
```

Example:

```
<?php
    $i= 1;
    $sum = 0;
    while ($i<=5){
       $sum = $sum + $i;
       $i= $i + 1;
    }
    echo "Sum of numbers from 1 to 5 is $sum";
?>
```

Figure 7.8: Snippet code - using while loop in PHP

do...while Loop: The do...while loop is similar to a while loop but guarantees that the code block will run/execute at least once, even if the condition is false. The do-while loop is known as an exit-controlled loop as the statements in the loop are executed first (at least once) and then the condition is checked.

Syntax:

```
do
{
    // Code to be executed in each iteration
} while (condition);
```

Example:

```
<?php
      $i = 1 ;
      $sum = 0;

      do
      {
         $sum = $sum + $i;
         $i= $i + 1;
      }while ($i<=5);

      echo "<br/> Sum of numbers from  1 to 5 is $sum ";
?>
```

Figure 7.9: Snippet code - using do..while loop in PHP

The break statement can be used to terminate the for loop, do...while, while loop, and even in the if constructs. The continue statement can be used in loop constructs to skip the remainder of the current iteration but continue execution at the beginning of the next iteration.

These control statements provide the flexibility to make decisions and repeat code execution based on certain conditions, making the PHP scripts more dynamic and responsive. Understanding how and when to use each of these constructs effectively is essential for writing structured and efficient PHP code.

7.5 Strings in PHP

In PHP, Strings are a sequence of characters. The characters are stored in the string starting with index 0. They can be stored and manipulated in various ways.

String literals are defined using either single quotes or double quotes.

- Single quotes – Special characters and escape sequences are not processed in single quotes strings. They appear the same in the output as it is typed. Even the value of the variables is not interpolated in single quoted strings.

  Example 1:

```
<?php
   $s1 = 'Welcome to GEU';
   echo "$s1";
?>
```

  Output:

  Welcome to GEU

  Example 2: The value of the variable $x is not printed when using single quotes. It appears the same as it is typed in the output.

```
<?php
   $x = 15;
   echo 'The value of the variable x is - $x';
?>
```

  Output:

  The value of the variable x is - $x

- Double quotes- Double-quoted strings allow variable interpolation and the interpretation of escape sequences. The embedded variables are replaced by their current values.

  Example 1:

```
<?php
   $x = 15;
   echo 'The value of the variable x is - $x';
?>
```

  Output:

  The value of the variable x is - 15

Example 2:

```
<?php
   $a = 10;
   $b = 20;
   $sum = $a + $b;
   echo "<br/> The sum of $a and $b is - $sum ";
?>
```

Output:

The sum of 10 and 20 is - 30

7.5.1 String Methods in PHP

PHP provides a wide range of built-in methods for working with strings. These methods allow to manipulate and extract information from strings. Some commonly used string methods with examples are discussed here.

- String Concatenation in PHP

  In PHP, strings can be concatenated using

  - dot(.) and
  - assignment(.=) operators.

  - Dot(.) operator

    In PHP, strings can be concatenated using the dot (.) operator

    Example:

```
<?php
    $str1 = "Happy";
    $str2 = "learning";
    $str3 = "PHP";
    $str4 = $str1." ".$str2." ".$str3;
    echo "<br/>The strings <b> '$str1' , '$str2',  '$str3'  </b>
        after concatenating is - <br/> <b>  $str4  </b>";
?>
```

Output:

The strings 'Happy', 'learning', 'PHP' after concatenating is -

Happy learning PHP

➢ Assignment(.=) operator

In PHP, strings can be concatenated using the assignment (.=) operator

Example:

```
<?php
    $str1  = "Happy ";
     $str2  = " learning";
     $str1 .= $str2;
     echo "<br/>The string after concatenating is-
<br/><b>$str1</b>";
?>
```

Output:

The string after concatenating is -

Happy learning

- strlen() - This function takes the string as an argument and returns the length of a string i.e., the number of characters in a string.

  Example:

```
<?php
    $str1 = "Graphic Era";
    $len = strlen($str1);  /* Finding the length of the string using strlen() */
    echo "<br/> The length of the string  '$str1' is - $len";
?>
```

  Output:

  The length of the string 'Graphic Era' is - 11

- trim() - Removes whitespace (spaces, tabs) and predefined characters from both ends of the string.

  Example 1: Removing whitespaces from both ends of the string

```
<?php
     $str1 = "   Graphic Era  ";
     trim($str1);
     echo "<br/> The string is  now -$str1-";
 ?>
```

  Output: // Whitespaces from both the ends are removed

  The string is now - Graphic Era –

Example 2: Removing predefined characters from both the ends of the string

