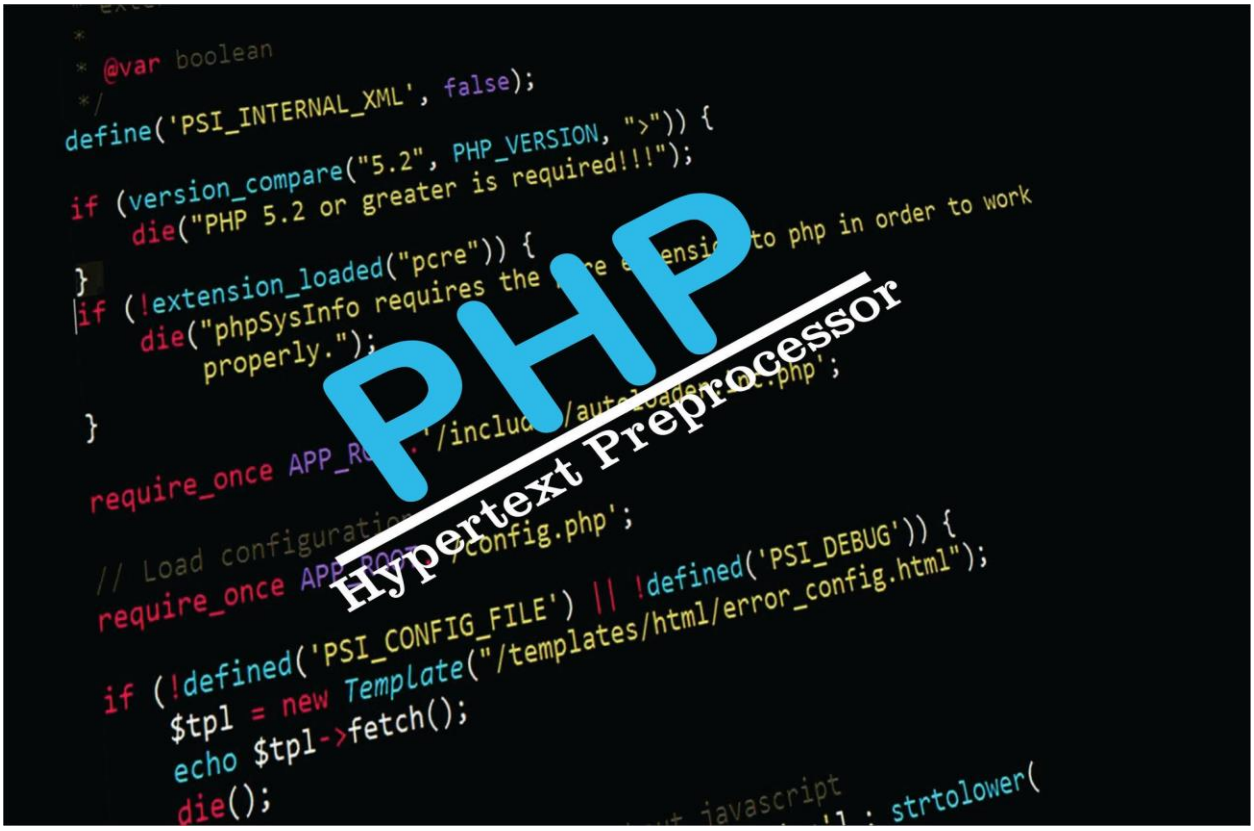


PHP TUTORIAL NOTES

by: SIR MUHAMMAD FARHAN

Contact No: +92316-2859445

Email: mohammadfarhan44500@gmail.com

GitHub: https://github.com/muhammadfarhandevelper

PHP Tutorials

Client Side Scripting Language:

JavaScript, VB Script

Server Side Scripting Language:

PHP, .Net, Python, JAVA, Node JS

What is PHP (Hypertext Preprocessor):

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.

PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

Why to Learn PHP:

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Pre requisite for PHP:

- HTML
- CSS
- JAVASCRIPT (BASIC)

PHP Extension:

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php";

NOTE: All Files Save in **htdocs** Folder that is exist in **Xampp server** folder.

Example:

```
<!DOCTYPE html>
<html>
<body>

<?php

echo “this is Muhammad Farhan website”;

?>

</body>
</html>
```

Echo & Print:

```
echo "Farhan Ali";
print "<br> Farhan Afan";
```

PHP Comment:

```
<!DOCTYPE html>
<html>
<body>

<?php
//This  is_single-linecomment

#  This  is  also  a  single-line  comment

/*
This  is  a  multiple-lines  comment  block
that  spans  over  multiple
lines
*/


?>

</body>
</html>
```

PHP Variables:

In PHP, a variable starts with the $ sign, followed by the name of the variable:

```
<?php

 $name = "Muhammad Farhan";

     ECHO "My Name is $name";
     echo "Ali Raza Khan"
 ?>
```

Strong Types:

A strongly typed language on the contrary wants types specified.

Loosely Types:

In programming we call a language loosely typed when you don't have to explicitly specify types of variables and objects.

PHP Variable Scope:

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:

- global
- local
- static

Global Variable:

A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed outside a function:

Further Global Variable:

The global keyword is used to access a global variable from within a function.

To do this, use the global keyword before the variables (inside the function)

Local Variable:

A variable declared **within** a function has a LOCAL SCOPE and can only be accessed within that function:

Static Variable:

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 you first declare the variable:

Example:

```php
<?php

function show(){
   static $a=5;
   echo $a;
   $a++;

}

show();
show();
show();


?>
```

PHP DATA TYPES:

PHP datatypes:

- String
- Integer
- Float
- Boolean
- Array
- Object (It will be learning in OOP Concept)
- NULL

var_dump();

this function is find the data type of variable.

Example:

```php
<?php
```

```php
    $a = "Farhan";
    $b = 23;
    $c = true;
    $d = 34.55;
    $e = null;
    $f = array("farhan","ali","rehman","afan");

   echo $a , "<br>";
   var_dump($a);
   echo $b , "<br>";
   var_dump($b);
   echo $c, "<br>";
   var_dump($c);
   echo $d , "<br>";
   var_dump($d);
   echo $e , "<br>";
   var_dump($e);
   echo $f . "<br>";
   var_dump($f);


?>
```

Constant Variable in PHP:

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).

Syntax:

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

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

Example:

```
define("var1",50,false);

echo var1;
```

Operators:

Arithmetic Operators:

- +
- -
- *
- /
- %
- **
- ++
- --

Example:

```
$a = 10;
$b = 2;

$c = $a+$b;
$d = $a-$b;
$e = $a*$b;
$f = $a/$b;
$g = $a%$b;
$h = $a**$b;


echo "<br>Variable A  : $a";
echo "<br>Variable B  :   $b";
echo "<br>sum : $c";
echo "<br>sub : $d";
echo "<br>mul : $e";
echo "<br>div : $f";
echo "<br>rem : $g";
echo "<br>exp : $h";
```

Assignment Operator:

Assignment Operator	Without assignment operator	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:

```
$a = 2;
$a+= 2;

echo $a;
```

```
$b = 2;
$b-= 2;

echo $b;

$c = 2;
$c*= 2;

echo $c;

$d = 2;
$d/= 2;

echo $e;
$e = 2;
$e+= 2;

echo $e;
```

Comparison Operator:

The PHP comparison operators are used to compare two values (number or string):

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
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y.

