# UNIT 3
# INTRODUCTION TO PHP

PHP stands for 'PHP: Hypertext Preprocessor'. PHP is an open source general-purpose scripting language that is suited for web development and can be embedded into HTML code. It is a server side scripting tool and its code is similar to Java, C and Perl. The main objective of PHP is to develop dynamic web pages at ease. PHP was originally created by *Rasmus Lerdorf* in 1994 but it is now developed by The PHP Group.

PHP is a powerful tool for making dynamic and interactive Web pages. PHP stands for Hypertext Pre-processor. PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

PHP code is inserted inside HTML code and when the user requests for a PHP web page, it is interpreted and executed on the web server. To process the PHP code on the web server, a PHP interpreter has to be installed on the web server. After execution of the PHP code in the web server, an HTML page is created, which is sent to the client browser.

One of the strongest and most significant features in PHP is its support for database programming. The most common database used with PHP is MySQL.

PHP interpreter is available for all operating systems. Linux platforms commonly use LAMP (Linux, Apache, MySQL and PHP) server software which is freely downloadable. LAMP uses Linux as the server operating system, Apache as the web server, MySQL as the database and PHP for server side scripting. Windows operating systems use WAMP server software which is alsoavailable for free download.

## Server side scripting

Server-side scripting is a method of designing websites so that the process or user request is run on the originating server. Server-side scripts provide an interface to the user and are used to limit access to proprietary data and help keep control of the script source code. Below is an example of client-side scripts vs. server-side scripts.

The **server-side environment** that runs a scripting language is a web server. A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. This HTML is then sent to the client browser. It is usually used to provide interactive web sites that interface to databases or other data stores on the server.

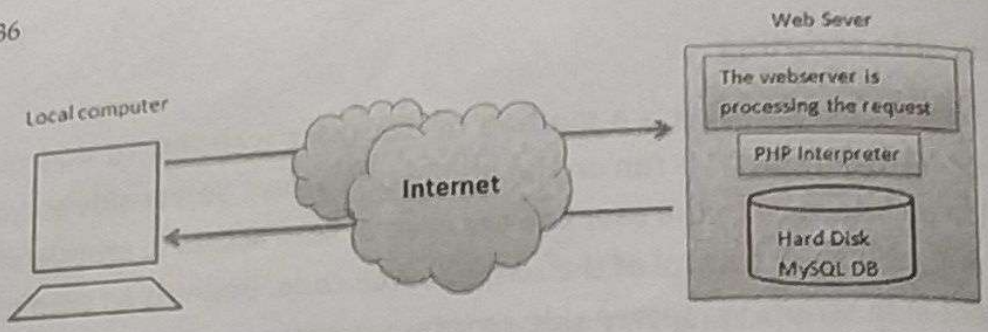
Server-side scripting as it relates to web pages usually refers to PHP code that is executed on the web server before the data is passed to the user's browser. In the case of PHP, all PHP code is executed server-side and no PHP code ever reaches the user. After the PHP code is executed, the information it outputs is embedded in the HTML, which is sent to the viewer's web browser.

The result of the PHP code is there because it is embedded in the HTML on the server before the web page is delivered to the browser.

This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

A script is a program or sequence of instructions that is interpreted or carried out by another program rather than by computer processor.

Server side scripting is used to connect to the database that reside on the web server. Server side scripting can access the file system residing at the web server. Response from a server-side is slower as compared to a client-side script because the scripts are processed on the remote computer.




## Role of web server software

Hosting websites refers to placing websites on web servers to bring them into access by people over internet.web servers play a significant role in web hosting services as they form the key elements.

The term web server is often used for referring to the server computer that hosts websites. It is also used to refer to a web server software that is installed in a server computer in order to make it a web server. It is also used to refer to a web server software that is installed in a server computer in order to make it a web server.

A web server enables us to deliver web pages or services like e-mail, blog, etc. to users on the Internet. A web server consists of a server computer that runs a server operating system and a web server software installed on it for providing services like www, e-mail, etc. over the Internet.

Following are a few functions performed by web servers in hosting.

- Stores and secures website: in web hosting services, web server stores all websites data and secures it from unauthorized users when it is properly configured.
- Provides web database access: a webservers responsibility is to provide access to websites that are hosted. Web hosting service providers own some web servers that are used in variable ways to provide different web hosting services such as backend database servers.
- Serve the end user requests: web servers accept requests from different users connected over the internet and serve them accordingly.

