Unit-4

PHP

Syllabus: Introduction to PHP,Creating PHP script,Running PHP script

Working with Variables and Constants: Using Variables, Using Constants,Data types,Operators,

Controlling Program Flow: Conditional statements,Control statements,Array Functions.

PHP Introduction

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- It is a widely-used, open source scripting language
- PHP scripts are executed on the server
- It is free to download and use
- Websites like www.facebook.com, www.yahoo.com are also built on PHP.
- One of the main reasons behind this is that PHP can be easily embedded in HTML files and HTML codes can also be written in a PHP file.
- The thing that differentiates PHP with client-side language like HTML is, PHP codes are executed on the server whereas HTML codes are directly rendered on the browser.
- PHP codes are first executed on the server and then the result is returned to the browser.
- The only information that the client or browser knows is the result returned after executing the PHP script on the server and not the actual PHP codes present in the PHP file. Also, PHP files can support other client-side scripting languages like CSS and JavaScript.
- Runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Easy to learn and runs efficiently on the server side

Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.

- You add, delete, modify elements within your database through PHP.
- Using PHP, you can restrict users to access some pages of your website.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible −

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

What is a PHP File?

- 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"
- PHP Language is NOT Case Sensitive. For example, print & PRINT mean the same. However, the variables are case sensitive. $num & $Num are treated as different variables.

Creating & Running PHP Script

- PHP is a server-side scripting language. So, to run PHP scripts, you need to install PHP and a web-server software like Apache. For scripting using databases, a DBMS such as MySQL should be installed.
- The most compatible options include PHP + Apache + MySQL
- The PHP files (.php files) should be copied to a particular folder (htdocs or html or www etc.) where the webpages need to be hosted.
- A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Basic PHP Syntax

- A PHP script can be placed anywhere in the document.
- A PHP script starts with <?php and ends with ?>:
- The basic syntax of a PHP code is as follows:

  Syntax:

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

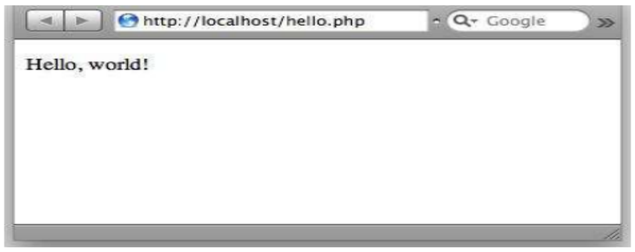
- A PHP file normally contains HTML tags, and some PHP scripting code.

  Example:

```
<HTML>
        <BODY>
                <?php
                        echo "Hello , World";
                ?>
        </BODY>
</HTML>
```

- After copying the files to the server, you can type the ip of the system followed by the php page in the URL bar of the browser . For example, http://192.169.1.112/test.php. To access the PHP page in the same system use, http://localhost/test.php
- Each statement in PHP ends with a semicolon (;).
- To output text or result to the screen in PHP, echo or print can be used. Example, print "Hello";
- Comments in PHP is similar to that of Javascript. For single line,// can be used. For multiline comments, /* and */ can be used.

➢ **Output:**



```
Hello, world!
```

PHP VARIABLES:

- Variable is nothing, it is just the name of the memory location.
- A Variable is simply a container i.e used to store both numeric and non-numeric information.
- Here are the most important things to know about variables in PHP.
  - All variables in PHP are denoted with a leading dollar sign ($).
  - The value of a variable is the value of its most recent assignment.
  - Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
  - PHP variables are Perl-like.
- PHP variables does a good job of automatically converting types from one to another when necessary.

Variable Naming:

Rules for naming a variable is:

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , (,) . & , etc.
- There is no size limit for variables.
- A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).
- Starts with the $ sign, followed by the name of the variable. eg : $x , $cse
- cannot start with a number
- Case-sensitive.