```
<?php
 $str1 = "Graphic Era University";
 $str2 = trim($str1,"Gray");
 echo "<br/>The string after removing character from both ends is - ";
 echo "<br/><b> $str2  <b>";
?>
```

Output:

The string after removing characters from both ends is - phic Era Universit

- strtolower() - It takes the string as an argument, converts the string to a lowercase string, and returns a new string.

  Example:

```
<?php
 $str1 = "Welcome to GRAPHIC ERA";
 $str2 = strtolower($str1);
 echo "<br/> The string '$str1'converted to lowercase is <b> - $str2
       </b>";
?>
```

  Output:

  The string 'Welcome to GRAPHIC ERA' converted to lowercase is –
  welcome to graphic era

- strtoupper() - It takes the string as an argument, converts the string to an uppercase string, and returns a new string.

  Example:

```
<?php
 $str1 = "Welcome to Graphic Era";
 $str2 = strtoupper($str1);
 echo "<br/>The string '$str1' converted to uppercase is <b> - $str2
      </b>";
?>
```

  Output:

  The string 'Welcome to Graphic Era' converted to uppercase is –
  WELCOME TO GRAPHIC ERA

- strpos() - It takes as arguments 2 strings. If the second string is present in the first string, it will return the starting position of the second string in the first string otherwise it returns false.

  Example 1:

```
<?php
  $str1 = "Welcome to Graphic  Era";
  $str2 = "end";
  $n = strpos($str1,$str2);
  if ($n == false) {
       echo "<br/> The string '$str2' is not found in '$str1' ";
  }
  else
  {
      echo "<br/> The string '$str2' is present in  '$str1' at
            position $n";
   }
?>
```

  Output:

  The substring 'end' is not found in 'Welcome to Graphic Era'

  Example 2:

```
<?php
 $str1 = "Welcome to Graphic  Era";
 $str2 = "Graph";
 $n = strpos($str1,$str2);
 if ($n == false) {
    echo "<br/> The substring '$str2' is not found in '$str1'";
 }
 else
 {
    echo "<br/> The string '$str2' is present in  '$str1' at position
          $n";
 }
?>
```

  Output:

  The string 'Graph' is present in 'Welcome to Graphic Era' at position 11

- strrev() – It takes a string as an argument and returns the reverse of the string. The original string remains unaffected.

  Example:

```
<?php
 $str1 = "Graphic";
 $str2 = strrev($str1);
 echo "<br/>The string '$str1' after reversing is <b> - $str2  </b>";
?>
```

  Output:

  The string 'Graphic' after reversing is - cihparG

- str_replace() – This function takes 4 arguments.

  Syntax –

  str_replace($search_str, $replace_str, $target_str, $count);

  Here,

  $search_str - It is a string to be searched

  $replace_str - It is a new string value used for replacement

  $target_str - It is a string where a search operation is carried out

  $count - It gives the number of occurrences that are replaced

  This function replaces all occurrences of the first argument with the second argument in the third argument and returns a new string. The original string is unaffected, it remains the same.

  Example:

```
<?php
  /* Replacing a substring */
  echo "<br/> Replacing a substring";
  $str1 = "JavaScript";
  $str2 = "PHP";
  $str3 = "Welcome to learning JavaScript";
  $str4 = str_replace($str1,$str2,$str3);
  echo "<br/>The original string is  <b> '$str3' </b> ";
  echo "<br/> Here <b> $str1 </b>  is being replaced with <b> $str2
        </b>";
  echo "<br/> After replacing, the new string is <b> - $str4  </b>";
  echo "<br/> Original string remains unaffected - <b> $str3 </b> ";
?>
```

Output:

Replacing a string

The original string is 'Welcome to learning JavaScript'

Here JavaScript is being replaced with PHP

After replacing, the new string is - Welcome to learning PHP

The original string remains unaffected - Welcome to learning JavaScript

- lcfirst() - It converts the first character of a string to lowercase

Example:

```
<?php
   $str1 = "Welcome to Learning PHP";
   $str2 = lcfirst($str1);
   echo "<br/>The string <b> '$str1'  </b> after conversion is –
         <br/> <b>  $str2  </b>";
?>
```

Output:

The string 'Welcome to Learning PHP' after conversion is - welcome to Learning PHP

- ucwords() – It converts the first character of each word in a string to uppercase

Example:

```
<?php
   $str1 = "welcome to learning pHP, gEU";
   $str2 = ucwords($str1);
   echo "<br/>The string <b> '$str1'  </b> after conversion is –
         <br/> <b> $str2  </b>";
?>
```

Output:

The string 'welcome to learning pHP, gEU' after conversion is –

Welcome To Learning PHP, GEU

7.6 Arrays in PHP

Arrays in PHP, are different from those of any other programming languages like 'C' and C++. They are a combination of typical indexed arrays like other languages and associative arrays, or hashes found in some other languages like Perl, Python, Ruby, etc.

Arrays in PHP consist of two parts, a key and a value. i.e. a Key-Value pair.

Arrays in PHP can have some elements with integer keys and some with string keys.

Here are some key concepts and examples related to arrays in PHP:

Creation of Arrays:

In PHP, arrays can be created in 2 ways -

- Assignment operator
- Array Construct

- Creating arrays using an assignment operator (Indexed Arrays)

Arrays can be created in PHP using square brackets [] and assigning a value to an array using an index. They are called indexed arrays.

Example 1: `$ary[0] = 10;`

Here the array - "ary" is created with a value of 10 at index 0

Example 2: `$ary[4] = 50;`

In the array - "ary", value 50 is stored at index 4

If empty brackets are used in an assignment, then the subscript is used implicitly. If the array has no elements, then subscript zero "0" is used otherwise highest subscript + 1.

Example 3:

```
$ary[]   = 20;     // value 20 is stored at index 0
```

Example 4:

```
$ary[2] = 30;      // value 30 is stored at index 2
$ary[] = 40;       // value 40 is stored at index 3 i.e. highest subscript + 1 (2 + 1)
```

- Creating an array using Array Construct (Associative Arrays)

Arrays can also be created in PHP using array() construct and populating them with values. When the array() constructor has many values, they are considered, elements of the array with keys 0,1,2,3, and so on. They are called as associative arrays.

Example 1: Creating an empty array using an array() construct

```
$ary = array();
```

Here, the array - "ary" is empty.

Example 2: Creating an array of 4 elements

```
$ary = array(10,20,30,40);
```

Here, the array - "ary" is created with 4 elements with values 10,20,30, and 40 with keys 0,1,2, and 3 respectively.

Example 3: Creating an array with key and value pair (numeric keys)

```
$ary = array(11=>20,12=>25,13=>42,14=>56);
```

Here the array - "ary" is created with numeric keys 11,12,13, and 14.

Example 4: Creating an array with key and value pair (string keys)

```
$ary = array("Jim"=>40, "Joe"=>60, "Mary"=>35);
```

Here the array - "ary" is created with string keys "Jim", "Joe", and "Mary"

Example 5: Creating an array with key and value pair (mixed numeric and string keys)

```
$ary = array("Jim"=>40, 4=>60, "Mary"=>35,5=>"Joe");
```

Here the array - "ary" is created with string and numeric keys "Jim", 4, "Mary", and 5. Arrays in PHP can contain a mixture of both data types numeric and strings. They need not be purely like traditional arrays or hashes.

Accessing Array Elements:

Individual elements in the array can be accessed using their keys.

Example :

```
$ary = array("Jim"=>40, "Joe"=>60, "Mary"=>35);
$x = $ary['Mary'];          // variable x will have the value 35
```

Modifying an element in an array:

The value of an element in an array can be changed by assigning a new value to it.

Example :

```
$ary = array("Jim"=>40, "Joe"=>60, "Mary"=>35);
$ary['Jim'] = 100;
```

The value of the element whose key is "Jim" is set to 100

Inserting an element in an array:

Example :

```
$ary = array("Jim"=>40, "Joe"=>60, "Mary"=>35);
$ary['Jone'] = 95;
```

The element with key "Jone" and value 95 is inserted in the array - "ary". The array will now contain - "Jim"=>40, "Joe"=>60, "Mary"=>35, "Jone"=>95,

Adding/Inserting and Removing/Deleting Elements:

Arrays have built-in methods for adding and removing elements – array_push(), and array_pop(). Using these methods provides a simple way to implement a stack in an array.

array_push(): Adds one or more elements to the end of an array. The first parameter of the method `array_push()` is the name of the array and there can be any number of additional parameters.

Example: One element is being inserted at the end of the array

```
$ary = array("Jim", "Joe");
$ary = array_push($ary, "Mary");
```

Now, array "ary" consists of ("Jim", "Joe", "Mary")

array_pop(): Removes and returns the last element from an array. If the array is empty, it returns null. This method takes a single parameter i.e. the name of the array.

Example:

```
$ary = array("Jim", "Joe", "Mary");
$x = array_pop($ary);
```

Now, array "ary" consists of ("Jim", "Joe");

Functions/methods for dealing with arrays in PHP

1. unset() - It is used to delete the whole array or an element of the array

   Example: `$ary = array("Joe" => 42, "Mary" =>41, "Jim"=>35);`

   Deleting an element with key ["Mary"] - unset($ary["Mary"]);

   Deleting a whole array - unset($ary);

2. array_keys() - It takes an array as its parameter and returns an array of the keys of the given array

   Example:

   ```
   $tempday = array("Mon"=>35,"Tue"=>32,"Wed"=>28,"Thu"=>27,"Fri"=>26);
   $days = array_keys($tempday);        // returns an array of keys
   ```

   Now $days consist of ("Mon", "Tue", "Wed", "Thu", "Fri"); with keys as 0,1,2,3,4

3. array_values() - It takes an array as its parameter and returns an array of the values of the given array

   Example:

   ```
   $tempday = array("Mon"=>35,"Tue"=>32,"Wed"=>28,"Thu"=>27,"Fri"=>26);
   ```