**Web Server Program**

When accessing a website, the browser (Chrome, Firefox ...) communicates with the server, requesting and receiving the data of the page in question. As mentioned, the physical server has specific programs to respond to the type of request made. In the case of a website, this request is made through a protocol known as HTTP (hypertext transfer protocol). All websites on the internet travel using this protocol (or its secure version known as HTTPS). Thus, there are specific programs to respond to HTTP requests. This type of program is also known as a web server.

For other requests, different types of web server are used, such as for sending and receiving emails. A web server that sends and receives emails can be installed on the same computer (server) used for HTTP requests.

**Here are some examples of web servers (software) commonly found on web servers (hardware):**

- **HTTP Server:** Sends the files that make up a site.
- **FTP server:** Upload and download files between computers and servers.
- **Email server:** Sends, receives, and stores e-mails.
- **Database server:** Stores data in a specific structure.

So, when we request a page on the internet, the request will be sent to the server that contains the files of the site in question. When the request arrives at the server, the software contained therein will be responsible for processing the requested information and responding accordingly.

A web server can receive requests and send files to thousands of users simultaneously or in a short time.

**Steps for creating webpages using PhP with LAMP**

1. Start the web server

   Use the following command to start Lampp websever.

   - \opt\lampp\lampp start.

2. Save your webpages.
   Create your own folder; that contains your webpages in the following path.
   - Click places-> computer-> opt folder-> lampp-> htdocs->
   - Create your folder in this *htdocs* folder.
   - Then save your Php programmes or webpages in this folder.
3. Viewing webpages in Lampp webserver.
   - Open a web browser.
   - Use the url like: http://localhost//*yourfolder/filename.php*

     Eg: http://localhost/bvmcollege/index.php

     http://localhost/bvmcollege/courses.html

**Let's continue with php**

PHP documents end with the extension *.php*. When a web server encounters this extension in a requested file, it automatically passes it to the PHP processor. A PHP document will output only HTML. To prove this, you can take any normal HTML document such as an *index.html* file and save it as *index.php*, and it will display identically to the original.

For inserting php commands use a new tag<?php —?>. Inside this tag you can insert your php commands. A simple 'hello world' php program shows below.

```
<?php
        echo "hello world";
?>
```

**php comments**

A comment in PHP code is a line that is not read as part of the program. Its only purpose is to be read by someone who is editing the code. There are two types of comments. Thay are:

1. **Single line comment**

For single line comment use // or # at the beginning of the line. For example

```
<?php
// This is a single-line comment
echo"welcome";
# This is also a single-line comment
?>
```

2. **Multi-line comment**

The multiple line PHP comment begins with " /* " and ends with " */ ". For example.

```
<?php
/*
This is a multiple-lines comment
that spans over multiple
lines
*/
?>
```

**echo and print**

echo and print statements to display the output in a web browser.

**Echo statement**

The echo statement can display anything that can be displayed to the browser, such as string, numbers, variables values, the results of expressions etc.

Example 1:

```
<?php
// Displaying string of text
echo "Hello World!";
?>
```

*Example 2:*

```
<?php
// Displaying HTML code
echo "<h4>This is a simple heading.</h4>";
?>
```

Example 3:

```
<?php
  $a = 2;
  $b = 2.5;
  $c= $a + $b;
  print("$a + $b = $c ");
?>
```

**Print statement**

You can also use the print statement to display output to the browser. It an alternative to echo statement .Like echo the print is also a language construct not a real function. So you can also use it without parentheses like: print or print().

Both echo and print statement works exactly the same way except that the print statement can only output one string, and always returns 1. That's why the echo statement considered marginally faster than the print statement since it doesn't return any value. For example

```
<?php
        $txt = "Hello World!";
        $num = 123456789;
        print $txt;
        print "<br>";
        print $num;
?>
```

**Difference between echo and print statement**

| Echo | Print |
|---|---|
| In PHP, echo is not a function but a language construct. | In PHP, print is not a really function but a language construct. However, it behaves like a function in that it returns a value. |
| It can accept multiple expressions. | It cannot accept multiple expressions. |
| It is faster than print as it does not return any value. | It is slower than echo as it returns a value. |

| | |
|---|---|
| It is a statement used to display the output and can be used with the parentheses echo or without the parentheses echo. | It is also a statement which is used to display the output and used with the parentheses print() or without the parentheses print. |
| It can pass multiple string separated as (,). | It cannot pass multiple arguments. |
| It doesn't return any value. | It always returns the value 1. |
| Echo is faster than print | It is slower than echo |

