**PHP**

Hypertext Preprocessor

* PHP is a server side scripting language. that is used to develop Static websites or Dynamic websites or Web applications. PHP stands for Hypertext Pre-processor, that earlier stood for Personal Home Pages.
* PHP scripts can only be interpreted on a server that has PHP installed.
* The client computers accessing the PHP scripts require a web browser only.
* A PHP file contains PHP tags and ends with the extension “.php”.

## **Why use PHP?**

* PHP is **open source and free.**
* Short learning curve compared to other languages such as JSP, ASP etc.
* Large community document
* Most web hosting servers support PHP by default unlike other languages such as ASP that need IIS. This makes PHP a cost effective choice.
* PHP is regular updated to keep abreast with the latest technology trends.
* Other benefit that you get with PHP is that it’s a **server side scripting language**; this means you only need to install it on the server and client computers requesting for resources from the server do not need to have PHP installed; only a web browser would be enough.
* PHP has **in built support for working hand in hand with MySQL**; this doesn’t mean you can’t use PHP with other database management systems. You can still use PHP with
  + Postgres
  + Oracle
  + MS [SQL](https://www.guru99.com/sql.html) Server
  + ODBC etc.
* PHP is **cross platform;** this means you can deploy your application on a number of different operating systems such as windows, Linux, Mac OS etc.

## **What is a Scripting Language?**

* A script is a set of programming instructions that is interpreted at runtime.
* A scripting language is a language that interprets scripts at runtime. Scripts are usually embedded into other software environments.
* The purpose of the scripts is usually to enhance the performance or perform routine tasks for an application.
* Server side scripts are interpreted on the server while client side scripts are interpreted by the client application.

PHP is a server side script that is interpreted on the server while [JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html) is an example of a client side script that is interpreted by the client browser. Both PHP and JavaScript can be embedded into HTML pages

## **Programming Language Vs Scripting Language**

| **Programming language** | **Scripting language** |
| --- | --- |
| Has all the features needed to develop complete applications. | Mostly used for routine tasks |
| The code has to be compiled before it can be executed | The code is usually executed without compiling |
| Does not need to be embedded into other languages | Is usually embedded into other software environments. |

PHP was designed to work with HTML, and as such, it can be embedded into the HTML code.

What is PHP? Write your first PHP Program

You can create PHP files without any html tags and that is called Pure PHP file .

The server interprets the PHP code and outputs the results as HTML code to the web browsers.

In order for the server to identify the PHP code from the HTML code, we must always enclose the PHP code in PHP tags.

A PHP tag starts with the less than symbol followed by the question mark and then the words “php”.

PHP is a case sensitive language, “VAR” is not the same as “var”.

The PHP tags themselves are not case-sensitive, but it is strongly recommended that we use lower case letter. The code below illustrates the above point.

<?php … ?>

| **Summary**  * PHP stands for Hypertext pre-processor * PHP is a server side scripting language. This means that it is executed on the server. The client applications do not need to have PHP installed. * PHP files are saved with the “.php” file extension, and the PHP development code is enclosed in tags. * PHP is open source and cross platform |
| --- |

## **What is XAMPP?**

**XAMPP** is an open-source, cross-platform web server that consists of a web server, MySQL database engine, and PHP and [Perl](https://www.guru99.com/perl-tutorials.html) programming packages. **OR**XAMPP is one of the widely used cross-platform web servers, which helps developers to create and test their programs on a local webserver.

It is compiled and maintained by Apache. It allows users to create WordPress websites online using a local web server on their computer. It supports Windows, Linux, and Mac.

It is compiled and maintained by apache. The acronym XAMPP stands for;

* X – [cross platform operating systems] meaning it can run on any OS Mac OX , Windows , [Linux](https://www.guru99.com/unix-linux-tutorial.html) etc.
* A – [Apache](https://www.guru99.com/apache.html) – this is the web server software.
* M – MySQL – Database.
* P – [PHP](https://www.guru99.com/php-tutorials.html)
* P – Perl – scripting language

## **Why use XAMPP?**

XAMPP provides an easy-to-use control panel to manage Apache, MySQL, and other programs without using commands. To use PHP, we need to install Apache and MySQL. It’s not easy to install Apache and configure it as it needs to be set up and integrated with PHP and Perl, among other things. XAMPP deals with all the complexity to set up and integrate Apache with PHP and Perl.

Unlike [Java](https://www.guru99.com/java-tutorial.html) that runs with the Java SDK only, PHP requires a web-server to work.

## **PHP Data Types**

A Data type is the classification of data into a category according to its attributes;

* Alphanumeric characters are classified as strings
* Whole numbers are classified integers
* Numbers with decimal points are classified as floating points.
* True or false values are classified as Boolean.

PHP is a loosely typed language; it does not have explicit defined data types. PHP determines the data types by analyzing the attributes of data supplied. PHP implicitly supports the following data types

* Integer – whole numbers e.g. -3, 0, 69. The maximum value of an integer is platform-dependent. On a 32 bit machine, it’s usually around 2 billion. 64 bit machines usually have larger values. The constant PHP\_INT\_MAX is used to determine the maximum value.

<?php

echo PHP\_INT\_MAX;

?>

**Output:**

9223372036854775807

* Floating point number – decimal numbers e.g. 3.14. They are also known as double or real numbers. The maximum value of a float is platform-dependent. Floating point numbers are larger than integers.
* Character string – e.g. Hello World
* Boolean – e.g. True or false.

## **PHP Variable**

A variable is a name given to a memory location that stores data at runtime.

The scope of a variable determines its visibility.

A Php global variable is accessible to all the scripts in an application.

A local variable is only accessible to the script that it was defined in.

Think of a variable as a glass containing water. You can add water into the glass, drink all of it, refill it again etc.

The same applies for variables.

Variables are used to store data and provide stored data when needed. Just like in other programming languages, PHP supports variables too. Let’s now look at the rules followed when creating variables in PHP.

Let’s now look at how PHP determines the data type depending on the attributes of the supplied data.