```
$val = array_values($tempday);          // returns array of values
```
Now $val consist of (35,32,28,27,26) with keys as 0,1,2,3,4

4. array_key_exists() - The existence of an element of a specific key can be determined by this function. It returns a Boolean value true if it exists otherwise false
   Example: Check whether an element with the key "Tue" exists in the given array

```
$tempday = array("Mon"=>35,"Tue"=>32,"Wed"=>28,"Thu"=>27,"Fri"=>26);
if array_key_exists("Tue", $tempday) {
     $x = $tempday["Tue"];
     print "Temperature on Tue is $x ";
}
```

   Here, the method `array_key_exists("Tue", $tempday)` returns true, since key `"Tue"` is present in the array and the statements in the if condition are executed.
   The Output is

   Temperature on Tue is 32

5. is_array() - It takes a variable as a parameter and returns TRUE if the variable is an array, FALSE otherwise
   Example 1 :

```
$tempday = array("Mon"=>35,"Tue"=>32,"Wed"=>28,"Thu"=>27,"Fri"=>26);
$x = is_array($tempday);
```

   Since `$tempday` is an array, it returns true.

   Example 2 :

```
$y = 10;
$x = is_array($y);
```

   Since `$y` is not an array, it returns false.

6. in_array() - It takes two parameters, an expression, and an array, and returns TRUE if the value of the expression is a value in the array, FALSE otherwise
   Example:

```
$tempday = array("Mon"=>35,"Tue"=>32,"Wed"=>28,"Thu"=>27,"Fri"=>26);
$x = in_array(28, $tempday);
```

   It returns TRUE if value 28 is present in the array otherwise FALSE

7. sizeof() - it determines the number of elements in the array

   Example:

```
$tempday = array("Mon"=>35,"Tue"=>32,"Wed"=>28,"Thu"=>27,"Fri"=>26);
$len = sizeof($tempday);    // returns 5 as the no. of elements present are 5
```

8. explode() - It explodes a string into substrings and returns them in an array. It takes two parameters. The delimiters of the substrings are defined by the 1st parameter and 2nd parameter is a string to be converted

   Example: A String is converted into an array

```
$str = "Graphic Era University";
$words = explode(" ",$str);                    //$words is an array
```

   Now $words consist of ("Graphic", "Era", "University")

9. implode() - It does the inverse of explode. It catenates the elements of the array together using the separator string between the elements and returns the result as a string.

   Example: An array is converted into a string

```
$words = array("Graphic", "Era", "University") ;
$str = implode(" ",$words);
```

   Now the string $str contains "Graphic Era University";

10. count() - It returns the number of items in the variable. If it is an array it returns the number of elements, if scalar it returns 1.

    Example 1:

```
$tempday = array("Mon"=>35,"Tue"=>32,"Wed"=>28,"Thu"=>27,"Fri"=>26);
$n = count(($tempday);     // $n contains 5 the number of elements
```

    Example 2:

```
$x = 100;
$n = count(($x);           // $n contains 1 since $x is scalar
```

In PHP, the arrays are internally stored as a linked list where each cell consists of a key and a value pair. The array elements can be accessed by using a hash function. The logical structure of an array in PHP is shown in Figure 7.10

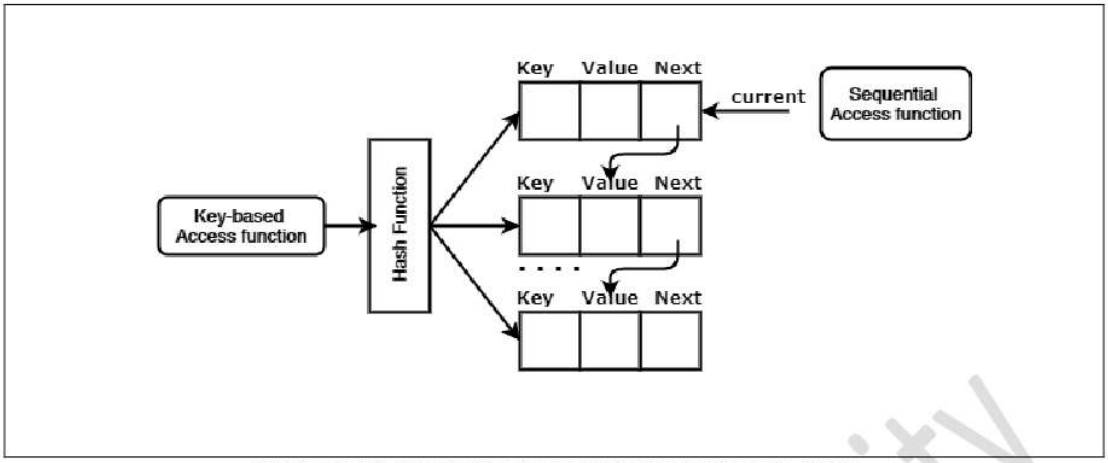

Figure 7.10: Internal logical structure of arrays in PHP

7.6.1 Sequential Access of Array Elements in PHP

In PHP, there are different ways of accessing array elements sequentially. There is a marker or internal pointer called the "current pointer" for each array that can be used to reference elements of the array. By default, the "current pointer" at the time of array creation is initialized to reference the 1st element of the array.

The methods that can be used for sequential access are - current(), next(), each(), prev(), end(), reset(), key()

- current() - It returns the element at the current position since by default the "current pointer" references the 1st element on an array.

  Example:

```
$cities = array("Mumbai", "Delhi", "Bangalore", "Pune");
$city = current($cities);
print(" The first city is $city  <br/>");
```

  Output of the code:

  The first city is Mumbai

- next() - It moves the pointer to the next array element and returns the value of that element. It returns FALSE when the end of the array is reached.

  Example :

```
$cities = array("Mumbai", "Delhi", "Bangalore", "Pune");
$city = current($cities);
print(" The City is  -  $city  <br/>");
```

```
while($city = next($cities))
    print(" The City is  -  $city  <br/>");
```

Output of the code:

The City is - Mumbai

The City is - Delhi

The City is - Bangalore

The City is - Pune

- each() - returns the element being referenced by the "current pointer" and then moves the pointer.

  It returns an array of two elements consisting of the key and value of the current element. The keys of the two elements are the string "key" and "value".

  Example :

```
$empsalaries= array("Jone"=> 48000,"Jerry"=>50000,"Jim" => 39000);
while($empsal = each($empsalaries))
{
     $name = $empsal["key"];
     $salary = $empsal["value"];
     print("The  salary of  $name is - $salary <br/>");
 }
```

  Output of the code:

  The salary of Jone is - 48000

  The salary of Jerry is - 50000

  The salary of Jim is - 39000

- prev() - The current pointer can be moved backward with this function. i.e. to the previous element of the current pointer.

  Like the next() function, the prev() function returns the value of the element being referenced after the pointer has been moved.

- end() – The current pointer can be set to the last element of the array using the end() function.

- reset() - The current pointer can be set to the first element of the array using the reset() function.

- key() - It returns the key of the current element of the array.

  Example:

```
$empsalaries= array("Jone"=> 48000, "Jerry"=>50000, "Jim" => 39000);
$x = key($empsalaries);
print("The key is  - $x <br/>");
```

  Output of the code:

  The key is - Jone

  ** Assuming the current pointer is pointing to 1st element of the array

Iterating Through Arrays Sequentially:

To loop through the elements in an array sequentially, the foreach() loop can be used.

The foreach() loop has 2 forms -

- foreach(array as scalar_variable) loop body
- foreach(array as key=>value) loop body

- foreach(array as scalar_variable) loop body

  The array value is set to the scalar variable during each iteration of the loop. The current pointer is implicitly reset to point to the 1st element of the array before the first iteration.

  Example:

```
$cities = array("Mumbai", "Delhi", "Bangalore", "Pune");
foreach($cities as $x)
    print(" $x <br/>");
```

  Output of the code:

  Mumbai
  Delhi
  Bangalore
  Pune

- foreach(array as key=>value) loop body

  This loop provides both the key and value of each element of the array.

  Example:

```
$empsalaries= array("Jone"=> 48000, "Jerry"=>50000, "Jim" => 39000);
foreach($empsalaries as  $k => $v)
          print("The salary of $k is - $v  <br/> ");
```

Here $k represents the key and
$v represents the value

Output of the code:

The salary of Jone is - 48000
The salary of Jerry is - 50000
The salary of Jim is - 39000

7.7 Functions in PHP

Functions in PHP are similar to functions in JavaScript. Functions in PHP are blocks of code that can be defined to perform specific tasks. Functions enable code reuse and organization. Defining functions reduces the time required to write many lines of code to perform similar tasks each time. It makes the programs compact due to less coding. Functions enhance code organization, readability, and reusability.

Defining Functions:

Functions are defined in PHP using the function keyword. A function consists of a function header and statements that describe the actions to be performed. Functions can take parameters, execute a block of code, and return a value.

Syntax:

```
<?php
    /* Declaring and defining a function */
    function function-name([arg1, arg2, arg3, .... argN]){
        // executable code
        // return value if any
    }

    /* calling a function without returning a value */
       function-name([arg1, arg2, arg3, .... argN]);
    /* calling a function that returns a value */
     $x = function-name([arg1, arg2, arg3, .... argN]);
?>
```

Functions may have 0 or more arguments. Arguments are optional.

PHP function without parameters/arguments

Functions can be defined without parameters(input values) to perform some actions.

Example:

```
<?php
      /* Defining a function in PHP */
      function displayMessage()
      {
        echo "GEU - Displaying message using a function in PHP !";
      }
      /* Calling a PHP Function */
      displayMessage();
 ?>
```

Figure 7.11: Simple function in PHP for displaying a message

Output:

GEU - Displaying message using a function in PHP !

Figure 7.12: Output - Simple function in PHP for displaying a message

PHP function with parameters/arguments

Functions can accept parameters(input values) to perform actions or calculations based on those values.

Example:

```
<?php
       /* Defining a function in PHP that takes arguments */
      function sumofnos($a, $b)
      {
            $sum =  $a + $b;
            echo "Sum of  $a and $b is :-  $sum" ;
      }
       /* Calling a PHP Function */
       sumofnos(10, 20);       // function call
 ?>
```

Figure 7.13: Simple function with arguments in PHP

Output:

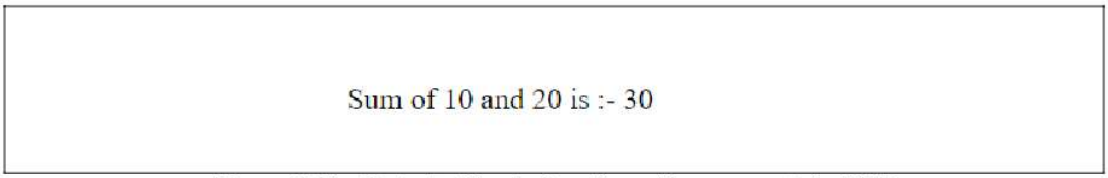

Figure 7.14: Output - Simple function with arguments in PHP

PHP function that returns a value

Functions can return values using the return statement. This returned value can be used in other parts of the code.

Example:

```
<?php

      /* Defining a function in PHP that returns a value */
      function sumofnos($a, $b)
      {
             $sum =  $a + $b;
             return $sum ;
      }
      /* Calling a PHP Function */
       $a = 50;
       $b = 60;
       $sum = sumofnos($a, $b);       // function call

       echo "Sum of  $a and $b is :-  $sum" ;

 ?>
```

Figure 7.15: Simple function that returns a value in PHP

Output:

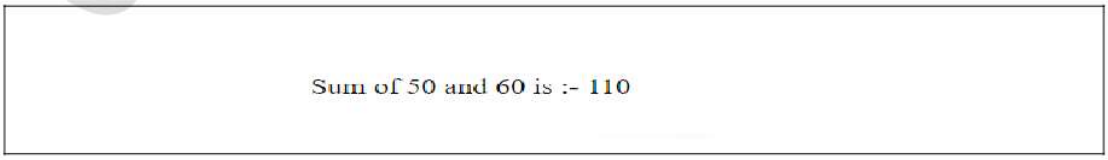

Figure 7.16: Output - Simple function that returns a value in PHP

7.7.1 Passing Parameter to Function [Pass by value and reference]

Passing parameters to functions

There are 2 ways of passing arguments to a function

- Pass by Value
- Pass by Reference

Pass by Value: By default, arguments are passed to a function as Pass by Value. Any changes made to the arguments in the function will not change(affect) the value of the original variable. The changes made to the arguments within the function are not reflected outside the function.

Example:

```
<?php
   // pass by value
   function passbyvalue($x)
   {
      $x += 5;
      echo "The changed value in function[passbyvalue] is <b>
            $x </b><br/>";
   }
      $a = 100;
      echo "The original value is <b> $a </b><br/>";
      passbyvalue($a);
      echo "The original value after a function call [pass by value] is
            still <b> $a </b><br/>";
?>
```

Figure 7.17: Functions called using pass-by-value in PHP

Output:

The original value is **100**
The changed value in function[passbyvalue] is **105**
The original value after a function call [pass by value] is still **100**

Figure 7.18: Output - Functions called using pass-by-value in PHP

Pass by Reference: Argument can be passed to a function as Pass by Reference. Any changes made to the arguments in the function will change(affect) the value of the original variable as the address of the variable is passed to the function by using the (&) sign in the function definition.

Example:

```
<?php

      /* pass by reference */
      function passbyref(&$x)  // & ampersand is used before the variable
      {
            $x += 5;
            echo "The changed value in function [passbyref] is  <b>
                  $x </b><br/>";
      }

