

## MODULE III

### INTRODUCTION TO PHP

PHP stands for Personal Home Page. Another stands is PHP Hypertext Pre-processor. PHP is popular general purpose server side scripting language that is especially suited to web development. In 1994 Rasmus Lerdorf designed PHP and developed by Zend Technologies. PHP files can contain text, HTML, CSS, JavaScript, and PHP code. The file extensions are .php, .phtml, .php3, .php4, .php5, .php7 or .phps.

PHP allows developers to build logic into the creation of webpage content and handle data returned from a web browser. PHP also contains a number of extensions that make it easy to interact with databases extracting data to be information entered by a website visitor back into the database.

PHP consists of Scripting Language and Interpreter. Scripts are embedded in to the HTML documents that are served by the web server. Interpreter takes the form of module that integrates in to the web server and then converting the script in to the commands. The computer then executes to achieve the results, defined on the scripts by the web developer.

In PHP, all keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are NOT case-sensitive. That is echo, ECHO and Echo are equal. But all variable names are case-sensitive. That is the variables \$color, \$COLOR, and \$coLOR are treated as three different variables.

#### Features

- PHP scripts are executed on the server
- PHP can generate dynamic page content
- PHP is free to download and use
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in our database
- PHP can be used to control user-access
- PHP can encrypt data
- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is easy to learn and runs efficiently on the server side

#### Basic PHP Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser. A PHP script can be placed anywhere in the document. A PHP script starts with `<?php` and ends with `?>`: A PHP file normally contains HTML tags, and some PHP scripting code. Each code line in PHP must end with a semicolon. The default file extension for PHP files is ".php".

```
<?php
// PHP code goes here
?>
```

### **Example:**

```
<html> <head>
<title> My First PHP Page
</title> </head>
<body bgcolor="red">
```

```
<?php
echo "Welcome To PHP";
?>
```

```
<?php
phpinfo();
?>
```

```
<?php
$a=10;
$b=5;
$c=$a+$b;
echo $c,"<br>";
?>
```

```
<?php
echo "<b>",date("d/m/y"),"<b>";
?>
```

```
</body></html>
```

## **SERVER SIDE SCRIPTING**

Server side web scripting is mostly about connecting websites to backend servers, processing data and controlling the behavior of higher layers such as html. These enable two way communication- Clients to Server and Server to Client. Server-side scripts are run on the server. This reduces the amount of bugs or compatibility issues since the code is run on one server using one language and hosting software. Server-side programming can also be encrypted when users send form variables, protecting users against any hack attempts. Some examples of server-side programming languages are C#, VB.NET, and PHP.

### **Advantages of Server side Scripting**

1. Ensure high level security to the source code
2. It does not require the user to download plug-in like Java or Flash (client-side scripting).
3. We can create a single website template for the entire website. Each new dynamic page we create will automatically use it.

4. We can configure a site to use a content management system, which simplifies the editing, publishing, adding of images, and creation of web applications. Many apps are often available in the form of extensions.
5. Load times are generally faster than client-side scripting.
6. Our scripts are hidden from view. Users only see the HTML output, even when they view the source.
7. The site can use a content management system which makes editing simpler.
8. Generally quicker to load than client-side scripting
9. User is able to include external files to save coding.

## **WEB SERVER SOFTWARE**

A Web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computers' HTTP clients. Dedicated computers and appliances may be referred to as Web servers as well. Leading Web servers include Apache (the most widely-installed Web server), Microsoft's Internet Information Server (IIS) and nginx (pronounced engine X) from NGNIX. Other Web servers include Novell's NetWare server, Google Web Server (GWS) and IBM's family of Domino servers.

When we create a website, it must be published on a server computer that is connected to the internet. There are many different types of web server software. The operating system determines the type of web server software that can be run. The web server is responsible for responding to http request for WebPages depending upon the type of website that we have created. The web server software might also need to make request to application system, programming scripts and databases. When the sever let we share information over the internet or through internet or extranet.

### **Role of Web Servers**

The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP).