<?php

$my\_var = 1;

echo $my\_var;

?>

Output:

1

**Floating point numbers**

<?php

$my\_var = 3.14;

echo $my\_var;

?>

Output:

3.14

**Character strings**

<?php

$my\_var ="Hypertext Pre Processor";

echo $my\_var;

?>

Output:

Hypertext Pre Processor

## **Use of Variables**

Variables help separate data from the program algorithms.

The same algorithm can be used for different input data values.

For example, suppose that you are developing a calculator program that adds up two numbers, you can create two variables that accept the numbers then you use the variables names in the expression that does the addition.

## **Variable Type Casting**

Performing arithmetic computations using variables in a language such as [C#](https://www.guru99.com/c-sharp-tutorial.html) requires the variables to be of the same data type.

Type casting is converting a variable or value into a desired data type.

This is very useful when performing arithmetic computations that require variables to be of the same data type.

* Type casting in PHP is done by the interpreter.

In other languages such as C#, you have to cast the variables. The code below shows type casting in C#.

* Above Code Output The var\_dump function is used to determine the data type. The code below demonstrates how to use the var\_dump function.

<?php

$a = 1;

var\_dump($a);

$b = 1.5;

var\_dump($b);

$c = "I Love PHP";

var\_dump($c);

$d = true;

var\_dump($d);

?>

**Output:**

int(1) float(1.5) string(10) "I Love PHP" bool(true)

## **PHP Constant**

**Define constant**– A constant is a variable whose value cannot be changed at runtime.

Suppose we are developing a program that uses the value of PI 3.14, we can use a constant to store its value.

Let’s now look at an example that defines a constant. define(‘PI’,3.14); //creates a constant with a value of 3.14 Once you define PI as 3.14 , writing a code like below will generate an error PI = 4; //PI has been defined as a constant therefore assigning a value is not permissible.

## **PHP Operators**

### **Arithmetic operators**

Arithmetic operators are used to perform arithmetic operations on numeric data. The concatenate operator works on strings values too. PHP supports the following operators.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| + | Addition | Summation of x and y | 1 + 1; | 2 |
| – | Subtraction | Difference between x and y | 1 – 1; | 0 |
| \* | Multiplication | Multiplies x and y | 3 \* 7; | 21 |
| / | Division | Quotient of x and y | 45 / 5; | 9 |
| % | PHP Modulus | Gives remainder of dividing x and y | 10 % 3; | 1 |
| -n | Negation | Turns n into a negative number | -(-5); | 5 |
| x . y | Concatenation | Puts together x and y | “PHP” . ” ROCKS”;10 . 3; | PHP ROCKS103 |

### **Assignment Operators**

Assignment operators are used to assign values to variables. They can also be used together with arithmetic operators.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| x = ? | assignment | Assigns the value of x to ? | $x = 5; | 5 |
| x += ? | addition | Increments the value of x by ? | $x = 2;$x += 1; | 3 |
| X -= ? | subtraction | Subtracts ? from the value of x | $x = 3;$x -= 2; | 1 |
| X \*=? | multiplication | Multiplies the value of x ? times | $x = 0;$x \*=9; | 0 |
| X /=? | division | Quotient of x and ? | $x = 6;$x /=3; | 2 |
| X %=? | modulus | The reminder of dividing x by? | $x = 3;$x %= 2; | 1 |
| X .=? | concatenate | Puts together items | ” $x = ‘Pretty’;$x .= ‘ Cool!’;” | Pretty Cool! |

### 

### **Comparison operators**

Comparison operators are used to compare values and data types.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| X == y | Equal | Compares x and y then returns true if they are equal | 1 == “1”; | True or 1 |
| X === y | identical | Compares both values and data types. | 1 === “1”; | False or 0. Since 1 is integer and “1” is string |
| X != y, x <> y | PHP Not equal | Compares values of x and y. returns true if the values are not equal | 2 != 1; | True or 1 |
| X > y | Greater than | Compares values of x and y. returns true if x is greater than y | 3 > 1; | True or 1 |
| X < y | Less than | Compares values of x and y. returns true if x is less than y | 2 < 1; | False or 0 |
| X >= y | Greater than or equal | Compares values of x and y. returns true if x is greater than or equal to y | 1 >=1 | True or 1 |
| X <= y | Less than or equal | Compares values of x and y. returns true if x is greater than or equal to y | 8 <= 6 | False or 0 |

### **Logical operators**

When working with logical operators, any number greater than or less than zero (0) evaluates to true. Zero (0) evaluates to false.

| **Operator** | **Name** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- | --- |
| X and y, x && y | And | Returns true if both x and y are equal | 1 and 4;True&& False; | True or 1False or 0 |
| X or y, x || y | Or | Returns true if either x or y is true | 6 or 9;0 || 0; | True or 1False or 0 |
| X xor y | Exclusive or, xor | Returns true if only x is true or only y is true | 1 xor 1;1 xor 0; | False or 0True or 1 |
| !x | Not | Returns true if x is false and false if x is true | !0; | True or 1 |

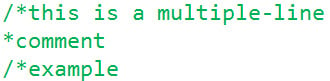
| **Summary**  * PHP is a loosely typed language. * Variables are memory locations used to store data * The value of constants cannot be changed at runtime * Type casting is used to convert a value or variable into a desired data type * Arithmetic operators are used to manipulate numeric data * Assignment operators are used to assign data to variables * Comparison operators are used to compare variables or values * Logical operators are used to compare conditions or values |
| --- |

## 

## **Why use Comments?**

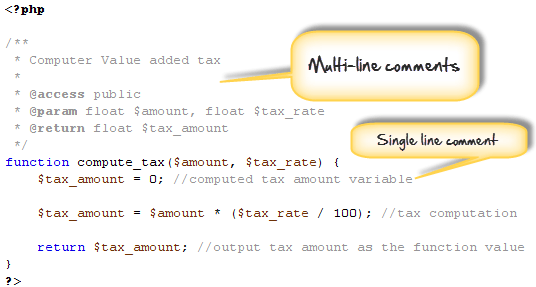
* If you don’t work on the source code for some time, it’s easy to forget what the code does. Commenting the source code helps remember what the code does.
* Commenting source code is also very important when multiple developers have to work on the same project. The changes made by one developer can be easily understood by other developers by simply reading the comments.
* As the best practice, you must have 3 lines of comments for every 10 lines of code

## **PHP Comments**

* Comments help us to understand the code
* Comments are explanations that we include in our source code. These comments are for human understanding.
* Single line comments start with double forward slashes // and they end in the same line.
* PHP Include, Require & Comments
* Multiple line comments start with a forward slash followed by the asterisk /\* and end with the asterisk followed by the forward slash \*/.
* 

The diagram below shows a PHP file with both multiple line and single line comments

PHP Example



## 

## 

## **PHP Include & PHP Include\_once**

The “include” php statement is used to include other files into a PHP file.