**Variables**

Variables are named locations to store data. In php all variable name must starts with dollar($) symbol. For example.

```
$var;
<?php
$a=10;
$b=5;
$c=$a+$b
echo "sum of "+$a+"and"+$b+"is"+$c
?>
```

Rules for PHP variables:

- Variable starts with the $ sign, followed by the name of the variable.
- Variable name cannot start with a number.
- Variable name must start with a letter or the underscore character.
- Variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _ ).
- Variable names are case-sensitive that is $x and $X are different.

**Data Types**

The values assigned to a PHP variable may be of different data types including simple string and numeric types to more complex data types like arrays and objects.

PHP supports total eight primitive data types: Integer, Floating point number or Float, String, Booleans, Array, Object, resource and NULL. These data types are used to construct variables.

| Datatype | Description | Example |
|---|---|---|
| String | A string is a sequence of characters, like "Hello world!". A string can be any text inside quotes. | <?php<br>$var="welcome";<br>echo $var;<br>?> |
| Integer | Non-decimal numbers as considered as integer datat type. It includes either negative or positive numbers. | <?php<br>$a=55;<br>echo "a="+$a;<br>?> |
| Float (double) | It is a number with decimal part. | <?php<br>$x = 10.5;<br>echo "x="+$x;<br>?> |
| Boolean | A Boolean represents two possible states( TRUE or FALSE). Booleans are often used in conditional testing. | $x = true; |
| Array | An array stores multiple values in a single variable. | <?php<br>$rno=(1,2,3,4,5);<br>foreach( $rno as $value ) {<br>echo " $value <br />";<br>}<br>?> |
| Object | Objects are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class. | |
| NULL | The special NULL value is used to represent empty variables in PHP. When a variable is created without a, it is automaticallyassigned a value of null. | <?php<br>$a = NULL;<br>echo $a;<br>?> |
| Resource | A resource is a special variable, holding a reference to an external resource.Resource variables typically hold special handlers to opened files and database connections. | <?php<br>// Open a file for reading<br>$handle =<br>fopen("note.txt", "r");<br>var_dump($handle);<br>?> |

**PHP operators**

Operators are symbols that are used to perform operations on operands(variables) and values. Different types of operators in php are:

- Arithmetic Operators
- Comparison Operators
- Logical(or Relational)Operators
- Assignment Operators
- Conditional(or ternary)

**Arithmetic Operators**

C Arithmetic operators are used to perform mathematical calculations like addition, subtraction, multiplication, division and modulus.

| Arithmetic Operators | Operation | Example |
|---|---|---|
| + | Addition | $a+$b |
| - | Subtraction | $a-$b |
| * | multiplication | $a*$b |
| / | Division | $a/$b |
| % | Modulus | $a%$b |
| ++ | Increment operator, increases integer value by one | $a++ |
| — | Decrement operator, decreases integer value by one | $a— |

*Example: program to perform arithmetic operations*

```
<html>
<body>
<?php
$a = 10;
$b = 5;
echo $a + $b;
```

```
echo $a - $b;
echo $a * $b;
echo $a / $b;
echo $a % $b;
 echo $a++;
 echo $a—;
?>
</body>
</html>
```

## Comparison Operators

Comparisonoperators are used to find the relation between two variables. i.e. to compare the values of two variables in the program.

| Operators | Example | Description |
|---|---|---|
| > | $x >$y | $x is greater than $y |
| < | $x <$y | $x is less than $y |
| >= | $x >= $y | $x is greater than or equal to $y |
| <= | $x <= $y | $x is less than or equal to $y |
| == | $x == $y | $x is equal to $y |
| != | $x != $y | $x is not equal to $y |

*Example*

```
<html>
<head>
<title>comparison operators</title>
</head>
<body>
<?php
    $a=10;
    if($a>0)
    print"$a is positive";
    else
    print"$a is negative";
    ?>
</body>
</html>
```