### **Popular Web Servers**

1. Apache HTP Server
2. Apache TomCat
3. Microsoft IIS
4. Sun Java Web Server
5. Ngnix Web Server
6. Klone
7. Oracle Web Server
8. Zeus
9. GWS Google
10. IBM Lotus

## Apache Web Server

It is the most popular web server in the world developed by Apache software foundation. Apache web server is an open source software and can be install all operating system include UNIX, LINUX, WINDOWS, MAC OS and more. About 60% of web server's machines are run in the Apache web server.

## Advantages of Apache

- Easy implementation of latest protocol
- It is customized (can be change by user)
- It follows modular architecture (can use or make by modules)
- Remote administration is very convenient
- Efficient
- Optimized
- Less system recourses needed
- Portable
- Stability
- Reliability
- World wide support

## WAPP Server

W --- Windows Component

A --- Apache

P ---PostgreSQL

P ---PHP

## INCLUDING FILES

We can include php file to another php files by *include()* function. For example, if we have a set of defend variables that need to be referenced in every page on our site. We could define once in a single php script. Then each of our pages where we want the variable to appear, we can use and include statement that defined the variables. When our script were passed, the parser inserts the code from the include file into a webpage just as if we would types it there ourselves. The final outputs then send to the browser.

## Example

<pre>Address.php &lt;?php echo "GEMS ASC &lt;br&gt;"; echo "Ramapuram &lt;br&gt;"; echo "Malappuram &lt;br&gt;"; ?&gt;</pre>	<pre>Info.php &lt;?php phpinfo(); ?&gt;</pre>	<pre>Welcome.php &lt;?php echo "Welcome to the world of PHP &lt;br&gt;"; include("address.php"); include("info.php"); ?&gt;</pre>
<b>GEMS ASC Ramapuram Malappuram</b>	<b>Information about PHP</b>	<b>Welcome to the world of PHP GEMS ASC Ramapuram Malappuram</b>

## COMMENTS

A comment in PHP code is a line that is not read/executed as part of the program. Its only purpose is to be read by someone who is looking at the code. Comments can be used to:

- Let others understand what we are doing
- Remind ourselves of what we did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind we of what we were thinking when we wrote the code

PHP supports several ways of commenting:

1. Single line Comment  
This uses the symbols // and #
2. Multiline Comment  
/\*.....\*/

## DATA TYPES

Data Types defines the type of data a variable can store. PHP allows eight different types of data types. The first five are called simple data types and the last three are compound data types:

### 1. String

Hold letters or any alphabets, even numbers are included. These are written within double quotes during declaration. The strings can also be written within single quotes but it will be treated differently while printing variables.

Example	Output
<pre>&lt;?php \$x = "Hello world!"; \$y = 'Hello world!'; echo \$x; echo "&lt;br&gt;"; echo \$y; ?&gt;</pre>	Hello world! Hello world!

### 2. Float

It can hold numbers containing fractional or decimal part including positive and negative numbers. By default, the variables add a minimum number of decimal places.

Example	Output
<pre>&lt;?php \$val1 = 50.85; \$val2 = 654.26; \$sum = \$val1 + \$val2; echo \$sum; ?&gt;</pre>	705.11

### 3. Integers

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

#### Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in three formats: decimal (10-based), hexadecimal (16-based - prefixed with 0x) or octal (8-based - prefixed with 0)

Example	Output
<pre>&lt;?php // decimal base integers \$dec1 = 50; \$dec2 = 654; // octal base integers \$oct1 = 07; // hexadecimal base integers \$octal = 0x45; \$sum = \$dec1 + \$dec2; echo \$sum; ?&gt;</pre>	704

### 4. Boolean

Hold only two values, either TRUE or FALSE. Successful events will return true and unsuccessful events return false. NULL type values are also treated as false in Boolean. Apart from NULL, 0 is also considering as false in boolean. If a string is empty then it is also considered as false in boolean data type.