It has two variations, include and include\_once. Include\_once is ignored by the PHP interpreter if the file to be included.

The include statement has the following syntax

<?php

include 'file\_name';

?>

The include\_once statement has the following syntax

<?php

include\_once 'file\_name';

?>

HERE,

* “Include/include\_once” is the statement that includes file
* “’file\_name’” is the name of the file to be included.

## **Example : Include / Include\_once**

Suppose you are developing a website that contains the same navigation menu across all the pages.

You can create a common header then include it in every page using the include statement Let’s see how this can be done.

* We will create 2 files names
* header.php, index.php

Below are the codes for; *header.php*

<a href="index.php">Home</a>

<a href="aboutus.php">About us</a>

<a href="services.php">Services</a>

<a href="contactus.php">Contact Us</a>

*index.php*

<?php

include 'header.php';

?>

The header page above will output

## **PHP Require & PHP require\_once**

The require statement has two variations, require and require\_once.

The require/require\_once statement is used to include file.

Require\_once is ignored if the required file has already been added by any of the four include statements.

It has the following syntax

<?php

require 'file\_name';

?>

<?php

require\_once 'file\_name';

?>

HERE,

* “require/require\_once” is the statement that includes file
* “’file\_name’” is the name of the file to be included.

## **Example : Require**

Suppose we are developing a database powered application.

We can create a configuration file that we can include in all pages that connect to the database using the require statement. config.php

<?php

$config['host'] = 'localhost';

$config['db'] = 'my\_database';

$config['uid'] = 'root';

$config['password'] = '';

?>

Let’s now look at the sample code that requires the config file. *Pages\_model.php*

<?php

require 'config.php'; //require the config file

//other code for connecting to the database

?>

## **PHP include vs require**

The difference between include / require

| **Include** | **Require** |
| --- | --- |
| Issues a warning when an error occurs | Does not issue a warning |
| Execution of the script continues when an error occurs | Execution of the script stops when an error occurs. |

Generally, it’s recommended using the include statement so that when an error occurs, execution of the script continues to display the webmaster email address or the contact us page.

The require statement should be used if the entire script cannot run without the requested file.

The “include” and “require” statements can be used at any line in the source codes where you want the code to appear.

| **Summary**  * Single HTML code such as headers, footers, side bars etc. can be shared across many pages. This makes it easy to update the website by just updating a single file. * PHP code such as database configuration settings, custom functions etc. can be shared across many pages ensuring the website/application uses the same settings. * Comments are used to help understand source code. They are for human understanding * Single line comment statements start with double forward slashes //. * Multi-line comment statements are enclosed between /\* statements \*/. * The “include, include\_once, require and require\_once” statements are used to include files. * Include\_once/require\_once is ignored if the requested file has already been included using any of the four statements. * The “include” statement issues a warning and continues with the execution if the requested file has not been found. * The require statement raises a fatal error and stops the script execution. * The “include” statement should be in most cases except in situations where without the requested file to be included, the entire script cannot run. |
| --- |

## **What is a PHP Array?**

A PHP array is a variable that stores more than one piece of related data in a single variable.

Think of an array as a box of chocolates with slots inside.

The box represents the array itself while the spaces containing chocolates represent the values stored in the arrays.

## **Numeric Arrays**

Numeric arrays use number as access keys.

An access key is a reference to a memory slot in an array variable.

The access key is used whenever we want to read or assign a new value an array element.

Below is the syntax for creating numeric array in php.

Array Example

<?php

$variable\_name[n] = value;

?>

Or

<?php

$variable\_name = array(n => value, …);

?>

HERE,

* “$variable\_name…” is the name of the variable
* “[n]” is the access index number of the element
* “value” is the value assigned to the array element.

Let’s now look at an example of a numeric array.

Suppose we have 5 movies that we want to store in array variables.

We can use the example shown below to do that.

<?php

$movie[0] = 'Shaolin Monk';

$movie[1] = 'Drunken Master';

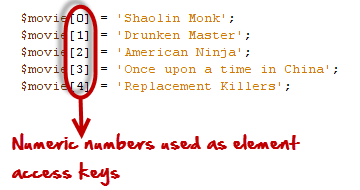
$movie[2] = 'American Ninja';

$movie[3] = 'Once upon a time in China';

$movie[4] = 'Replacement Killers';

?>

Here,



Each movie is given an index number that is used to retrieve or modify its value. Observe the following code-

<?php

$movie[0]="Shaolin Monk";

$movie[1]="Drunken Master";

$movie[2]="American Ninja";

$movie[3]="Once upon a time in China";

$movie[4]="Replacement Killers";

echo $movie[3];

$movie[3] = " Eastern Condors";

echo $movie[3];

?>

**Output:**

Once upon a time in China Eastern Condors

As you can see from the above examples, working with arrays in PHP when dealing with multiple values of the same nature is very easy and flexible.

Alternatively, the above array variables can also be created using the following code.

<?php

$movie = array(0 => "Shaolin Monk",

1 => "Drunken Master",

2 => "American Ninja",

3 => "Once upon a time in China",

4 =>"Replacement Killers" );

echo $movie[4];

?>

**Output:**

Replacement Killers

## **PHP Associative Array**

Associative array differ from numeric array in the sense that associative arrays use descriptive names for id keys.

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 name of the variable
* “[‘key\_name’]” is the access index number of the element
* “value” is the value assigned to the array element.

