**What is PHP?**

=>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.

=>A PHP file contains PHP tags and ends with the extension “.php”.

## PHP Data Types

* 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.

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.

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 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.

## 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 |

## PHP Comments

:-

[PHP Include, Require & Comments](https://www.guru99.com/images/2013/04/php_single_comment.jpg)single line comment

/\* any thinkkkkk \*/ multiline comment….

## What is a PHP Array?

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

* [Numeric Arrays](https://www.guru99.com/arrays.html#1):-

$movie[0] = 'Shaolin Monk';

* [PHP Associative Array](https://www.guru99.com/arrays.html#2):-

$variable\_name['key\_name'] = value;

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

* [PHP Multi-dimensional arrays](https://www.guru99.com/arrays.html#3):-

$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")

);

* [PHP Array operators](https://www.guru99.com/arrays.html#4):-

|  |  |
| --- | --- |
| x + y | Union |
| X == y | Equal |
| X === y | Identical |
| X != y, x <> y | Not equal |
| X !== y | Non identical |

## 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);

?>

### 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);

?>

### 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);

?>

### 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);

?>

### 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);

?>

**Let's introduce some Logic!**

## 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.

## PHP IF Else

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

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

<?php

if (condition is true) {

block one

else

block two

}

?>

Example:-

<?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";

}

?>

## 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;

}

?>

Example:-

<?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;

}

?>

# PHP Loop

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

<?php

for (initialize; condition; increment){

//code to be executed

}

?>

Example:-

<?php

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

$product = 10 \* $i;

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

}

?>

### 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

}

?>

**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>";

}

?>

Both for loop and foreach loop are used to repeat a set of statements, but the syntax is different. The key difference between for Loop and foreach loop is that **the for loop is a general purpose control structure while the foreach loop is an enhanced for loop that is applicable only to arrays and collections**.

## While Loop

### PHP While loop

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

**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;

}

?>

**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++;

}

?>

### 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.

<?php

do{

block of code to be executed

}

?>

Example:-

<?php

$i = 9;

do{

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

}

while($i < 9);

?>

## What is String in PHP?

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

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

<?php

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

?>

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

<?php

$name='Alicia';

echo "$name is friends with kalinda";

?>

**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.

**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;

?>

**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 [https://php.net/manual/en/ref.strings.php](https://www.php.net/manual/en/ref.strings.php)

## strrev() - Reverse a String

The PHP strrev() function reverses a string.

### **Example**

Reverse the string "Hello world!":

<?php  
echo strrev("Hello world!"); // outputs !dlrow olleH  
?>

## PHP NaN

NaN stands for Not a Number.

NaN is used for impossible mathematical operations.

PHP has the following functions to check if a value is not a number:

* [is\_nan()](https://www.w3schools.com/php/func_math_is_nan.asp)

## PHP Casting Strings and Floats to Integers

Sometimes you need to cast a numerical value into another data type.

The (int), (integer), or intval() function are often used to convert a value to an integer.

### **Example**

Cast float and string to integer:

<?php  
// Cast float to int  
$x = 23465.768;  
$int\_cast = (int)$x;  
echo $int\_cast;  
  
echo "<br>";  
  
// Cast string to int  
$x = "23465.768";  
$int\_cast = (int)$x;  
echo $int\_cast;  
?>

## PHP min() and max() Functions

The min() and max() functions can be used to find the lowest or highest value in a list of arguments:

### **Example**

<?php  
echo(min(0, 150, 30, 20, -8, -200));  // returns -200  
echo(max(0, 150, 30, 20, -8, -200));  // returns 150  
?>

## PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

**Note:** Unlike variables, constants are automatically global across the entire script.

## Create a PHP Constant

To create a constant, use the define() function.

### **Syntax**

define(*name*, *value*, *case-insensitive*)

<?php  
define("GREETING", "Welcome to W3Schools.com!");  
echo GREETING;  
?>

## PHP Constant Arrays

In PHP7, you can create an Array constant using the define() function.

### **Example**

Create an Array constant:

<?php  
define("cars", [  
  "Alfa Romeo",  
  "BMW",  
  "Toyota"  
]);  
echo cars[0];  
?>

* sort() - sort arrays in ascending order
* rsort() - sort arrays in descending order
* asort() - sort associative arrays in ascending order, according to the value
* ksort() - sort associative arrays in ascending order, according to the key
* arsort() - sort associative arrays in descending order, according to the value
* krsort() - sort associative arrays in descending order, according to the key

## PHP $GLOBALS

$GLOBALS is a PHP super global variable which is used to access global variables from anywhere in the PHP script (also from within functions or methods).

PHP stores all global variables in an array called $GLOBALS[*index*]. The index holds the name of the variable.