Example	Output
<pre>&lt;?php if(TRUE)     echo "This condition is TRUE"; if(FALSE)     echo "This condition is not TRUE";  // Assign the value TRUE to a variable \$show_error = true; var_dump(\$show_error); ?&gt;</pre>	This condition is TRUE  bool(true)

### 5. NULL

The special NULL value is used to represent empty variables in PHP. A variable of type NULL is a variable without any data. NULL is the only possible value of type null. If a variable is created without a value, it is automatically assigned a value of NULL. Variables can also be emptied by setting the value to NULL:

Example	Output
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;&lt;body&gt; &lt;?php \$x = "Hello world!"; \$x = null; var_dump(\$x); ?&gt; &lt;/body&gt;&lt;/html&gt;</pre>	NULL

## 6. Arrays

Array is a compound data-type which can store multiple values of same data type. An array is a variable that can hold more than one value at a time. It is useful to aggregate a series of related items together, for example a set of country or city names. An array is formally defined as an indexed collection of data values. Each index (also known as the key) of an array is unique and references a corresponding value.

Example	Output
<pre>&lt;?php \$cars = array("Volvo","BMW","Toyota"); var_dump(\$cars);  \$intArray = array( 10, 20 , 30); echo "First Element: \$intArray[0]\n"; echo "Second Element: \$intArray[1]\n"; echo "Third Element: \$intArray[2]\n"; ?&gt;</pre>	<pre>array(3) { [0]=&gt; string(5) "Volvo" [1]=&gt; string(3) "BMW" [2]=&gt; string(6) "Toyota" } First Element: 10 Second Element: 20 Third Element: 30</pre>

## 7. Object

An object is a data type that not only allows storing data but also information on, how to process that data. An object is a specific instance of a class which serve as templates for objects. Objects are created based on this template via the new keyword. Every object has properties and methods corresponding to those of its parent class. Every object instance is completely independent, with its own properties and methods, and can thus be manipulated independently of other objects of the same class. Objects are defined as instances of user defined classes that can hold both values and functions.

Example	Output
<pre>&lt;?php class Car {     function Car() {         \$this-&gt;model = "BMW";     } } // create an object \$vehiclw = new Car(); // show object properties echo \$vehicle-&gt;model; ?&gt;</pre>	BMW

## 8. Resources

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. It is the storing of a reference to functions and resources external to PHP. A common example of using the resource data type is a database call.

## VARIABLES AND SCOPE

The main way to store information in the middle of the PHP program is by using a Variable. It is a named storage location capable of containing data that can be modified during program execution. A variable starts with the \$ sign, followed by the name of the variable. A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). PHP variables are Case-Sensitive.

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

After the execution of the statements above, the variable \$txt will hold the value Hello world!, the variable \$x will hold the value 5, and the variable \$y will hold the value 10.5.

<pre>&lt;?php \$txt = "Balan"; echo "I love \$txt!"; ?&gt;</pre>	<pre>&lt;?php \$txt = " Balan"; echo "I love " . \$txt . "!"; ?&gt;</pre>	<pre>&lt;?php \$x = 5; \$y = 4; echo \$x + \$y; ?&gt;</pre>
<b>I love Balan</b>	<b>I love Balan</b>	<b>9</b>

## Rules for naming a Variable

- 1) A variable starts with the \$ sign, followed by the name of the variable
- 2) A variable name must start with a letter or the underscore character
- 3) A variable name cannot start with a number
- 4) A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_)
- 5) Variable names are case-sensitive (\$age and \$AGE are two different variables)

## Scope of the Variables

In PHP, variables can be declared anywhere in the script. The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function. A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function. Normally, when a function is completed/executed, all of its variables are deleted. However,



sometimes we want a local variable NOT to be deleted. We need it for a further job. To do this, use the static keyword when we first declare the variable:

Example	Output
<pre> &lt;html&gt;&lt;body&gt; &lt;h3&gt; Handling Scope of the Variable&lt;/h3&gt; &lt;?php \$amount=500; echo "The value is : ", \$amount; scope(); echo "&lt;br&gt;The value still is : ", \$amount; function scope() {     \$amount=800;     echo "&lt;br&gt;The value in the function is : ", \$amount; } ?&gt; &lt;/body&gt;&lt;/html&gt; </pre>	<p><b>Handling Scope of the Variable</b></p> <p>The value is : 500  The value in the function is : 800  The value still is : 500</p>