## Logical Operators

Assume variable A holds 10 and variable B holds 20 then:

| Operator | Description | Example |
|---|---|---|
| and | Called Logical AND operator. If both the operands are true then condition becomes true. But | (A and B) is true. |
| Or | Called Logical OR Operator. If any of the two operands are non zero then condition becomes true. | (A or B) is true. |
| && | Called Logical AND operator. If both the operands are non zero then condition becomes true. | (A && B) is true. |
| \|\| | Called Logical OR Operator. If any of the two operands are non zero then condition becomes true. | (A \|\| B) is true. |
| ! | Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false. | !(A && B) is false. |

The operators ***and*** and ***or*** have a lower precedence, so in some cases, you may need extra parentheses to force the required precedence.

## Assignment Operators

These operators are used to assign values to variables.

| Operator | Example | Equivalent to |
|---|---|---|
| = | $x = 5 | $x = 5 |
| += | $x+= 5 | $x = $x+ 5 |
| -= | $x -= 5 | $x = $x - 5 |
| *= | $x *= 5 | $x = $x * 5 |
| /= | $x /= 5 | $x = $x / 5 |
| .= | $x .= $y | $x = $x . $y |
| %= | $x %= 5 | $x = $x % 5 |

**Note:** String concatenation uses the period (.) operator to append one string of characters to another. The simplest way to do this is as follows:

```
echo "You have " . $msgs . " messages.";
```

Assuming that the variable $msgs is set to the value 5, the output from this line of code will be:

```
You have 5 messages.
```

**Conditional (or ternary)**

There is one more operator called conditional operator. This first evaluates an expression for atrue or false value and then execute one of the two given statements depending upon the resultof the evaluation. The conditional operator has this syntax:

| Operator | Description | Example |
|---|---|---|
| ? : | Conditional Expression | If Condition is true ? Then value X: Otherwise value Y |

**Branching statements**

The special types of PHP statements used for making decisions are called decision making statements or branching statements. Branching statements are the set of commands used to perform different actions based on different conditions. The following are the different types of branching statements.

1. If statement
2. If –else statement
3. Else if statement
4. Switch statement

**1. If statement**

If statement allows you control the execution of statements. If statement can only executes one or more php statement(s) only when the conditional expression is evaluated as true. The general syntax of if statement is:

```
if (conditional expression)
{
statement(s);
}
```

*For example:*

```
if ($age >= 18)
{
echo "You are eligible for voting";
}
```

**2. If –else statement**

If you want to execute some code if a condition is true and another code if a condition is false, use the if....else statement. General syntax is:

```
if (conditional expression)
{
        statement(s);
}
else if (conditional expression)
{
        statement(s);
}
For example
$d=date("D");
if ($d=="Fri")
        echo "Have a nice weekend!";
else
    echo "Have a nice day!";
```

**3. Else if statement**

If you want to execute some code if one of several conditions are true use the elseif statement. General syntax is:

```
if (condition_1)
{
  statement_1
}
[elseif (condition_2)
{
```

```
    statement_2
  }]
  ...
  [elseif (condition_n_1)
  {
    statement_n_1
  }]
  [else
  {
    statement_n
  }]
For example
$d=date("D");
if ($d=="Fri")
        echo "Have a nice weekend!";
elseif ($d=="Sun")
        echo "Have a nice Sunday!";
else
    echo "Have a nice day!";
```

**4. Switch statement**

Switch statements work the same as if statements. However the difference is that they can check for multiple values. A switch statement allows a program to evaluate an expression and attempt to match the expression's value to a case label. If a match is found, the program executes the associated statement. The syntax for the switch statement as follows:

```
switch (expression)
{
 case label_1:
        statements_1
        [break;]
 case label_2:
        statements_2
        [break;]
```

```
...
default:
  statements_n
  [break;]
}
```

*For example*

```
$n=5;
switch ($n)
{
case $n:
        echo "This is positive";
        break;
case $n:
        echo "This is negative";
        break;
case $n:
        echo "This is zero";
        break;
default:
        echo "enter a number !!!";
}
```

**Loops**

Looping statements are used to repeat the same block of code a given number of times, or until a certain condition is met. Php has four types of loops. In addition, you can use the Break and Continue statements within looping statements.