      $a = 100;
      echo "The original value is <b> $a </b><br/>";

      passbyref($a);

      echo "The original value after a function call [pass by reference]
            changes to <b> $a </b><br/>";
?>
```

Figure 7.19: Functions called using pass-by-reference in PHP

Output:

The original value is **100**
The changed value in function [passbyref] is **105**
The original value after a function call [pass by reference] changes to **105**

Figure 7.20: Output - Functions called using pass-by-reference in PHP

7.7.2 Functions – Scope of variables [local and Global]

Functions - Scope of Variables: Local and Global

Variables declared within a function are locally scoped and are not accessible outside the function unless explicitly returned. Variables declared outside functions are global and can be accessed within the function by using the keyword global.

Example: Scope of local and global variables in functions in PHP

```
<?php
    // Example of Local and global variables
    $n = 20;       //global variable
    function display()
    {
      global $n;
      $a = 10;           // local variable in function
      $n = $n + 1;       // global variable accessed in function
      echo "Value of local variable a in function is –
           <b> $a </b> <br/>";
      echo "Value of global variable n in function after increment is
          - <b> $n </b> <br/>";
   }
   echo "The global variable value outside function is –
        <b> $n </b><br/>";
   display();
   echo "The global variable value after a function call is - ";
   echo " <b> $n </b><br/>";

?>
```

Figure 7.21: Program to demonstrate the scope of local and global variables in PHP

Output:

The global variable value outside function is - **20**
Value of local variable a in function is - **10**
Value of global variable n in function after increment is - **21**
The global variable value after a function call is - **21**

Figure 7.22: Output – of program to demonstrate the scope of local and global variables in PHP

7.8 Self-Assessment Questions

Q1. Give an overview of PHP. Explain the modes of operation in PHP. [4 marks, L2]

Q2. Describe the basic syntax of embedding PHP code in an XHTML document with a suitable example. [4 marks, L2]

Q3. Explain how variables are declared and the data types in PHP. [4 marks, L2]

Q4. Explain the Output statements used in PHP with suitable examples for each. [6 marks, L2]

Q5. What are the differences between echo and print statements? [3 marks, L2]

Q6. Explain in brief control statements in PHP with suitable examples. [8 marks, L2]

Q7. Describe various loops used in PHP with examples.[8 marks, L2]

Q8. How are string literals defined in PHP? Explain the 2 ways of concatenating strings in PHP with suitable examples. [8 marks, L2]

Q9. Explain any 10 string methods in PHP with suitable examples for each. [10 marks, L2]

Q10. Explain the different ways of creating arrays in PHP with suitable examples. Describe how elements are inserted, deleted, modified, and accessed in PHP with suitable examples for each. [8 marks, L2]

Q11. How are arrays created in PHP? Explain at least 10 methods used with arrays in PHP with suitable examples. [10 marks, L2]

Q12. Explain with a diagram the logical internal structure of arrays in PHP. How are arrays in PHP accessed sequentially? Explain various methods used for sequential access of arrays in PHP with a suitable example. [10 marks, L2]

Q13. Explain the 2 forms of foreach() loop with examples for iterating through arrays in PHP Sequentially. Give examples for each. [10 marks, L2]

Q14. Explain the use of functions in PHP. Explain with examples defining and using functions with and without parameters. [6 marks, L2]

Q15. With suitable examples explain passing parameters to functions (pass-by-value and pass-by-reference). [8 marks, L2]

7.9 Self-Assessment Activities

A1. Explain the string methods in PHP with suitable examples of your own.

A2. Explain array methods in PHP with suitable examples of your own.

A3. Explain passing parameters to functions (pass-by-value and pass-by-reference) with Suitable examples of your own.

A4. Install xampp and execute the PHP scripts using strings, arrays, functions, etc.

7.10 Multiple-Choice Questions

1. PHP stands for _____. [1 mark, L1]
   a) Private Home Page
   b) Preprocessed Hypertext Processor
   c) Hypertext Preprocessor
   d) Private Hypertext Processor

2. _____ of the following method is used to output text to the browser in PHP.[1 mark, L1]
   a) echo()
   b) cout()
   c) document.write()
   d) display()

3. In PHP, variables are denoted with the symbol _____.[1 mark, L1]
   a) @
   b) $
   c) %
   d) &

4. What will be the output of the following code? [1 mark, L1]