## ECHO AND PRINT

- Both are use to display output data to the screen
- Echo has no return value
- Print has return value of 1
- Print can be used in expression
- Echo can take multiple parameters
- Print can take only one parameter
- Echo is faster than Print
- Echo can be used with or without parenthesis

## OPERATORS

Operators are used to perform operations on variables and values. PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

### Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	Name	Example	Result
+	Addition	\$x + \$y	Sum of \$x and \$y
-	Subtraction	\$x - \$y	Difference of \$x and \$y

*	Multiplication	$\$x * \$y$	Product of $\$x$ and $\$y$
/	Division	$\$x / \$y$	Quotient of $\$x$ and $\$y$
%	Modulus	$\$x \% \$y$	Remainder of $\$x$ divided by $\$y$
**	Exponentiation	$\$x ** \$y$	Result of raising $\$x$ to the $\$y$ 'th power

Example	Output
<pre>&lt;html&gt;&lt;body&gt; &lt;?php echo "x=10 &lt;br&gt;"; echo "y=6&lt;br&gt;"; \$x = 10; \$y = 6; echo "Addition is: ",\$x + \$y; echo "&lt;br&gt; Subtraction is: ",\$x - \$y; echo "&lt;br&gt; Multiplication is: ",\$x * \$y; echo "&lt;br&gt; Division is: ",\$x / \$y; echo "&lt;br&gt; Modulud is: ",\$x%\$y; ?&gt; &lt;/body&gt;&lt;/html&gt;</pre>	<pre>x=10 y=6 Addition is: 16 Subtraction is: 4 Multiplication is: 60 Division is: 1.66666666666667 Modulud is: 4</pre>

### Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable. The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same As...	Description
$x = y$	$x = y$	The left operand gets set to the value of the expression on the right
$x += y$	$x = x + y$	Addition
$x -= y$	$x = x - y$	Subtraction
$x *= y$	$x = x * y$	Multiplication
$x /= y$	$x = x / y$	Division
$x \% = y$	$x = x \% y$	Modulus

Example	Output
<pre>&lt;html&gt; &lt;body&gt; &lt;?php echo "x=20 &lt;br&gt;"; echo "y=100&lt;br&gt;"; \$x = 20; \$y = 15; echo "Assignment is: ",\$x; echo "&lt;br&gt; Addition is: ",\$x += 100; echo "&lt;br&gt; Subtraction is: ",\$x -= 50; echo "&lt;br&gt; Multiplication is: ",\$y *= 30; echo "&lt;br&gt; Division is: ",\$y /= 40; echo "&lt;br&gt; Modulud is: ",\$x \% = 40; ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>x=20 y=15 Assignment is: 20 Addition is: 120 Subtraction is: 70 Multiplication is: 450 Division is: 11.25 Modulud is: 30</pre>

## Comparison Operators

The PHP comparison operators are used to compare two values (number or string): The output of comparison operators are True or False.

Operator	Name	Example	Result
==	Equal	$\$x == \$y$	Returns true if $\$x$ is equal to $\$y$
===	Identical	$\$x === \$y$	Returns true if $\$x$ is equal to $\$y$ , and they are of the same type
!=	Not Equal	$\$x != \$y$	Returns true if $\$x$ is not equal to $\$y$
<>	Not Equal	$\$x <> \$y$	Returns true if $\$x$ is not equal to $\$y$
!==	Not Identical	$\$x !== \$y$	Returns true if $\$x$ is not equal to $\$y$ , or they are not of the same type
<	Less Than	$\$x < \$y$	Returns true if $\$x$ is less than $\$y$
>	Greater Than	$\$x > \$y$	Returns true if $\$x$ is greater than $\$y$
<=	Less Than or Equal To	$\$x <= \$y$	Returns true if $\$x$ is less than or equal to $\$y$
>=	Greater Than or Equal To	$\$x >= \$y$	Returns true if $\$x$ is greater than or equal to $\$y$

## Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value. The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++ $\$x$	Pre-increment	Increments $\$x$ by one, then returns $\$x$
$\$x$ ++	Post-increment	Returns $\$x$ , then increments $\$x$ by one
-- $\$x$	Pre-decrement	Decrements $\$x$ by one, then returns $\$x$
$\$x$ --	Post-decrement	Returns $\$x$ , then decrements $\$x$ by one