- The While Loop
- The Do...While Loop
- The For Loop
- The Foreach Loop
- Break and Continue Statements

**While Loop**

It tells PHP to execute the nested statements repeatedly, as long as the while expression evaluates to TRUE. If the condition becomes false, the statements within the loop stop executing and control passes to the statement following the loop. Note that If the first evaluation of the statement return FALSE, the while loop will not be executed at all. The While loop syntax is as follows:

```
while (condition)
{
 code to be executed;
}
```

*for example:*

```
$i = 1;                          //initialization
While($i <= 5)     //test condition
{
   echo $i."</br>";          //body of the loop
        $i++;                      //increment statement
}
```

**Output**

**1**
**2**
**3**
**4**
**5**

## Do-while loop

Do-while loop first executes a piece of code once and then repeat the loop until the condition is satisfied. It executes statement once even if the condition is false.

```
Syntax
do
{
code to be executed;
}
while (condition);
```

*Example:*

```
<?php
$a = 3;
$b = 7;
do
{
echo "$a is less than $b <br>";
$a++;
}
while ($a < $b);
?>
```

**Output**

3 is less than 7
4 is less than 7
5 is less than 7
6 is less than 7

### Difference Between while and do...while Loop

The while loop differs from the do-while loop in one important way – with a while loop, the condition to be evaluated is tested at the beginning of each loop iteration, so if the conditional expression evaluates to false, the loop will never be executed.

With a do-while loop, on the other hand, the loop will always be executed once, even if the conditional expression is false, because the condition is evaluated at the end of the loop iteration rather than the beginning.

### For loop

It executes a piece of code a specific number of times. PHP for loops execute a block of code for a specified number of times. This loop is preferred when you are sure about the number of iterations of the script.

Syntax

```
for (initial counter; test counter; increment counter)
{
piece of code to be executed;
}
```

*for loop has three parameters*

- Initial counter is used to initialize the loop counter value.
- Test counter is used to evaluate each and every loop iteration. If it's true, the loop will execute otherwise it will stop.
- Increment counter will be used to increment or decrement the loop counter value (e.g: $a++,$b–).

*Example*

```
<?php
for ($a = 0; $a <= 5; $a++) {
echo "The number is: $a <br>";// will echo numbers till 5 from 0
}
?>
```

**Output**

The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5

## Foreach loop

The foreach construct provides an easy way to iterate over arrays. foreach works only on arrays and objects, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variable. There are two syntaxes:

```
foreach (array_variable as $array_value){
  statement
}
```

and

```
foreach (array_ variable as $key => $array_value){
  statement
}
```

The first form loops over the array given by array_ variable. On each iteration, the value of the current element is assigned to $array_value and the internal array pointer is advanced by one (so on the next iteration, you'll be looking at the next element). When foreach first starts executing, the internal array pointer is automatically reset to the first element of the array.

The second form will additionally assign the current element's key to the $key variable on each iteration.

HERE,

"$array_variable" is the array variable to be looped through

"$array_value " is the temporary variable that holds the current array item values.

**How it works**

In every iteration, value of current array element will be assigned to $array_value and the array pointer is moved by one. The process repeats until the last array element is reached.

*Example:*

```
<?php
$cars = array("Skoda", "Nissan", "Toyota", "Suzuki");
foreach ($cars as $brand)
{
echo "$brand <br>";
}
?>
```

**Output**

Skoda
Nissan
Toyota
Suzuki

In the above example, four different brand names are stored in an array variable $cars. For each loop matches every key and then displays output.

**Difference between for and foreach**

| | For loop | Foreach loop |
|---|---|---|
| 1 | When we use for loop we need a condition after satisfying which the loop will continue. For this reason we use some counters in our loop which obviously occupies some memory. | But while using foreach loop we don't have to think about the counter. In case of foreach loop there is no extra memory required for the counters also. Because we don't have any counter in foreach loop. |
| 2 | for loop is used if we know already that how many times the script should be run | In the case of foreach loop, we don't have any idea about the number of iteration. |
| 3 | for statement can be used in any program. That is for works with a normal variable. | foreach loop is used to iterate only arrays and objects |

**Arrays**

Array is a collection a collection of values of same type under a common name. If we want to store the names 50 students; then we have to use 50 variables. It is very complex to maintain 50 variables. These problems of handling bulk amount of variables by using arrays.