Let’s suppose that we have a group of persons, and we want to assign the gender of each person against their names.

We can use an associative array to do that.The code below helps us to do that.

<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

print\_r($persons);

echo "";

echo "Mary is a " . $persons["Mary"];

?>

**HERE,**

****

**Output:**

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

Associative array are also very useful when retrieving data from the database.

The field names are used as id keys.

## **PHP Multi-dimensional arrays**

These are arrays that contain other nested arrays.

The advantage of multidimensional arrays is that they allow us to group related data together.

Let’s now look at a practical example that implements a php multidimensional array.

The table below shows a list of movies by category.

| **Movie title** | **Category** |
| --- | --- |
| Pink Panther | Comedy |
| John English | Comedy |
| Die Hard | Action |
| Expendables | Action |
| The Lord of the rings | Epic |
| Romeo and Juliet | Romance |
| See no evil hear no evil | Comedy |

The above information can be represented as a multidimensional array. The code below shows the implementation.

<?php

$movies =array(

"comedy" => array("Pink Panther", "John English", "See no evil hear no evil"),

"action" => array("Die Hard", "Expendables"),

"epic" => array("The Lord of the rings"),

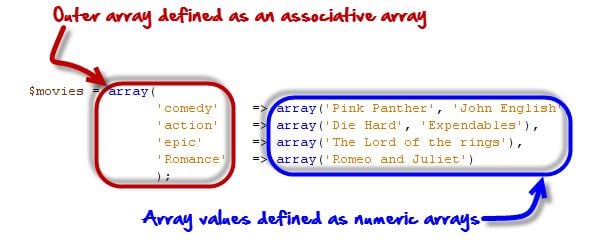
"Romance" => array("Romeo and Juliet")

);

print\_r($movies);

?>

HERE,



**Output:**

Array ( [comedy] => Array ( [0] => Pink Panther [1] => John English [2] => See no evil hear no evil ) [action] => Array ( [0] => Die Hard [1] => Expendables ) [epic] => Array ( [0] => The Lord of the rings ) [Romance] => Array ( [0] => Romeo and Juliet ) )

Another way to define the same array is as follows

<?php

$film=array(

"comedy" => array(

0 => "Pink Panther",

1 => "john English",

2 => "See no evil hear no evil"

),

"action" => array (

0 => "Die Hard",

1 => "Expendables"

),

"epic" => array (

0 => "The Lord of the rings"

),

"Romance" => array

(

0 => "Romeo and Juliet"

)

);

echo $film["comedy"][0];

?>

**Output:**

Pink Panther

Note: the movies numeric array has been nested inside the categories associative array

## **PHP Arrays: Operators**

| **Operator** | **Name** | **Description** | **How to do it** | **Output** |
| --- | --- | --- | --- | --- |
| x + y | Union | Combines elements from both arrays | <?php  $x = array('id' => 1);  $y = array('value' => 10);  $z = $x + $y;  ?> | Array([id] => 1 [value] => 10) |
| X == y | Equal | Compares two arrays if they are equal and returns true if yes. | <?php  $x = array("id" => 1);  $y = array("id" => "1");  if($x == $y)  {  echo "true";  }  else  {  echo "false";  }  ?> | True or 1 |
| X === y | Identical | Compares both the values and data types | <?php  $x = array("id" => 1);  $y = array("id" => "1");  if($x === $y)  {  echo "true";  }  else  {  echo "false";  }  ?> | False or 0 |
| X != y, x <> y | Not equal |  | <?php  $x = array("id" => 1);  $y = array("id" => "1");  if($x != $y)  {  echo "true";  }  else  {  echo "false";  }  ?> | False or 0 |
| X !== y | Non identical |  | <?php  $x = array("id" => 1);  $y = array("id" => "1");  if($x !== $y)  {  echo "true";  }  else  {  echo "false";  }  ?> | True or 1 |

## **PHP Array Functions**

### **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

### **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

### **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 )

### **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 )

### **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 )

### **Why use arrays?**

* Contents of Arrays can be stretched,
* Arrays easily help group related information such as server login details together
* Arrays help write cleaner code.

| **Summary**  * Arrays are special variables with the capacity to store multi values. * Arrays are flexibility and can be easily stretched to accommodate more values * Numeric arrays use numbers for the array keys * PHP Associative array use descriptive names for array keys * Multidimensional arrays contain other arrays inside them. * The count function is used to get the number of items that have been stored in an array * The is\_array function is used to determine whether a variable is a valid array or not. * Other array functions include sort, ksort, assort etc. |
| --- |

# **PHP Control Structures: If else, Switch Case**

## **What is a control structure?**

Code execution can be grouped into categories as shown below

* **Sequential** – this one involves executing all the codes in the order in which they have been written.
* **Decision** – this one involves making a choice given a number of options. The code executed depends on the value of the condition.

A control structure is a block of code that decides the execution path of a program depending on the value of the set condition.

Let’s now look at some of the control structures that PHP supports.

## **PHP IF Else**

If… then… else is the **simplest** **control** **structure**. It evaluates the conditions using Boolean logic

**When to use if… then… else**

* You have a block of code that should be executed only if a certain condition is true
* You have two options, and you have to select one.
* If… then… else if… is used when you have to select more than two options and you have to select one or more

**Syntax** The syntax for if… then… else is;

<?php

if (condition is true) {

block one

else

block two

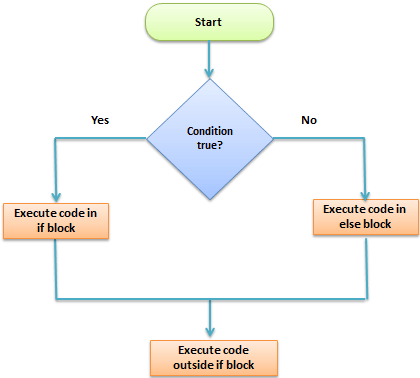
}

?>

**HERE,**

* “**if (condition is true)”** is the control structure
* “**block one**” is the code to be executed if the condition is true
* **{…else…}** is the fallback if the condition is false
* “**block two**” is the block of code executed if the condition is false

**How it works** The flow chart shown below illustrates how the if then… else control structure works



**Let’s see this in action** The code below uses “if… then… else” to determine the larger value between two numbers.

<?php

$first\_number = 7;

$second\_number = 21;

if ($first\_number > $second\_number){

echo "$first\_number is greater than $second\_number";

}else{

echo "$second\_number is greater than $first\_number";

}

?>

**Output:**

21 is greater than 7

## **PHP Switch Case**

**Switch… case** is similar to the **if then… else** control structure.

It only **executes** a single block of code depending on the **value** of the condition.