Example	Output
<pre>&lt;html&gt; &lt;body&gt; &lt;?php echo "p = 35 &lt;br&gt;"; echo "x = 30 &lt;br&gt;"; echo "y = 25&lt;br&gt;"; echo "z = 10&lt;br&gt;"; \$p = 35; \$x = 30; \$y = 25; \$z = 10; echo "Pre-increment is: ",++\$p; echo "&lt;br&gt; post-increment is: ",\$x++; echo "&lt;br&gt; Pre-decrement is: ",--\$y; echo "&lt;br&gt; Post-decrement is: ",\$z--; ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>p = 35 x = 30 y = 25 z = 10 Pre-increment is: 36 post-increment is: 30 Pre-decrement is: 24 Post-decrement is: 10</pre>

## Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	\$x and \$y	True if both \$x and \$y are true
or	Or	\$x or \$y	True if either \$x or \$y is true
xor	Xor	\$x xor \$y	True if either \$x or \$y is true, but not both
&&	And	\$x && \$y	True if both \$x and \$y are true
	Or	\$x    \$y	True if either \$x or \$y is true
!	Not	!\$x	True if \$x is not true

Example	Output
<pre>&lt;html&gt; &lt;body&gt; &lt;?php \$x = 100; \$y = 50; if (\$x == 100 and \$y == 50) {     echo "Hello world! &lt;br&gt;"; } if (\$x == 100 or \$y == 80) {     echo "Hello world! &lt;br&gt;"; } if (\$x == 100 xor \$y == 80) {     echo "Hello world! &lt;br&gt;"; } if (\$x == 100 &amp;&amp; \$y == 50) {     echo "Hello world! &lt;br&gt;"; } if (\$x == 100    \$y == 80) {     echo "Hello world! &lt;br&gt;"; } if (\$x !== 90) {     echo "Hello world!"; } ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>Hello world! Good Morning Have a Nice Day! See You Are you OK? Then Bye...</pre>

## String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	\$txt1 . \$txt2	Concatenation of \$txt1 and \$txt2
.=	Concatenation assignment	\$txt1 .= \$txt2	Appends \$txt2 to \$txt1

Example	Output
<pre>&lt;html&gt; &lt;body&gt; &lt;?php \$txt1 = "Hello"; \$txt2 = " world!";</pre>	<pre>Hello world! Hello world!</pre>

<pre>echo \$txt1 . \$txt2."&lt;br&gt;"; \$txt1 .= \$txt2; echo \$txt1; ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	
----------------------------------------------------------------------------------------------------------------	--

## Array Operators

The PHP array operators are used to compare arrays.

Operator	Name	Example	Result
+	Union	$\$x + \$y$	Union of $\$x$ and $\$y$
==	Equality	$\$x == \$y$	Returns true if $\$x$ and $\$y$ have the same key/value pairs
===	Identity	$\$x === \$y$	Returns true if $\$x$ and $\$y$ have the same key/value pairs in the same order and of the same types
!=	Inequality	$\$x != \$y$	Returns true if $\$x$ is not equal to $\$y$
<>	Inequality	$\$x <> \$y$	Returns true if $\$x$ is not equal to $\$y$
!==	Non-identity	$\$x !== \$y$	Returns true if $\$x$ is not identical to $\$y$

## CONDITIONAL OPERATORS

Conditional statements are used to perform different actions based on different conditions. When we write code, we want to perform different actions for different conditions. We can use conditional statements in our code to do this. In PHP we have the following conditional statements:

- **if statement** - executes some code if one condition is true
- **if...else statement** - executes some code if a condition is true and another code if that condition is false
- **if...elseif...else statement** - executes different codes for more than two conditions
- **switch statement** - selects one of many blocks of code to be executed

### The if Statement

The if statement executes some code if one condition is true.