There are three types of arrays in PHP are:

1. Numeric arrays or indexed arrays.
2. Associative arrays
3. Multidimensional arrays

Some arrays are referenced by numeric indices; others allow alphanumeric identifiers.

**Numeric arrays**

Numeric array uses numbers as access keys (index). Each and every element in the array is identified by this access key or index. By default the value of the index is start with zero. This access key is used to retrieve or modify its value Below is the syntax for creating numeric array in php.

```
<?php
$variable_name[n] = value;
?>

Or

<?php
$variable_name = array(n => value, ...);
?>
```

Here; $variable_nam is the array name, "[n]" is the access index number of the element and "value" is the value assigned to the array element.

The following examples show two ways of creating an indexed array:

**Method 1:**

```
<?php
$name=array("alex","rani","balu");
  print_r($name);
?>
Or
<?php
$name[0]="alex";
$name[0]="rani";
$name[0]="balu";
print_r($name);
?>
```

**Method 2:**

```
<?php
$name = array(0 => "alex",
        1 => "rani",
        2 => "balu",
        3 => "karthika",
        4 =>"john" );
echo $name[4];
?>
```

**Output**

john

**Associative arrays**

Associative array differ from numeric array in the sense that associative arrays use descriptive names for id keys. In an associative array, the keys assigned to values can be arbitrary and user defined strings. Below is the syntax for creating associative array in php.

```
<?php
        $variable_name['key_name'] = value;
        $variable_name = array('keyname' => value);
?>
```

Here, "$variable_name..." is the array name, "['key_name']" is the access index of the element and "value" is the value assigned to the array.

Sometimes it's better to use the index name instead of index number. For example if you want to save three students' names and numbers so your best option will be to use each student's name as index value for the array and his numbers as the values.

```
<?php
        $name['alex'] = 10;
        $name['balu'] = 19;
        $name['rani'] = 31;
        print_r($name);
?>
```

Associative array are also very useful when retrieving data from the database. The field names are used as id keys.

**Multidimensional arrays**

We can define multi-dimensional arrays as array of arrays. As the name suggests, every element in this array can be an array and they can also hold other sub-arrays within. Values in the multi-dimensional array are accessed using multiple index. The advantage of multidimensional arrays is that they allow us to group related data together. For example

```
<?php
    $marks = array(
     "mohammad" => array (
       "physics" => 35,
       "maths" => 30,
       "chemistry" => 39
     ),

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

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

    /* Accessing multi-dimensional array values */
    echo "Marks for mohammad <br" ;
    echo "> physics : ".$marks['mohammad']['physics']."<br>
maths :".$marks['mohammad']['maths']."<br> chemistry :
".$marks['mohammad']['chemistry']. "<br />";
?>
```

Example 2:

```
<?php
      $employee = array
(
array("smith","manager",52000),
array("ram","clerk",23000),
array("alice","clerk",19500),
);
```

```
echo "employee name : ".$employee[0][0]." Designation :
".$employee[0][1]." salary : ".$employee[0][2]."<br>";

echo "employee name : ".$employee[1][0]." Designation :
".$employee[1][1]." salary : ".$employee[1][2]."<br>";

echo "employee name : ".$employee[2][0]." Designation :
".$employee[2][1]." salary : ".$employee[2][2]."<br>";

 ?>
```

*Example 3*

```
<?php
        $cars = array(
                        array("Honda Accord", "V6", 830000),
                        array("Toyota Camry", "LE", 924000),
                        array("Nissan Altima", "V1", 890000),
                );

for($i=0;$i<count($cars);$i++)
{
   $c=0;
   foreach($cars[$i] as $key=>$value)
        {
           $c++;
           echo $key."=".$value;
           if($c<count($cars[$i])) echo ",";
   }
   echo "<br>";
}
 ?>
```

## Viewing Array Structure and Values

You can see the structure and values of any array by using one of two statements — var_dump() or print_r(). The print_r() statement, however, gives somewhat less information. Consider the following example:

```
<?php
// Define array
$cities = array("London", "Paris", "New York");

// Display the cities array
print_r($cities);
?>
```

**Output**

Array ( [0] => London [1] => Paris [2] => New York )

This output shows the key and the value for each element in the array.

Following example shows the use of var_dump().

```
<?php
// Define array
$cities = array("London", "Paris", "NewYork");

// Display the cities array
var_dump($cities);
?>
```

**Output**

array(3) { [0]=> string(6) "London" [1]=> string(5) "Paris" [2]=> string(8) "NewYork" }

This output shows the data type of each element, such as a string of 6 characters, in addition to the key and value.