If no condition has been met then the default block of code is executed.

It has the following basic syntax.

<?php

switch(condition){

case value:

//block of code to be executed

break;

case value2:

//block of code to be executed

break;

default:

//default block code

break;

}

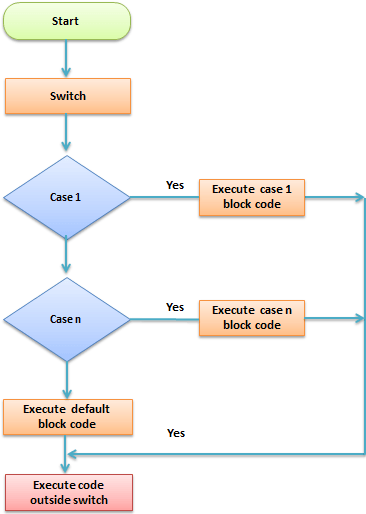
?>

**HERE,**

* **“switch(…){…}”** is the control structure block code
* **“case value: case…”** are the blocks of code to be executed depending on the value of the condition
* **“default:”** is the block of code to be executed when no value matches with the condition

**How it works**

The flow chart shown below illustrates how the switch control structure works



**Practical example**

The code below uses the switch control structure to display a message depending on the day of the week.

<?php

$today = "wednesday";

switch($today){

case "sunday":

echo "pray for us sinners.";

break;

case "wednesday":

echo "ladies night, take her out for dinner";

break;

case "saturday":

echo "take care as you go out tonight.";

break;

default:

echo "have a nice day at work";

break;

}

?>

**Output:**

ladies night, take her out for dinner

| **Summary**   * Control structures are used to control the execution of the program * The if then… else is when you have more than route block of code to execute depending on the value of the condition * Switch… case is used to when you have a number of block codes, and you only have to execute one of them depending on the value of the set case. |
| --- |

# **PHP Loop: For, ForEach, While, Do While [Example]**

A Loop is an Iterative Control Structure that involves executing the same number of code a number of times until a certain condition is met.

### **PHP For Loop**

The above code outputs “21 is greater than 7” For loops For… loops execute the block of code a specifiednumber of times. There are basically two types of for loops;

* for
* for… each.

Let’s now look at them separately. **For loop** It has the following basic **syntax**

<?php

for (initialize; condition; increment){

//code to be executed

}

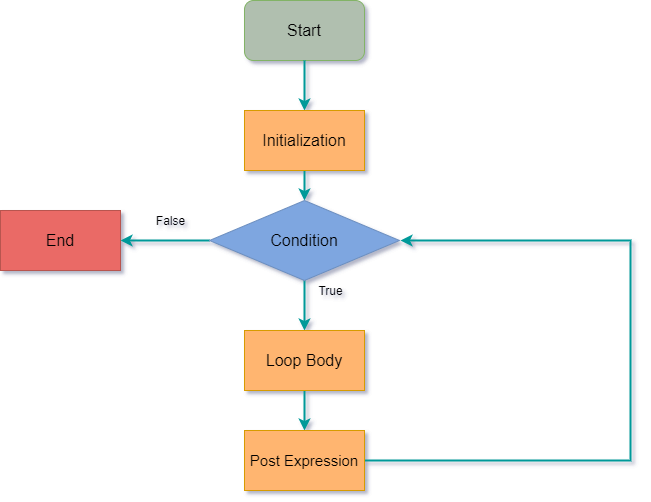
?>

**HERE,**

* **“for…{…}”** is the loop block
* “**initialize**” usually an integer; it is used to set the counter’s initial value.
* **“condition”** the condition that is evaluated for each php execution. If it evaluates to true, then execution of the for… loop continues. If it evaluates to false, the execution of the for… loop is terminated.
* **“increment”** is used to increment the initial value of counter integer.

**How it works**

The flowchart shown below illustrates how for loop in php works



**How to code**

The code below uses the “for… loop” to print values of multiplying 10 by 0 through to 10

<?php

for ($i = 0; $i < 10; $i++){

$product = 10 \* $i;

echo "The product of 10 \* $i is $product <br/>";

}

?>

**Output:**

The product of 10 x 0 is 0

The product of 10 x 1 is 10

The product of 10 x 2 is 20

The product of 10 x 3 is 30

The product of 10 x 4 is 40

The product of 10 x 5 is 50

The product of 10 x 6 is 60

The product of 10 x 7 is 70

The product of 10 x 8 is 80

The product of 10 x 9 is 90

### **PHP For Each loop**

The php foreach loop is used to iterate through array values. It has the following basic syntax

<?php

foreach($array\_variable as $array\_values){

block of code to be executed

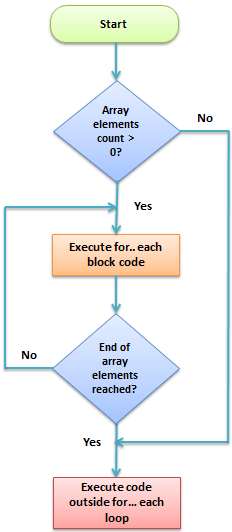
}

?>

**HERE,**

* **“foreach(…){…}”** is the foreach php loop block code
* **“$array\_data”** is the array variable to be looped through
* **“$array\_value “** is the temporary variable that holds the current array item values.
* “block of code…” is the piece of code that operates on the array values

**How it works** The flowchart shown below illustrates how the for… each… loop works



**Practical examples**

The code below uses for… each loop to read and print the elements of an array.