#### Syntax

```
if (condition) {
    code to be executed if condition is true;
}
```

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

Example	Output
<pre>&lt;html&gt;&lt;body&gt; &lt;?php echo "y=13&lt;br&gt;"; \$y = 13;</pre>	<pre>y=13 It is a Positive Number</pre>

<pre> if (\$y &gt; 0) {     echo "It is a Positive Number"; } ?&gt; &lt;/body&gt;&lt;/html&gt; </pre>	
-------------------------------------------------------------------------------------------------------	--

### The if...else Statement

The if...else statement executes some code if a condition is true and another code if that condition is false.

#### Syntax

```

if (condition) {
    code to be executed if condition is true;
} else {
    code to be executed if condition is false;
}

```

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

Example	Output
<pre> &lt;html&gt;&lt;body&gt; &lt;?php echo "x=8&lt;br&gt;"; echo "y=21&lt;br&gt;"; \$x = 8; \$y = 21;  if (\$y &gt; \$x) {     echo "&lt;br&gt; The No. 21 is Large"; } else {     echo "&lt;br&gt;The No. 8 is Large"; } ?&gt; &lt;/body&gt;&lt;/html&gt; </pre>	<pre> x=8 y=21 The No. 21 is Large </pre>

### The if...elseif...else Statement

The if...elseif...else statement executes different codes for more than two conditions.

#### Syntax

```

if (condition) {
    code to be executed if this condition is true;
} elseif (condition) {
    code to be executed if this condition is true;
} else {
    code to be executed if all conditions are false;
}

```

The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

Example	Output
<pre> &lt;html&gt;&lt;body&gt; &lt;?php echo "x=27&lt;br&gt;"; echo "y=7&lt;br&gt;"; echo "zy=17&lt;br&gt;"; \$x = 27; \$y = 7; \$z = 17;  if (\$x &gt; \$y) {   if (\$x &gt; \$z)   {     echo "&lt;br&gt;The Largest is:". \$x;   }   else   {     echo "&lt;br&gt; The Largest is:". \$z;   } } elseif (\$y &gt; \$z) {   echo "&lt;br&gt; The Largest is:". \$y; } else {   echo "&lt;br&gt; The Largest is:". \$z; } ?&gt; &lt;/body&gt;&lt;/html&gt; </pre>	<pre> x=27 y=7 zy=17  The Largest is:27 </pre>

## BRANCHING STATEMENTS

- Switch Statement
- Continue Statement
- Break Statement

### Switch Statement

The switch statement is used to perform different actions based on different conditions. Use the switch statement to select one of many blocks of code to be executed.

#### Syntax

```

switch (n) {
  case label1:
    code to be executed if n=label1;
    break;

```

```

case label2:
    code to be executed if n=label2;
    break;
case label3:
    code to be executed if n=label3;
    break;
...
default:
    code to be executed if n is different from all labels;
}

```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use `break` to prevent the code from running into the next case automatically. The default statement is used if no match is found.

Example	Output
<pre> &lt;html&gt; &lt;body&gt; &lt;?php \$favcolor = "red";  switch (\$favcolor) {     case "blue":         echo "Your favorite color is blue!";         break;     case "red":         echo "Your favorite color is red!";         break;     case "green":         echo "Your favorite color is green!";         break;     default:         echo "Your favorite color is neither red, blue, nor green!"; } ?&gt; &lt;/body&gt;&lt;/html&gt; </pre>	<p>Your favorite color is red!</p>

## LOOPS

When we write code, we want the same block of code to run over and over again in a row. Instead of adding several almost equal code-lines in a script, we can use loops to perform a task like this.

In PHP, we have the following looping statements:

- **while** - loops through a block of code as long as the specified condition is true
- **do...while** - loops through a block of code once, and then repeats the loop as long as the specified condition is true



- **for** - loops through a block of code a specified number of times
- **foreach** - loops through a block of code for each element in an array

## While Loop

The while loop executes a block of code as long as the specified condition is true.