Example:

```
$a = 2;
$b = 5;
```

```
echo "<br>" , $a < $b;
echo "<br>" , $a <= $b;
echo "<br>" , $a > $b;
echo "<br>" , $a >= $b;
echo "<br>" , $a == $b;
echo "<br>" , $a === $b;
echo "<br>" , $a != $b;
echo "<br>" , $a !== $b;
echo "<br>" , $a <=> $b;
```

PHP Increment / Decrement Operator:

++$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:

```
$a =2;

echo ++$a;   //pre increment

$a =2;
```

```
echo $a++;   //post increment
```

Logical Operator:

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:

```
$a =5;
$b = 3;
$c = 1;
```

```
echo ($a > $b) && ($a > $c);
echo ($a > $b) || ($a > $c);
echo !($a > $b) && ($a > $c);
```

PHP 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

PHP Array Operators:

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

PHP LOOPS:

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- 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

PHP While LOOP:

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

Example:

```
$x = 1;

while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
```

```
}
```

PHP Do While LOOP:

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

Example:

```
$x = 1;

do {
  echo "The value is: $x <br>";
  $x++;
} while ($x <= 5);
```

PHP FOR LOOP:

The for loop is used when you know in advance how many times the script should run.

Example:

```
for ($x = 0; $x <= 10; $x++) {
    echo "The Counting is: $x <br>";
  }
```

PHP For Each LOOP:

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

Example:

```
$names = array("ali", "rehman", "afan", "imran");

foreach ($colors as $value) {
  echo "$value <br>";
}
```

PHP Break and Continue:

PHP Break:

The break statement used to jump out of a loop.

Example:

```
for ($x = 0; $x < 10; $x++) {
   if ($x == 4) {
     echo "Counting stoppes";
       break;

   }
   echo "The Counting is: $x <br>";
  }
```

PHP Continue:

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Example:

```
for ($x = 1; $x < 10; $x++) {
   if ($x == 5) {
     continue;
   }
   echo "The counting number is: $x <br>";
 }
```

PHP GOTO:

The **goto** statement is used to send flow of the program to a certain location in the code. The location is specified by a user defined label. Generally, goto statement comes in the script as a part of conditional expression such as if, else or case (in switch construct)

Example:

```
   echo "this is line 1<br>";
   echo "this is line 2<br>";
   echo "this is line 3<br>";
   goto end;
   echo "this is line 4<br>";
   echo "this is line 5<br>";
   end:
   echo "this is line 6<br>";
   echo "this is line 7<br>";
```

PHP Functions:

There are two types of function in any programming.

1. System Defined

2. User Defined

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

PHP User Defined Function:

Besides the built-in PHP functions, it is possible to create your own functions.

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute automatically when a page loads.
- A function will be executed by a call to the function.

Example:

```
//declaration of function and defining of function

function ShowMessage() {
   echo "This is Show Message";
}

ShowMessage();  // call the function
```

Another Example:

```
function ShowMessage($a)
{
   echo "My name is  $a <br>";
}

ShowMessage("Muhammad Farhan");
ShowMessage("Arif Alvi");
ShowMessage("Imran Khan");
```

To specify strict we need to set declare(strict_types=1);. This must be on the very first line of the PHP file.

In the following example we try to send both a number and a string to the function, but here we have added the strict declaration:

Example:

```
<?php declare(strict_types=1);


function addNumbers(int $a, int $b) {
  return $a + $b;
}
echo addNumbers(5, "5 days");

?>
```

Passing Argument by Reference:

Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Example:

```
function myage(&$a)
{
   $a = 45;

   echo "this is my age $a";
}

$a = 20;
myage($a);
echo "<br>bhahar age is $a";
```

Recursive Function:

PHP also supports recursive function call like C/C++. In such case, we call current function within function. It is also known as recursion.

Example:

```
function myage($a)
{
   if($a < 5){
```

```
        echo "value is $a <br>";
        myage($a +1);
      }
   }

   myage(1);
```

PHP ARRAYS:

An array stores multiple values in one single variable:

Example:

```
   $name = array("ali","rehaman",'afaq');

   echo "<pre>";
   print_r($name);
  echo "</pre>";
```

PHP Associative Arrays:

Associative arrays are arrays that use named keys that you assign to them.