## Important array functions

PHP provides various array functions to access and manipulate the elements of array. The important PHP array functions are given below.

### 1. PHP array() function

PHP array() function creates and returns an array. It allows you to create indexed, associative and multidimensional arrays.

*Example*

```
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:

Season are: summer, winter, spring and autumn

**2. count function**

The count function is used to count the number of elements that an php array contains. The code below shows the implementation.

```
<?php
$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");
echo count($lecturers);
?>
```

Output:

3

**3. is_array function**

The is_array function is used to determine if a variable is an array or not. Let's now look at an example that implements the is_array functions.

```
<?php
$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");
echo is_array($lecturers);
?>
```

Output:

1

**4. sort()**

This function is used to sort arrays by the values. If the values are alphanumeric, it sorts them in alphabetical order. If the values are numeric, it sorts them in ascending order. It removes the existing access keys and add new numeric keys.

The output of this function is a numeric array

```
<?php
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam"
=> "Female");
sort($persons);
print_r($persons);
?>
```

Output:

Array ( [0] => Female [1] => Female [2] => Male )

5. **ksort()**

This function is used to sort the array using the key. The following example illustrates its usage.

```
<?php
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam"
=> "Female");
ksort($persons);
print_r($persons);
?>
```

Output:

Array ( [John] => Male [Mary] => Female [Mirriam] => Female )

6. **asort()**

This function is used to sort the array using the values. The following example illustrates its usage.

```
<?php
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam"
=> "Female");
asort($persons);
print_r($persons);
?>
```

Output:

Array ( [Mary] => Female [Mirriam] => Female [John] => Male )

7. **array_search()**

PHP array_search() function searches the specified value in an array. It returns key if search is successful.

```
<?php
$season=array("summer","winter","spring","autumn");
$key=array_search("spring",$season);
echo $key;
?>
```

Output:

2

8. **array_reverse()**

PHP array_reverse() function returns an array containing elements in reversed order.

Example

```
<?php
$season=array("summer","winter","spring","autumn");
$reverseseason=array_reverse($season);
foreach( $reverseseason as $s )
{
 echo "$s<br />";
}
?>
```

Output:

autumn
spring
winter
summer

9. **sizeof()**

Count elements in an array, or properties in an object. Or Returns the number of elements in an array.

```
<?php
  $a[0] = 1;
  $a[1] = 3;
  $a[2] = 5;
  $result = sizeof($a);
  print($result);
?>
```

This will produce the following result "

3

**10. array_unique()**

The array_unique() function removes duplicate values from an array.

```
<?php
  $input = array("a" => "green", "red", "b" => "green", "blue", "red");
  $result = array_unique($input);
  print_r($result);
?>
```

This will produce the following result "

Array ( [a] => green [0] => red [1] => blue )

**11. array_sum()**

It calculates the sum of values in an array and returns the sum of values in an array as an integer or float.

```
<?php
  $ba = array(2, 4, 6, 8);
  echo "sum(ba) = " . array_sum($ba);
  print "\n" ;
  $ab = array("a" => 1.2, "b" => 2.3, "c" => 3.4);
  echo "sum(ab) = " . array_sum($ab);
?>
```

This will produce the following result "

sum(ba) = 20

sum(ab) = 6.9

12. array_reverse()

This function reverse the order of all the elements of a padded array.

```
<?php
  $input = array("a"=>"banana","b"=>"mango","c"=>"orange");
  print_r(array_reverse($input));
?>
```

This will produce following result "

Array ( [c] => orange [b] => mango [a] => banana )

## REVIEW QUESTIONS

### Part A

1. What is LAMP?
2. What is PHP?
3. What is server side scripting?
4. What are different methods of giving comments in PHP?
5. What are variables? How to create variables in PHP?
6. What are echo and print statement?

### Part B

1. What is the role of web server software?
2. Explain foreach statement with example.
3. Explain the different data types in PHP.
4. Explain PHP operators.
5. What is the difference between for and foreach loop?
6. Explain any four array functions in PHP.

### Part C

1. Explain the following.

a. Branching statements in PHP

b. Loops in PHP

2. Explain arrays with suitable examples.