<?php

$animals\_list = array("Lion","Wolf","Dog","Leopard","Tiger");

foreach($animals\_list as $array\_values){

echo $array\_values . "<br>";

}

?>

**Output:**

Lion

Wolf

Dog

Leopard

Tiger

Let’s look at another example that loops through an **associative array**.

An associative array uses alphanumeric words for access keys.

<?php

$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

foreach($persons as $key => $value){

echo "$key is $value"."<br>";

}

?>

The names have been used as array keys and gender as the values.

**Output:**

Mary is Female

John is Male

Mirriam is Female

## **While Loop**

### **PHP While loop**

They are used to execute a block of code a repeatedly until the set condition gets satisfied

**When to use while loops**

* While loops are used to execute a block of code until a certain condition becomes true.
* You can use a while loop to read records returned from a database query.

**Types of while loops**

* **Do… while** – executes the block of code at least once before evaluating the condition
* **While…** – checks the condition first. If it evaluates to true, the block of code is executed as long as the condition is true. If it evaluates to false, the execution of the while loop is terminated.

**While loop**

It has the following syntax

<?php

while (condition){

block of code to be executed;

}

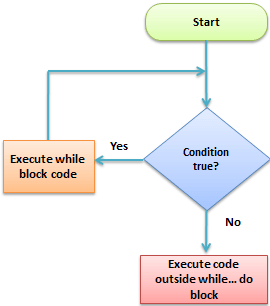
?>

**HERE,**

* **“while(…){…}”** is the while loop block code
* **“condition”** is the condition to be evaluated by the while loop
* **“block of code…”** is the code to be executed if the condition gets satisfied

**How it works**

The flow chart shown below illustrates how the while… loop works



**Practical example**

The code below uses the while… loop to print numbers 1 to 5.

<?php

$i = 0;

while ($i < 5){

echo $i + 1 . "<br>";

$i++;

}

?>

**Output:**

1

2

3

4

5

### **PHP Do While**

The difference between While… loop and Do… while loop is do… while is executed at-least once before the condition is evaluated.

Let’s now look at the basic syntax of a do… while loop

<?php

do{

block of code to be executed

}

?>

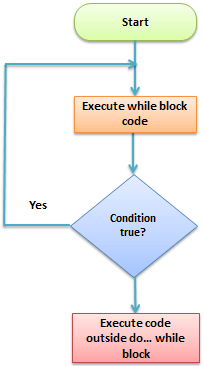
while(condition);

**HERE,**

* **“do{…} while(…)”** is the do… while loop block code
* **“condition”** is the condition to be evaluated by the while loop
* **“block of code…”** is the code that is executed at least once by the do… while loop

**How it works**

The flow chart shown below illustrates how the while… loop works



**Practical example**

We are now going to modify the while… loop example and implement it using the do… while loop and set the counter initial value to 9.

The code below implements the above modified example

<?php

$i = 9;

do{

echo "$i is"." <br>";

}

while($i < 9);

?>

**The above code outputs:**

9

***Note*** *the above example outputs 9 only.*

*This is because the do… while loop is executed at least once even if the set condition evaluates to false.*

| **Summary**  * The for… loop is used to execute a block of a specified number of times * The foreach… loop is used to loop through arrays * While… loop is used to execute a block of code as long as the set condition is made to be false * The do… while loop is used to execute the block of code at least once then the rest of the execution is dependent on the evaluation of the set condition |
| --- |

# **PHP Strings: PHP String Functions Explained with Examples**

## **What is String in PHP?**

A string is a collection of characters. String is one of the data types supported by PHP.

The string variables can contain alphanumeric characters. Strings are created when;

* You declare variable and assign string characters to it
* You can directly use PHP Strings with echo statement.
* PHP String functions are language construct, it helps capture words.
* Learning how strings work in PHP and how to manipulate them will make you a very effective and productive developer.

## **PHP Create Strings Using Single quotes with Example**

Let’s now look at the four different ways of creating PHP string functions and string manipulation in PHP.

Creating PHP Strings Using Single quotes: The simplest way to create a string is to use single quotes.

Let’s look at an example that creates a simple string in PHP.

<?php

var\_dump('You need to be logged in to view this page');

?>

**Output:**

string(42) "You need to be logged in to view this page"

If the single quote is part of the string value, it can be escaped using the backslash.

The code below illustrates how to escape a single quote.

<?php

echo 'I \'ll be back after 20 minutes';

?>

**Output:**

I'll be back after 20 minutes

## **PHP Create Strings Using Double quotes with Example**

The double quotes are used to create relatively complex strings compared to single quotes.

Variable names can be used inside double quotes and their values will be displayed.

Let’s look at an example.

<?php

$name='Alicia';

echo "$name is friends with kalinda";

?>



HERE,

* The above example creates a simple string with the value of Alicia.
* The variable name is then used in the string created using double quotes and its value is interpolated at run time.

**Output:**

Alicia is friends with kalinda

In addition to variable interpolations, the double quote string can also escape more special characters such as “\n for a linefeed, \$ dollar for the dollar sign” etc.

More examples Let’s suppose that we have the following code

<?php $pwd = "pas$word"; echo $pwd; ?>

**Output:**

NOTICE : Undefined variable

pas

executing the above codes issues a notice “Notice: Undefined variable”.

This is because $word is treated as a variable.

If we want the dollar sign to be treated as a literal value, we have to escape it.

<?php

$word="word";

$pwd = "pas\$word";

echo $pwd; ?>

**Output:**

pas$word

## **PHP Heredoc with Example**

This heredoc methodology is used to create fairly complex strings as compared to double quotes.

The heredoc supports all the features of double quotes and allows creating string values with more than one line without PHP string concatenation.

Using double quotes to create strings that have multiple lines generates an error.

You can also use double quotes inside without escaping them.

The example below illustrates how the Heredoc method is used to create string values.

<?php

$baby\_name = "Shalon";

echo <<<EOT

When $baby\_name was a baby,

She used to look like a "boy".