The example below shows how to use the super global variable $GLOBALS:

### **Example**

<?php  
$x = 75;  
$y = 25;  
   
function addition() {  
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];  
}  
   
addition();  
echo $z;  
?>

$\_SERVER['PHP\_SELF']

=>Returns the filename of the currently executing script

$\_SERVER['GATEWAY\_INTERFACE']

=>Returns the version of the Common Gateway Interface (CGI) the server is using

$\_SERVER['SERVER\_ADDR']

=>  
Returns the IP address of the host server

$\_SERVER['SERVER\_NAME']

=>Returns the name of the host server (such as [www.w3schools.com](http://www.w3schools.com))

$\_SERVER['SERVER\_SOFTWARE']

=>Returns the server identification string (such as Apache/2.2.24)

$\_SERVER['SERVER\_PROTOCOL']

=>Returns the name and revision of the information protocol

$\_SERVER['REQUEST\_METHOD']

=>Returns the request method used to access the page (such as POST)

$\_SERVER['REQUEST\_TIME']

=>Returns the timestamp of the start of the request (such as 1377687496)

$\_SERVER['QUERY\_STRING']

=>Returns the query string if the page is accessed via a query string

$\_SERVER['HTTP\_ACCEPT']

=>  
Returns the Accept header from the current request

$\_SERVER['HTTP\_ACCEPT\_CHARSET']

=>Returns the Accept\_Charset header from the current request (such as utf-8,ISO-8859-1)

$\_SERVER['HTTP\_HOST']

=>  
Returns the Host header from the current request

$\_SERVER['HTTP\_REFERER']

=>Returns the complete URL of the current page (not reliable because not all user-agents support it)

$\_SERVER['HTTPS']

=>Is the script queried through a secure HTTP protocol

$\_SERVER['REMOTE\_ADDR']

=>Returns the IP address from where the user is viewing the current page

$\_SERVER['REMOTE\_HOST']

=>Returns the Host name from where the user is viewing the current page

$\_SERVER['REMOTE\_PORT']

=>Returns the port being used on the user's machine to communicate with the web server

$\_SERVER['SCRIPT\_FILENAME']

=>Returns the absolute pathname of the currently executing script

$\_SERVER['SERVER\_ADMIN']

=>Returns the value given to the SERVER\_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as [someone@w3schools.com](mailto:someone@w3schools.com))

$\_SERVER['SERVER\_PORT']

=>Returns the port on the server machine being used by the web server for communication (such as 80)

$\_SERVER['SERVER\_SIGNATURE']

=>Returns the server version and virtual host name which are added to server-generated pages

$\_SERVER['PATH\_TRANSLATED']

=>  
Returns the file system based path to the current script

$\_SERVER['SCRIPT\_NAME']

=>Returns the path of the current script

$\_SERVER['SCRIPT\_URI']

=>Returns the URI of the current page

## PHP $\_REQUEST

PHP $\_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $\_REQUEST to collect the value of the input field:

### **Example**

<html>  
<body>  
  
<form method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">  
  Name: <input type="text" name="fname">  
  <input type="submit">  
</form>  
  
<?php  
if ($\_SERVER["REQUEST\_METHOD"] == "POST") {  
  // collect value of input field  
  $name = $\_REQUEST['fname'];  
  if (empty($name)) {  
    echo "Name is empty";  
  } else {  
    echo $name;  
  }  
}  
?>  
  
</body>  
</html>

## PHP $\_POST

PHP $\_POST is a PHP super global variable which is used to collect form data after submitting an HTML form with method="post". $\_POST is also widely used to pass variables.

The example below shows a form with an input field and a submit button. When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag. In this example, we point to the file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $\_POST to collect the value of the input field:

### **Example**

<html>  
<body>  
  
<form method="post" action="<?php echo $\_SERVER['PHP\_SELF'];?>">  
  Name: <input type="text" name="fname">  
  <input type="submit">  
</form>  
  
<?php  
if ($\_SERVER["REQUEST\_METHOD"] == "POST") {  
  // collect value of input field  
  $name = $\_POST['fname'];  
  if (empty($name)) {  
    echo "Name is empty";  
  } else {  
    echo $name;  
  }  
}  
?>  
  
</body>  
</html>

## PHP $\_GET

PHP $\_GET is a PHP super global variable which is used to collect form data after submitting an HTML form with method="get".

$\_GET can also collect data sent in the URL.

### **Example**

<html>  
<body>  
  
<?php  
echo "Study " . $\_GET['subject'] . " at " . $\_GET['web'];  
?>  
  
</body>  
</html>

## 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.

## 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**

* 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?**

* 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.

<?php

//define a function that displays hello function

function add\_numbers(){

echo 1 + 2;

}

add\_numbers ();

?>

## What is Form?

When you login into a website or into your mail box, you are interacting with a form.

Forms are used to get input from the user and submit it to the web server for processing.