It is just like object in JavaScript.

Example:

```
$name = array(
   "farhan" => 45,
   "ali" => 20,
   "rehman" => 30,

);

echo $name["ali"];
```

Example with Loop:

```
$name = array(
```

```
    "farhan" => 45,
    "ali" => 20,
    "rehman" => 30,

);

foreach($name as $x=> $value){
    echo  $x . "=>" .$value;
    echo "<br>";
}
```

Another Example:

```
$name = [
    "farhan" => 45,
    "ali" => 20,
    "rehman" => 30,

];

foreach($name as $value){
    echo  "=>" .$value;
    echo "<br>";
}
```

PHP Multidimensional Arrays:

A multidimensional array is an array containing one or more arrays.

PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

Example:

```
$name = array(
    array(1,"farhan",20,34499),
    array(2,"ali",40,60000),
    array(3,"rehman",10,34000),
    array(4,"farhan",26,12000),
);
```

```
foreach($name as $row){
   foreach($row as $col){
      echo $col;
   }
   echo "<br>";
}
```

Multidimensional Array with Foreach and list():

Example:

```
$name = array(
   array(1,"farhan",20,34499),
   array(2,"ali",40,60000),
   array(3,"rehman",10,34000),
   array(4,"farhan",26,12000),
);


 foreach($name as list($id,$name,$age,$salary)){
       echo "$id $name $age $salary <br>";
 }
```

Another Example:

```
$a = [
  [ "id" => 1, "name" => "farhan", "age" => 23, "salary"=>4500],
  [ "id" => 2, "name" => "afan", "age" => 25, "salary"=>6500],
  ["id" => 3, "name" => "rehman", "age" => 21, "salary"=>3500],
  ["id" => 4, "name" => "ali", "age" => 30, "salary"=>46500]
];


foreach($a as list("id" =>$a,"name" =>$b,"age" =>$c,"salary" =>$d)){
   echo "$a $b $c $d <br>";
}
```

Array Functions:

count array Function:

It is return the number of element in an array.

Example:

```
echo count($age);
```

sizeof array Function:

It is return the number of element in an array.

Example:

```
echo sizeof($age);
```

Sorting Function for Arrays:

sort():

The following example sorts the elements of the array in ascending order:

Example:

```
$age = [100,2,57,90,8];

sort($age);
print_r($age);
```

rsort():

The following example sorts the elements of the array in descending order:

Example:

```
$age = [100,2,57,90,8];

rsort($age);
print_r($age);
```

asort():

The following example sorts the elements of the associate array in ascending order, according to the value

ksort():

The following example sorts the elements of the associate array in ascending order, according to the key

arsort():

The following example sorts the elements of the associate array in descending order, according to the value

krsort():

The following example sorts the elements of the associate array in descending order, according to the key

Inarray():

Search for the value in an array and output 1 or 0:

Example:

```
$array = ["farhan","ali","rehman","afan"];


echo in_array("farhan",$array);
```

array_search():

Search an array for the value and return its key or index:

Example:

```
$array = ["farhan","ali","rehman","afan"];
echo array_search("ali",$array);
```

array_replace():

Replace the values of the first array ($a) with the values from the second array ($b).

Example:

```
$a = ['banana','apple',55,'laptop','mouse'];

$b = ['farhan','afan','bisma'];

$c = array_replace($a,$b);

print_r($c);
```

array_replace_recursive():

Replace the values of the first array with the values from the second array recursively but in two dimensional:

Example:

```
$a = [ [1,2],
     [3,4],  [11,12]];

$b = [ [5,6 ], [7,8]];

print_r(array_replace_recursive($a,$b));
```

array_push():

Insert new element in the end of an array:

Example:

```
$array = ["farhan","ali","rehman","afan"];


array_push($array,"newchild1");

print_r($array);
```

array_pop():

delete element in the end of an array:

Example:

```
$array = ["farhan","ali","rehman","afan"];


array_pop($array);

print_r($array);
```

array_shift():

Delete element in the start of an array:

Example:

```
$array = ["farhan","ali","rehman","afan"];


array_shift($array);
```

```
print_r($array);
```

array_unshift():

Add element in the start of an array:

Example:

```
$array = ["farhan","ali","rehman","afan"];

array_unshift($array,"new name");

print_r($array);
```

array_merge():

Merge two and more arrays into new one array:

```
$array1 = ["farhan","ali","rehman","afan"];
$array2 = ["bisma","ayesha"];


$comarray = array_merge($array1,$array2);

print_r($comarray);
```

array_merge_recursively():

Merge two arrays into one array but it is use in two dimensional:

Example:

```
$a = ['a'=> "banana", 'b' => "apple", 'c'=> 'peach'];
$b = ['b'=> "gauva", 'e' => "orange", 'f'=> 'laddo'];

print_r(array_merge_recursive($a,$b));
```

array_combine():

Create an array by using the elements from one "keys" array and one "values" array:

Example:

```
$array1 = ["farhan","ali","rehman"];
$array2 = [14,20,25];


$comarray = array_combine($array1,$array2);

print_r($comarray);
```

array_slice():

Start the slice from the third array element, and return the rest of the elements in the array:

Example:

```
$a = ['farhan','aysha','husain','hassan'];

$b = array_slice($a,2,2);

print_r($b);
```

array_splice():

Remove elements from an array and replace it with new elements:

Example:

```
$a = ['farhan','aysha','husain','hassan'];
$b = ['bb','dd'];

$c = array_splice($a,0,2,$b);

print_r($a);
```

array_sum():

Return the sum of all the values in the array:

Example:

```
$myarray = array(6,20,21,45);
echo array_sum($myarray);
```

array_product():

Return the product of all the values in the array:

Example:

```
$myarray = array(2,2,2);
echo array_product($myarray);
```

array_rand():

Return an array of random keys:

Example:

```
$myarray = array(6,10,3);

echo array_rand($myarray,1);
```

shuffle():

Randomize the order of the elements in the array:

Example:

```
$myarray = array(6,10,3);
```

```
shuffle($myarray);

print_r($myarray);
```

Array_values:

This function gets the all values from the array.

Array_unique:

This function gets the all unique values from the array.

Array with List():

Assign variables as if they were an array:

Example:

```
$my_array = array("Farhan","Ali","Huzaifa");

list($a, $b, $c) = $my_array;
echo "I have several Good Persons:  sa $a, a $b and a $c";
```

String Search Functions:

String Functions:

Chr():

Returns the character equivalent to the specified ASCII Code.

Example:

```
<?php>
echo chr(65) . "<br>";
?>
```

Strtolower():

Convert the string into lower case.

Example:

```
echo strtolower("FarHan");
```

Strtoupper():

Convert the string into upper case.

Example:

```
echo strtoupper("farhan");
```

Strrev():

Return the reverse of the string.

Example:

```
echo strrev("pakistan");
```

Strlen():

Return the string length.

Example:

```
echo strlen("farhan ali");
```

Str_word_count:

Example:

```
$a = “Muhammad Farhan Ali”;
```

```
echo str_word_count($a);
```

Substr_count():

Example:

```
$a = "farhan ali khan ali";
echo substr_count($a,'ali');
```

Strcmp():

Compare two strings, Return zero if string1 is equal to string2. It return less than zero, if string1 is less than string2. Otherwise, it returns greater than zero, when string1 is greater than string2.

Example:

```
$st1 = "farhan";
$st2 = "farhan";

echo strcmp($st1,$st2);
```

Str_replace():

Replace the characters "farhan" in the string "my name is farhan" with "ali":

Example:

```
$st1 = "my name is farhan";

echo str_replace("farhan","ali",$st1);
```

Trim():

Remove spaces from both sides of a string.