### Syntax

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

The example below first sets a variable \$x to 1 (\$x = 1). Then, the while loop will continue to run as long as \$x is less than, or equal to 5 (\$x <= 5). \$x will increase by 1 each time the loop runs (\$x++):

Example	Output
<pre>&lt;html&gt;&lt;body&gt; &lt;?php \$x = 1;  while(\$x &lt;= 5) {     echo "The number is: \$x &lt;br&gt;";     \$x++; } ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>The number is: 1 The number is: 2 The number is: 3 The number is: 4 The number is: 5</pre>

## Do...While Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

### Syntax

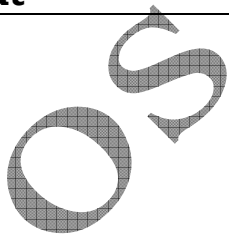
```
do {
    code to be executed;
} while (condition is true);
```

The example below first sets a variable \$x to 1 (\$x = 1). Then, the do while loop will write some output, and then increment the variable \$x with 1. Then the condition is checked (is \$x less than, or equal to 5?), and the loop will continue to run as long as \$x is less than, or equal to 5:

Example	Output
<pre>&lt;html&gt;&lt;body&gt; &lt;?php \$x = 1;  do {     echo "The number is: \$x &lt;br&gt;";     \$x++; } while (\$x &lt;= 5);</pre>	<pre>The number is: 1 The number is: 2 The number is: 3 The number is: 4 The number is: 5</pre>

<pre>?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	
----------------------------------------------	--

Notice that in a do while loop the condition is tested AFTER executing the statements within the loop. This means that the do while loop would execute its statements at least once, even if the condition is false the first time. The example below sets the \$x variable to 6, then it runs the loop, and then the condition is checked:

Example	Output
<pre>&lt;html&gt;&lt;body&gt; &lt;?php \$x = 6; do {     echo "The number is: \$x &lt;br&gt;";     \$x++; } while (\$x &lt;= 5); ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<p>The number is: 6</p> 

### For Loops

PHP for loops execute a block of code a specified number of times. The for loop is used when we know in advance how many times the script should run.

### Syntax

```
for (init counter; test counter; increment counter) {
    code to be executed;
}
```

### Parameters:

- init counter: Initialize the loop counter value
- test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- increment counter: Increases the loop counter value

The example below displays the numbers from 0 to 10:

Example	Output
<pre>&lt;html&gt;&lt;body&gt;  &lt;?php for (\$x = 0; \$x &lt;= 10; \$x++) {     echo "The number is: \$x &lt;br&gt;"; } ?&gt;  &lt;/body&gt; &lt;/html&gt;</pre>	<p>The number is: 0  The number is: 1  The number is: 2  The number is: 3  The number is: 4  The number is: 5  The number is: 6  The number is: 7  The number is: 8  The number is: 9  The number is: 10</p>

## foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

### Syntax

```
foreach ($array as $value) {  
    code to be executed;  
}
```

For every loop iteration, the value of the current array element is assigned to \$value and the array pointer is moved by one, until it reaches the last array element. The following example demonstrates a loop that will output the values of the given array (\$colors):

Example	Output
<pre>&lt;html&gt;&lt;body&gt; &lt;?php \$colors = array("red", "green", "blue", "yellow"); foreach (\$colors as \$value) {     echo "\$value &lt;br&gt;"; } ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	red green blue yellow

## CONTINUE STATEMENT

The continue statement is used within looping structures to skip the rest of the current loop iteration and continue execution at the condition evaluation and then the beginning of the next iteration. Note: In PHP the switch statement is considered a looping structure for the purposes of continue.

Sometimes a situation arises where we want to take the control to the beginning of the loop (for example for, while, do while etc.) skipping the rest statements inside the loop which have not yet been executed. The keyword continue allow us to do this. When the keyword continue executed inside a loop the control automatically passes to the beginning of loop. Continue is usually associated with the if statement.

Example	Output
<pre>&lt;html&gt;&lt;body&gt; &lt;?php for (\$i = 0; \$i &lt; 5; \$i++) {     if (\$i == 2)         continue;     echo \$i."&lt;br&gt;"; } ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	0 1 3 4

## BREAK STATEMENT

The keyword break ends execution of the current for, foreach, while, do while or switch structure. When the keyword break executed inside a loop the control automatically passes to the first statement outside the loop. A break is usually associated with the if statements. The break ends execution of the current for, foreach, while, do-while or switch structure. The break statement accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of. The default value is 1, only the immediate enclosing structure is broken out of.