The diagram below illustrates the form handling process.

A form is an HTML tag that contains graphical user interface items such as input box, check boxes radio buttons etc.

The form is defined using the <form>…</form> tags and GUI items are defined using form elements such as input.

## Create a form

We will use HTML tags to create a form. Below is the minimal list of things you need to create a form.

* Opening and closing form tags <form>…</form>
* Form submission type POST or GET
* Submission URL that will process the submitted data
* Input fields such as input boxes, text areas, buttons,checkboxes etc.

**The code below creates a simple registration form**

<html>

<head>

<title>Registration Form</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<h2>Registration Form</h2>

<form action="registration\_form.php" method="POST"> First name:

<input type="text" name="firstname"> <br> Last name:

<input type="text" name="lastname">

<input type="hidden" name="form\_submitted" value="1" />

<input type="submit" value="Submit">

</form>

</body>

</html>

## GET vs POST Methods

**POST:-**

* Values not visible in the URL
* Has not limitation of the length of the values since they are submitted via the body of HTTP
* Has lower performance compared to Php\_GET method due to time spent encapsulation the Php\_POST values in the HTTP body
* Supports many different data types such as string, numeric, binary etc.
* Results cannot be book marked

**GET:-**

* Values visible in the URL
* Has limitation on the length of the values usually 255 characters. This is because the values are displayed in the URL. Note the upper limit of the characters is dependent on the browser.
* Has high performance compared to POST method dues to the simple nature of appending the values in the URL.
* Supports only string data types because the values are displayed in the URL
* Results can be book marked due to the visibility of the values in the URL

## Processing the registration form data

The registration form submits data to itself as specified in the action attribute of the form.

When a form has been submitted, the values are populated in the $\_POST super global array.

We will use the PHP isset function to check if the form values have been filled in the $\_POST array and process the data.

We will modify the registration form to include the PHP code that processes the data. Below is the modified code

<html>

<head>

<title>Registration Form</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

</head>

<body>

<?php if (isset($\_POST['form\_submitted'])): ?> //this code is executed when the form is submitted

<h2>Thank You <?php echo $\_POST['firstname']; ?> </h2>

<p>You have been registered as

<?php echo $\_POST['firstname'] . ' ' . $\_POST['lastname']; ?>

</p>

<p>Go <a href="/registration\_form.php">back</a> to the form</p>

<?php else: ?>

<h2>Registration Form</h2>

<form action="registration\_form.php" method="POST">

First name:

<input type="text" name="firstname">

<br> Last name:

<input type="text" name="lastname">

<input type="hidden" name="form\_submitted" value="1" />

<input type="submit" value="Submit">

</form>

<?php endif; ? >

</body>

</html>

HERE,

* <?php if (isset($\_POST[‘form\_submitted’])): ?> checks if the form\_submitted hidden field has been filled in the $\_POST[] array and display a thank you and first name message.

If the form\_fobmitted field hasn’t been filled in the $\_POST[] array, the form is displayed.

<https://www.guru99.com/php-forms-handling.html> this link for more example….

## PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

Concate var with string..

### **Example**

<?php  
$txt1 = "Learn PHP";  
$txt2 = "W3Schools.com";  
$x = 5;  
$y = 4;  
  
echo "<h2>" . $txt1 . "</h2>";  
echo "Study PHP at " . $txt2 . "<br>";  
echo $x + $y;  
?>

## 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.

## Why and when to use Cookies?

* Http is a stateless protocol; cookies allow us to track the state of the application using small files stored on the user’s computer.The path were the cookies are stored depends on the browser.Internet Explorer usually stores them in Temporal Internet Files folder.
* Personalizing the user experience – this is achieved by allowing users to select their preferences.The page requested that follow are personalized based on the set preferences in the cookies.
* Tracking the pages visited by a user

## Creating Cookies

Let’s now look at the basic syntax used to create a cookie.

<?php

setcookie(cookie\_name, cookie\_value, [expiry\_time], [cookie\_path], [domain], [secure], [httponly]);

?>

<?php

Setcookie(cookie\_name,cookie\_value,[expiry\_time],[cookie\_path],[domain],[secure],[httponly]);

?>

* Php“setcookie” is the PHP function used to create the cookie.

<?php

setcookie("user\_name", "Guru99", time()+ 60,'/'); // expires after 60 seconds

echo 'the cookie has been set for 60 seconds';

?>

**Output:**

the cookie has been set for 60 seconds

## Retrieving the Cookie value

Create another file named “cookies\_read.php” with the following code.