EOT;

?>

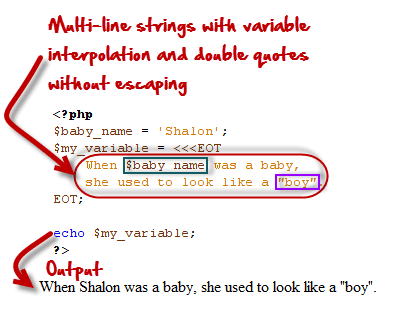
HERE,

**<<<EOT** is the string delimiter.

EOT is the acronym for end of text.

It should be defined in its on line at the beginning of the string and at the end.

Note: you can use anything you like in place of EOT



**Output:**

When Shalon was a baby, She used to look like a "boy".

## **PHP Nowdoc with Example**

The Nowdoc string creation method is similar to the heredoc method but works like the way single quotes work.

No parsing takes place inside the Nowdoc.

Nowdoc is ideal when working with raw data that do not need to be parsed.

The code below shows the Nowdoc implementation

<?php

$baby\_name = "Shalon";

$my\_variable = <<<'EOT'

When $baby\_name was a baby,

She used to look like a "boy".

EOT;

echo $my\_variable;

?>

**Output:**

When $baby\_name was a baby, She used to look like a "boy".

## **PHP String Function Examples**

String functions in PHP are used to manipulate string values.

We are now going to look at some of the commonly used string functions in PHP

| **Function** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| strtolower | Used to convert all string characters to lower case letters | echo strtolower( ‘Benjamin’); | outputs benjamin |
| strtoupper | Used to convert all string characters to upper case letters | echo strtoupper(‘george w bush’); | outputs GEORGE W BUSH |
| strlen | The string length function is used to count the number of character in a string. Spaces in between characters are also counted | echo strlen(‘united states of america’); | 24 |
| explode | Used to convert strings into an array variable | $settings = explode(‘;’, “host=localhost; db=sales; uid=root; pwd=demo”); print\_r($settings); | Array ( [0] => host=localhost [1] => db=sales [2] => uid=root [3] => pwd=demo ) |
| substr | Used to return part of the string. It accepts three (3) basic parameters. The first one is the string to be shortened, the second parameter is the position of the starting point, and the third parameter is the number of characters to be returned. | $my\_var = ‘This is a really long sentence that I wish to cut short’;echo substr($my\_var,0, 12).’…’; | This is a re… |
| str\_replace | Used to locate and replace specified string values in a given string. The function accepts three arguments. The first argument is the text to be replaced, the second argument is the replacement text and the third argument is the text that is analyzed. | echo str\_replace (‘the’, ‘that’, ‘the laptop is very expensive’); | that laptop is very expensive |
| strpos | Used to locate the and return the position of a character(s) within a string. This function accepts two arguments | echo strpos(‘PHP Programing’,’Pro’); | 4 |
| sha1 | Used to calculate the SHA-1 hash of a string value | echo sha1(‘password’); | 5baa61e4c 9b93f3f0 682250b6cf8331b 7ee68fd8 |
| md5 | Used to calculate the md5 hash of a string value | echo md5(‘password’); | 9f961034ee 4de758 baf4de09ceeb1a75 |
| str\_word\_count | Used to count the number of words in a string. | echo str\_word\_count (‘This is a really long sentence that I wish to cut short’); | 12 |
| ucfirst | Make the first character of a string value upper case | echo ucfirst(‘respect’); | Outputs Respect |
| lcfirst | Make the first character of a string value lower case | echo lcfirst(‘RESPECT’); | Outputs rESPECT |

For a complete list of PHP strings, check

## **Summary**

* Define string in PHP: A string function in PHP is a set of characters
* Explain string function in PHP: Strings are created when you declare a variable and assign string characters to it.
* Single quotes are used to specify simple strings in PHP
* Double quotes are used to create fairly complex strings in PHP
* heredoc is used to create complex strings
* Nowdoc is used to create strings that cannot be parsed.

## **What is a Function in PHP?**

A **Function in PHP** is a reusable piece or block of code that performs a specific action. It takes input from the user in the form of parameters, performs certain actions, and gives the output. Functions can either return values when called or can simply perform an operation without returning any value.

PHP has over 700 functions built in that perform different tasks.

**In this tutorial, you will learn-**

* [Why use Functions?](https://www.guru99.com/functions-in-php.html#1)
* [PHP Built in Functions](https://www.guru99.com/functions-in-php.html#2)
* [String Functions](https://www.guru99.com/functions-in-php.html#3)
* [Numeric Functions](https://www.guru99.com/functions-in-php.html#4)
* [Date Function](https://www.guru99.com/functions-in-php.html#5)
* [Why use User Defined Functions?](https://www.guru99.com/functions-in-php.html#6)

## **Why use Functions?**

* Better code organization – PHP functions allow us to group blocks of related code that perform a specific task together.
* Reusability – once defined, a function can be called by a number of scripts in our PHP files. This saves us time of reinventing the wheel when we want to perform some routine tasks such as connecting to the database
* Easy maintenance- updates to the system only need to be made in one place.

## **PHP Built in Functions**

Built in functions are predefined functions in PHP that exist in the installation package.

These PHP inbuilt functions are what make PHP a very efficient and productive scripting language.

The built in functions of PHP can be classified into many categories. Below is the list of the categories.

## **String Functions**

These are functions that manipulate string data, refer to the article on strings for implementation examples of string functions

## **Numeric Functions**

Numeric functions in PHP are the functions that return numeric results.

Numeric php function can be used to format numbers, return constants, perform mathematical computations etc.

**The table below shows the common PHP numeric functions**

| **Function** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| is\_number | Accepts an argument and returns true if its numeric and false if it’s not | <?php  if(is\_numeric("guru"))  {  echo "true";  }  else  {  echo "false";  }  ?> | false |
| <?php  if(is\_numeric (123))  {  echo "true";  }  else  {  echo "false";  }  ?> | true |
| number\_format | Used to formats a numeric value using digit separators and decimal points | <?php  echo number\_format(2509663);  ?> | 2,509,663 |
| rand | Used to generate a random number. | <?php  echo rand();  ?> | Random number |
| round | Round off a number with decimal points to the nearest whole number. | <?php  echo round(3.49);  ?> | 3 |
| sqrt | Returns the square root of a number | <?php  echo sqrt(100);  ?> | 10 |
| cos | Returns the cosine | <?php  echo cos(45);  ?> | 0.52532198881773 |
| sin | Returns the sine | <?php  echo sin(45);  ?> | 0.85090352453412 |
| tan | Returns the tangent | <?php  echo tan(45);  ?> | 1.6197751905439 |
| pi | Constant that returns the value of PI | <?php  echo pi();  ?> | 3.1415926535898 |

## **Date Function**

The date function is used to format [Unix](https://www.guru99.com/unix-linux-tutorial.html) date and time to human readable format.