```
<?php
        $n = 9;
         if ($n % 2 == 0) {
            echo "Even";
           } else {
            echo "Odd";
           }
?>
```

   a) 9
   b) Even
   c) Odd
   d) Error: Undefined variable

5. Two strings can be concatenated in PHP using ____.[1 mark, L1]
   a) join()
   b) combine()

c) merge()
d) the dot (.) operator

6. The ____ function is used to convert a string to lowercase in PHP. [1 mark, L1]
   a) strtolower()
   b) lc()
   c) tolower()
   d) str_lower()

7. The value of an element with the key "Tue" in an associative array in PHP can be accessed using _____.[1 mark, L1]
   $tempday = array("Mon"=>35, "Tue"=>32, "Wed"=>28, "Thu"=>27, "Fri" =>26);
   a) $tempday->Tue;
   b) $tempday['Tue'];
   c) $tempday(Tue);
   d) $tempday->get('Tue');

8. In PHP, to check if a specific key exists in an associative array the method used is ____. [1 mark, L1]
   a) in_array()
   b) exists_key()
   c) is_key_exists()
   d) array_key_exists()

9. The "current" pointer at the time of array creation is initialized by default to reference the ____ element of the array.[1 mark, L1]
   a) First
   b) Last
   c) Middle
   d) None of the above

10. The purpose of the global keyword in PHP functions is _____. [1 mark, L1]
    a) It declares a variable as a constant.
    b) It indicates that a variable is accessible from anywhere in the script.
    c) It specifies the data type of a variable.
    d) It restricts the scope of a variable to the function.

7.11 Key Answers to Multiple-Choice Questions

1. PHP stands for Hypertext Preprocessor.[c]
2. echo() of the following method is used to output text to the browser in PHP.[a]
3. In PHP, variables are denoted with the symbol $. [b]
4. The output of the following code in PHP is Odd [c]

```
<?php
        $n = 9;
         if ($n % 2 == 0) {
            echo "Even";
           } else {
             echo "Odd";
           }
?>
```

5. Two strings can be concatenated in PHP using the dot(.) operator.[d]
6. The strtolower() function is used to convert a string to lowercase in PHP.[a]
7. The value of an element with the key "Tue" in an associative array in PHP can be accessed using $tempday['Tue'].[b]
8. In PHP, to check if a specific key exists in an associative array the method used is array_key_exists(). [d]
9. The "current" pointer at the time of array creation is initialized by default to reference the first element of the array.[a]
10. The purpose of the global keyword in PHP functions is, it indicates that a variable is accessible from anywhere in the script.[b]

7.12 Summary

PHP, which stands for "Hypertext Preprocessor," is an open-source and widely used server-side scripting language designed for web development. It is particularly well-suited for creating dynamic and interactive web applications. PHP scripts are executed on the server, generating HTML code that is then sent to the client's web browser for rendering. PHP supports a wide range of databases and is widely used in developing web applications using databases. PHP also supports the use of cookies and session variables in web applications.

PHP processor has two modes of operation: Copy mode and Interpret mode

PHP code is embedded within XHTML, and it is enclosed in special tags: <?php and ?>. A PHP script starts with <?php and ends with ?>. A PHP script can be placed anywhere in

the document. PHP statements end with a semicolon (;).The default file extension for PHP files is ".php".

In PHP, the output statements are used to display the content on the browser. The output statements used in PHP are – echo, print, and printf().

PHP provides control structures and loops that allow one to manage the flow of the code and execute specific tasks repeatedly.

In PHP, Strings are a sequence of characters. The characters are stored in the string starting with index 0. They can be stored and manipulated using various methods.

Arrays in PHP, are different from those of any other programming languages like 'C' and C++. They are a combination of typical indexed arrays like other languages and associative arrays, or hashes found in some other languages like Perl, Python, Ruby, etc.

Arrays in PHP consist of two parts, a key and a value. i.e. a Key-Value pair.

Arrays in PHP can have some elements with integer keys and some with string keys.

Functions in PHP are similar to functions in JavaScript. Functions in PHP are blocks of code that can be defined to perform specific tasks. Functions enable code reuse and organization. Defining functions reduces the time required to write many lines of code to perform similar tasks each time. It makes the programs compact due to less coding. Functions enhance code organization, readability, and reusability.

7.13 Keywords

- PHP (Hypertext PreProcessor)
- Copy mode
- Interpret mode
- Dot(.) operator
- Assignment(.=) operator
- Associative Arrays
- Pass-by-value
- Pass-by-reference

7.14 Recommended resources for further reading

a. Essential Reading

1. Sebesta, R. W. (2010). Programming the World Wide Web (6th ed.), Pearson education.
2. Subramanian, V. (2019). Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node (2nd Ed.), Apress.

b. Recommended Reading

1. DT Editorial Services. (2016). HTML 5: Covers CSS3, JavaScript, XML, XHTML, AJAX, PHP & jQuery: Black Book, Dreamtech Press.
2. Koroliova, E. W. I., (2018). MERN Quick Start Guide: Build Web applications with MongoDB, Express.js, React and Node, Packt.

--*--