Example:

```
$st1 = "   farhan    ";

echo trim($st1);
```

Math Function:

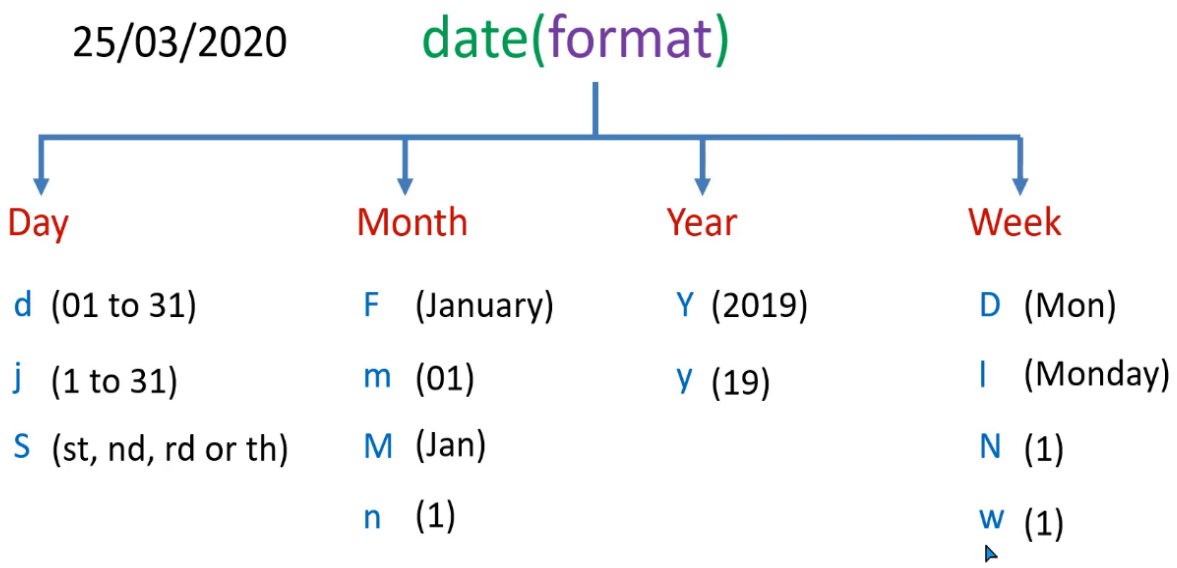
Function	Description
abs()	Returns the absolute (positive) value of a number
ceil()	Rounds a number up to the nearest integer
floor()	Rounds a number down to the nearest integer
max()	Returns the highest value in an array, or the highest value of several specified values
min()	Returns the lowest value in an array, or the lowest value of several specified values

pi()	Returns the value of PI
pow()	Returns x raised to the power of y
rand()	Generates a random integer
round()	Rounds a floating-point number
sqrt()	Returns the square root of a number

Date Functions:

Date():

It is return the current date and time but it is requiring format.