Sometimes a situation arises where we want to exit from a loop immediately without waiting to get back to the conditional statement. The keyword break ends execution of the current for, foreach, while, do while or switch structure. When the keyword break executed inside a loop the control automatically passes to the first statement outside the loop. A break is usually associated with the if.

### Example :

In the following example we test the value of \$sum, if it is greater than 1500 the break statement terminate the execution of the code. As the echo statement is first statement outside loop it will print the current value of \$sum.

Example	Example
<pre>&lt;html&gt;&lt;body&gt; &lt;?php for (\$i = 0; \$i &lt; 8; \$i++) {     if (\$i == 4)         break;     echo \$i."&lt;br&gt;"; } ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>0 1 2 3</pre>

## PHP FUNCTIONS

The real power of PHP comes from its functions; it has more than 1000 built-in functions.

### User Defined Functions

Besides the built-in PHP functions, we can create our own functions. A function is a block of statements that can be used repeatedly in a program. It will not execute immediately when a page loads. A function will be executed by a call to the function. A user defined function declaration starts with the word "function". A function name can start with a letter or underscore (not a number). Function names are NOT case-sensitive. Give the function a name that reflects what the function does!

### Syntax

```
function functionName() {
    code to be executed;
}
```

In the example below, we create a function named "writeMsg()". The opening curly brace ( { ) indicates the beginning of the function code and the closing curly brace ( } ) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name:

Example	Output
<pre data-bbox="118 342 647 654">&lt;html&gt;&lt;body&gt; &lt;?php function writeMsg() {     echo "Hello world!"; } writeMsg(); // call the function ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<p data-bbox="826 342 1029 376">Hello world!</p>

### Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable. Arguments are specified after the function name, inside the parentheses. We can add as many arguments as we want, just separate them with a comma. The following example has a function with one argument (\$fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the name is used inside the function, which outputs several different first names, but an equal last name:

Example	Output
<pre data-bbox="118 1075 647 1547">&lt;html&gt;&lt;body&gt; &lt;?php function familyName(\$fname) {     echo "\$fname Refsnes.&lt;br&gt;"; } familyName("Jani"); familyName("Hege"); familyName("Stale"); familyName("Kai Jim"); familyName("Borge"); ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<p data-bbox="826 1075 1109 1272">Jani Refsnes. Hege Refsnes. Stale Refsnes. Kai Jim Refsnes. Borge Refsnes.</p>

The following example has a function with two arguments (\$fname and \$year);

Example	Output
<pre data-bbox="118 1686 798 2076">&lt;html&gt;&lt;body&gt; &lt;?php function familyName(\$fname, \$year) {     echo "\$fname Refsnes. Born in \$year &lt;br&gt;"; } familyName("Hege","1975"); familyName("Stale","1978"); familyName("Kai Jim","1983"); ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<p data-bbox="826 1686 1348 1798">Hege Refsnes. Born in 1975 Stale Refsnes. Born in 1978 Kai Jim Refsnes. Born in 1983</p>

## Default Argument Value

The following example shows how to use a default parameter. If we call the function `setHeight()` without arguments it takes the default value as argument:

Example	Output
<pre>&lt;html&gt;&lt;body&gt; &lt;?php function setHeight(\$minheight = 50) {     echo "The height is : \$minheight";     &lt;br&gt;; } setHeight(350); setHeight(); // will use the default value of 50 setHeight(135); setHeight(80); ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>The height is : 350 The height is : 50 The height is : 135 The height is : 80</pre>

## Functions - Returning values

To let a function return a value, use the return statement:

Example	Output
<pre>&lt;html&gt;&lt;body&gt; &lt;?php function sum(\$x, \$y) {     \$z = \$x + \$y;     return \$z; } echo "5 + 10 = " . sum(5, 10) . "&lt;br&gt;"; echo "7 + 13 = " . sum(7, 13) . "&lt;br&gt;"; echo "2 + 4 = " . sum(2, 4); ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>5 + 10 = 15 7 + 13 = 20 2 + 4 = 6</pre>