<?php

print\_r($\_COOKIE); //output the contents of the cookie array variable

?>

**Output:**

Array ( [PHPSESSID] => h5onbf7pctbr0t68adugdp2611 [user\_name] => Guru99 )

## What is a Session?

* A session is a global variable stored on the server.
* Each session is assigned a unique id which is used to retrieve stored values.
* Whenever a session is created, a cookie containing the unique session id is stored on the user’s computer and returned with every request to the server. If the client browser does not support cookies, the unique php session id is displayed in the URL
* Sessions have the capacity to store relatively large data compared to cookies.
* The session values are automatically deleted when the browser is closed. If you want to store the values permanently, then you should store them in the database.
* Just like the $\_COOKIE array variable, session variables are stored in the $\_SESSION array variable. Just like cookies, the session must be started before any HTML tags.

## Creating a Session

In order to create a session, you must first call the PHP session\_start function and then store your values in the $\_SESSION array variable.

Let’s suppose we want to know the number of times that a page has been loaded, we can use a session to do that.

The code below shows how to create and retrieve values from sessions

<?php

session\_start(); //start the PHP\_session function

if(isset($\_SESSION['page\_count']))

{

$\_SESSION['page\_count'] += 1;

}

else

{

$\_SESSION['page\_count'] = 1;

}

echo 'You are visitor number ' . $\_SESSION['page\_count'];

?>

**Output:**

You are visitor number 1

## Destroying Session Variables

The session\_destroy() function is used to destroy the whole Php session variables.

If you want to destroy only a session single item, you use the unset() function.

The code below illustrates how to use both methods.

<?php

session\_destroy(); //destroy entire session

?>

<?php

unset($\_SESSION['product']); //destroy product session item

?>

Session\_destroy removes all the session data including cookies associated with the session.

Unset only frees the individual session variables.

Other data remains intact.

## What is a File?

A file is simply a resource for storing information on a computer.

Files are usually used to store information such as:

* Configuration settings of a program
* Simple data such as contact names against the phone numbers.
* Images, Pictures, Photos, etc.

[PHP File Formats Support](https://www.guru99.com/php-file-processing.html#1)

[PHP file() Function](https://www.guru99.com/php-file-processing.html#2)

[PHP file\_exists() Function](https://www.guru99.com/php-file-processing.html#3)

[PHP fopen() Function](https://www.guru99.com/php-file-processing.html#4)

[PHP fwrite() Function](https://www.guru99.com/php-file-processing.html#5)

[PHP fclose() Function](https://www.guru99.com/php-file-processing.html#6)

[PHP fgets() Function](https://www.guru99.com/php-file-processing.html#7)

[PHP copy() Function](https://www.guru99.com/php-file-processing.html#8)

[Deleting a file](https://www.guru99.com/php-file-processing.html#9)

[PHP file\_get\_contents() Function](https://www.guru99.com/php-file-processing.html#10)

## PHP File Formats Support

PHP file functions support a wide range of file formats that include:

* File.txt
* File.log
* File.custom\_extension i.e. file.xyz
* File.csv
* File.gif, file.jpg etc
* Files provide a permanent cost effective data storage solution for simple data compared to databases that require other software and skills to manage DBMS systems.
* You want to store simple data such as server logs for later retrieval and analysis
* You want to store program settings i.e. program.ini

## PHP file\_exists() Function

This function is used to determine whether a file exists or not.

* It comes in handy when we want to know if a file exists or not before processing it.
* You can also use this function when creating a new file and you want to ensure that the file does not already exist on the server.

The file\_exist function has the following syntax.

<?php

file\_exists($filename);

?>

<?php

if (file\_exists('my\_settings.txt'))

{

echo 'file found!';

}

else

{

echo 'my\_settings.txt does not exist';

}

?>

## PHP fopen() Function

The fopen function is used to open files. It has the following syntax

<?php

fopen($file\_name,$mode,$use\_include\_path,$context);

?>

|  |  |
| --- | --- |
| **Mode** | **Description** |
| r | * Read file from beginning. * Returns false if the file doesn’t exist. * Read only |
| r+ | * Read file from beginning * Returns false if the file doesn’t exist. * Read and write |
| w | * Write to file at beginning * truncate file to zero length * If the file doesn’t exist attempt to create it. * Write only |
| w+ | * Write to file at beginning, truncate file to zero length * If the file doesn’t exist attempt to create it. * Read and Write |
| a | * Append to file at end * If the file doesn’t exist attempt to create it. * Write only |
| a+ | * Php append to file at end * If the file doesn’t exist attempt to create it * Read and write |

## PHP fwrite() Function

The fwrite function is used to write files.

It has the following syntax

<?php

fwrite($handle, $string, $length);

?>

HERE,

* “fwrite” is the PHP function for writing to files
* “$handle” is the file pointer resource
* “$string” is the data to be written in the file.
* “$length” is optional, can be used to specify the maximum file length.

## PHP fclose() Function

The fclose() function is used to close a file in php which is already open

It has the following syntax.s

<?php

fclose($handle);

?>