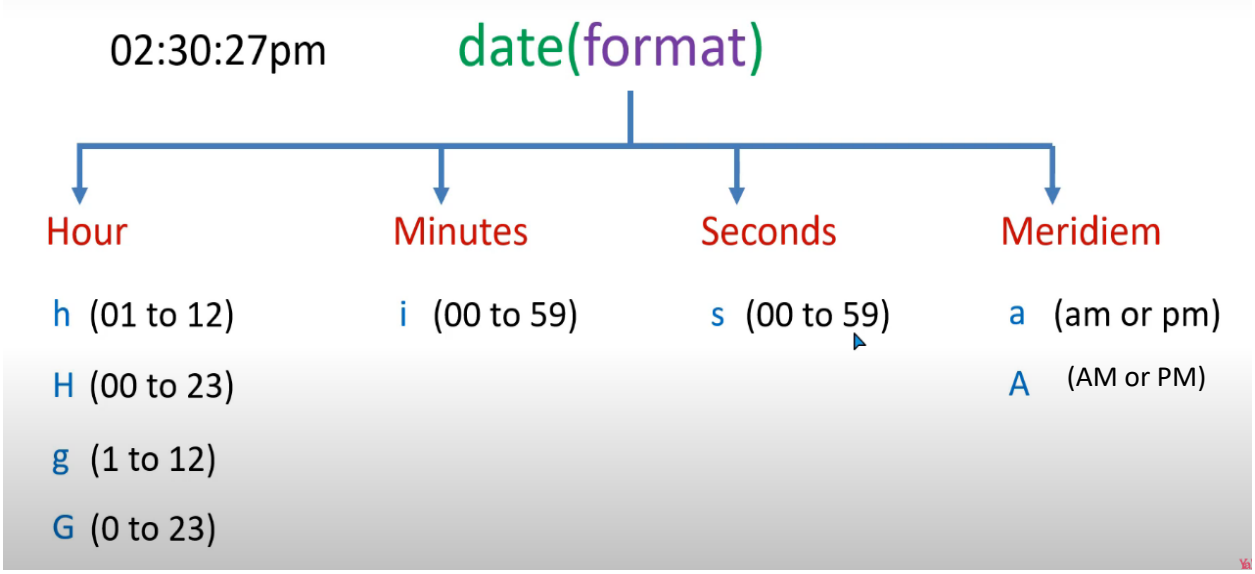


Example:

```
echo date('D   jS-F-Y');
```

Time Function:

It is return the current date and time but it is requiring format.



Example:

```
echo date('H:i:s a');
```

Set Default Time Zone:

It is set the time zone.

Example:

```
date_default_timezone_set("Asia/Kuwait");
```

mktime():

It is a timestamp that is use in date function.

Syntax:

mktime(*hour, minute, second, month, day, year*)

Example:

```
echo date('l',mktime(0,0,0,06,19,2001));
```

Explode:

Using this function Break a string into an array:

Example:

```
$a = "hey this is farhan";

$b = explode(" ",$a);

print_r($b);
```

Implode:

Join array elements with a string.

Example:

```
$a = ['red', 'green', 'yellow', 'brown'];

   $b = implode(" ",$a);

   echo $b;
```

SQL Injection:

SQL injection is a code injection technique that might destroy your database.

SQL injection is one of the most common web hacking techniques.

SQL injection is the placement of malicious code in SQL statements, via web page input.

Addslashes:

Add a backslash in front of each double quote ("):

Example:

```
$a = "hey this ' is' farhan";

echo addslashes($a);
```

StripSlashes:

Example:

```
$a = "hey this ' is' farhan";
echo $a . "<br>";

$b  = addslashes($a);
echo "$b <br>";

echo stripslashes($b);
```

HTML Entities:

This function use for Convert some characters or script into HTML entities.

Htmlentities(string,flags)

Flags: ENT_COMPAT, ENT_QUOTES, ENT_NOQUOTES.

Example:

```
$a = "<a href='#'>This is Link</a>";

echo $a . "<br>";
echo htmlentities($a);
```

html_entity_decode:

Convert HTML entities to characters:

Example:

```
$a = "<a href='#'>This is Link</a>";

echo $a . "<br>";
$b = htmlentities($a);
echo $b . "<br>";

echo html_entity_decode($b);
```

MD5 (message digest algorithm):

Calculate the MD5 hash of the string "farhan".

Md5(string,raw)

Raw: false (default) , true

Example:

```
$a = "farhan12";

echo md5($a,false);
```

Website: https://md5.gromweb.com

Sha1 (US Secure Has Algorithm):

Calculate the SHA-1 hash of the string "Hello":

Example:

```
$a = "farhan12";
```

```
echo sha1($a,false);
```