Check the article on PHP date functions for more details.

Other functions

These include;

* Arrays – see the article on arrays for examples
* Files – see the article on files for examples
* Database functions – see the article on [MySQL PHP and other database access methods](https://www.guru99.com/mysql-php-and-other-database-access-methods.html) v2

## **Why use User Defined Functions?**

User defined functions come in handy when;

* you have routine tasks in your application such as adding data to the database
* performing validation checks on the data
* Authenticating users in the system etc.

These activities will be spread across a number of pages.

Creating a function that all these pages can be calling is one of the features that make PHP a powerful scripting language.

Before we create our first user defined function, let’s look at the rules that we must follow when creating our own functions.

* Function names must start with a letter or an underscore but not a number
* The function name must be unique
* The function name must not contain spaces
* It is considered a good practice to use descriptive function names.
* Functions can optionally accept parameters and return values too.

Let’s now create our first function. We will create a very basic function that illustrates the major components of a function in PHP.

<?php

//define a function that displays hello function

function add\_numbers(){

echo 1 + 2;

}

add\_numbers ();

?>

**Output:**

3

HERE,

* “function…(){…}” is the function block that tells PHP that you are defining a custom function
* “add\_numbers” is the function name that will be called when using the function.
* “()” can be used to pass parameters to the function.
* “echo ‘Hello function!’;” is the function block of code that is executed. It could be any code other than the one used in the above example.

Let’s now look at a fairly complex example that accepts a parameter and display a message just like the above function.

Suppose we want to write a function that prints the user name on the screen, we can write a custom function that accepts the user name and displays it on the screen.

**The code below shows the implementation.**

<?php

function display\_name($name)

{

echo "Hello " . $name;

}

display\_name("Martin Luther King");

?>

**Output:**

Hello Martin Luther King

HERE,

* “…($name){…” is the function parameter called name and is initialized to nameless. If no parameter is passed to the function, nameless will be displayed as the name. This comes in handy if not supplying any parameter to the function can result in unexpected errors.

Let’s now look at a function that accepts a parameter and then returns a value. We will create a function that converts kilometers to miles. The kilometers will be passed as a parameter. The function will return the miles equivalent to the passed kilometers. The code below shows the implementation.

<?php

function kilometers\_to\_miles($kilometers = 0)

{

$miles\_scale = 0.62;

return $kilometers \* $miles\_scale;

}

echo kilometers\_to\_miles(100);

?>

**Output:**

62

| **Summary**  * Define function in PHP: Function is a block of code that performs specific task. * Built in function in PHP is a function that is shipped with PHP * PHP has over 700 built in functions * String functions manipulate string data * Numeric functions manipulate numeric data * Date functions manipulate date data * Other functions such as is\_array, fopen etc. are used to manipulate arrays and files respectively * User defined functions are functions that you can create yourself to enhance PHP |
| --- |

# **PHP Session & PHP Cookies with Example**

## **What is Cookie?**

A cookie is a small file with the maximum size of 4KB that the web server stores on the client computer.

Once a cookie has been set, all page requests that follow return the cookie name and value.

A cookie can only be read from the domain that it has been issued from. For example, a cookie set using the domain [www.estuate.com](http://www.estuate.com) can not be read from the domain [career.guru99.com](https://career.guru99.com/).

Most of the websites on the internet display elements from other domains such as advertising. The domains serving these elements can also set their own cookies. These are known as third party cookies.

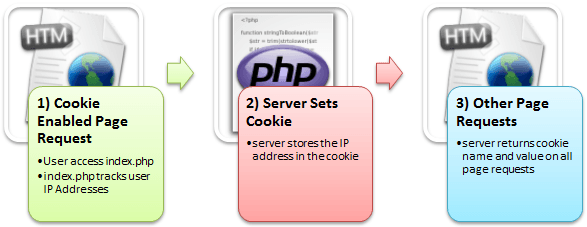
A cookie created by a user can only be visible to them. Other users cannot see its value.

Most web browsers have options for disabling cookies, third party cookies or both.

If this is the case then PHP responds by passing the cookie token in the URL.

**The diagram shown below illustrates how cookies work.**

Here,

1) A user requests for a page that stores cookies****

2) The server sets the cookie on the user’s computer

3) Other page requests from the user will return the cookie name and value

PHP Interview Questions and Answer

1. **What is PHP**

PHP stands for Hypertext preprocessor.

It is a server side scripting language embedded into HTML generally used for

dynamic web development.

It also works efficiently with almost 20 different databases such as MysQL

PostgreSQL etc.

1. **What is session in PHP**

Sessions in PHP are used to store data for users under a unique ID which can be then

used across multiple webpages of a website

1. **What is PEAR in PHP.**

PEAR stands for PHP Extension and Application Repository.

It is a framework and repository for the reusable components of PHP containing

code snippets and libraries.

1. **Is PHP Considered a Case-Sensitive Language?**

It is called partially case-sensitive since variable names in PHP are

case-sensitive while the functions and user-defined functions are not

1. **What are the different types of data type in PHP.**
2. Integer
3. Float
4. String
5. Array
6. Boolean
7. Null
8. Object
9. Resource

**6) How is "echo" different from "print"?**

| **echo** | **print** |
| --- | --- |
| * It can output multiple strings * No return value * Faster | * It can output a single string * Always return 1 * Slower |

**7) What are the Rules for Naming a Variable?**

* Variable name must always begin with underscore or letter.
* Variable names can include numbers but not special characters.
* A variable in PHP declared using $' sign followed by the variable name.