Example :

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

The escape-sequence replacements are −

\n is replaced by the newline character

\t is replaced by the tab character

\$ is replaced by the dollar sign itself ($)

\" is replaced by a single double-quote (")

\\ is replaced by a single backslash (\)

PHP VARIABLE SCOPE

- The scope of a variable is the part of the script where the variable can be referenced/used.
- PHP has three different variable scopes:
    - local
    - global
    - static

LOCAL SCOPE:

- A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function.

```
<?php
        function myTest()
        {
                $x = 5; // local scope
                echo "<p>Variable x inside function is: $x</p>";
        }
?>
```

GLOBAL SCOPE:

- A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function.
- 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)

Example :

```
<?php
        $x = 5;
        $y = 10;
        function myTest()
         {
                $avc=123;
                global $x, $y;
                $y = $x + $y;
        }
        myTest();
        echo $y;     // outputs 15
?>
```

STATIC SCOPE:

- A STATIC variable will not lose its value when the function exits and will still hold that value.
- PHP The **static** Keyword
- 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
        function myTest()
        {
                static $x = 0;
                echo $x;
                $x++;
        }
         myTest();
        myTest();
        myTest();

?>
```

PHP DATA TYPES:

- PHP has a total of eight data types which we use to construct our variables −

1. Integers −

- Integers are whole numbers, without a decimal point, like 4195.
- An integer data type is a non-decimal number between - 2,147,483,648

and 2,147,483,647.

Rules for integers:

- must have at least one digit
- must not have a decimal point
- can be either positive or negative

```
<?php

  $x = 5985;

  var_dump($x);

?>
```

- In the following example $x is an integer.
- The var_dump() function is used to dump information about a variable.
- This function displays structured information such as type and value of the given variable.

2. Doubles −

- Doubles are floating-point numbers, like 3.14159 or 49.1.
- A float (floating point number) is a number with a decimal point or a number in exponential form.

Example :

```
<?php
        $x = 10.365;
        var_dump($x);
?>
```

- In the following example $x is a float.
- The PHP var_dump() function returns the data type and value

3. Booleans –

- Booleans have only two possible values either true or false.

4. NULL –

- Null is a special type that only has one value: NULL.

5. Strings –

- Strings are sequences of characters, like 'PHP supports string operations.'
- A string is a sequence of characters, like "Hello world!".
- A string can be any text inside quotes. You can use single or double quotes:

Example :

```
<?php
        $x = " Hello world! ";
        $y = 'Hello world! ';
        echo $x;
        echo "<br>";
        echo $y;
?>
```

6. Arrays –

- Arrays are named and indexed collections of other values.
- An array stores multiple values in one single variable.

Example :

```
<?php
        $cars = array("Volvo","BMW","Toyota");
        var_dump($cars);
?>
```

- In the following example $cars is an array.
- The PHP var_dump() function returns the data type and value.

7. Objects –

- Objects are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.

8. Resources –

- Resources are special variables that hold references to resources external to PHP (such as database connections).

PHP OPERATORS

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

  •Arithmetic operators

  •Assignment operators

  • Comparison operators

  • Increment/Decrement operators

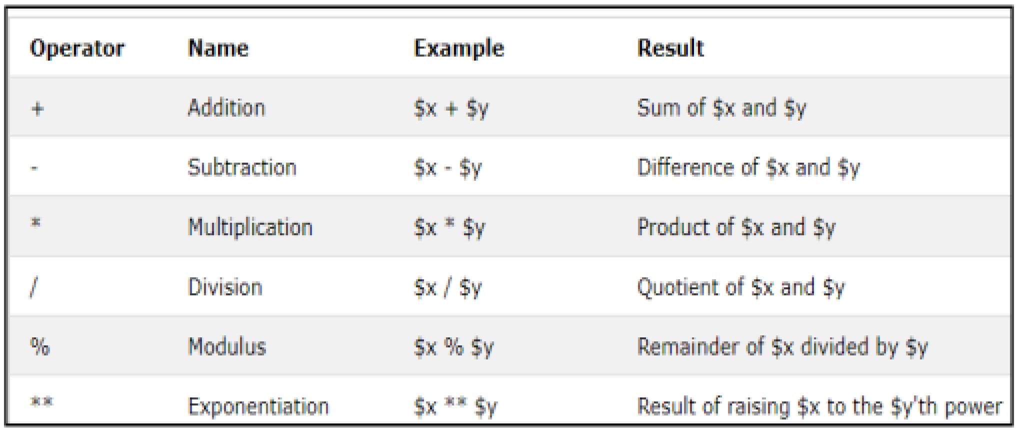
  • Logical operators

  • String operators

  •Array operators

Arithmetic Operators:

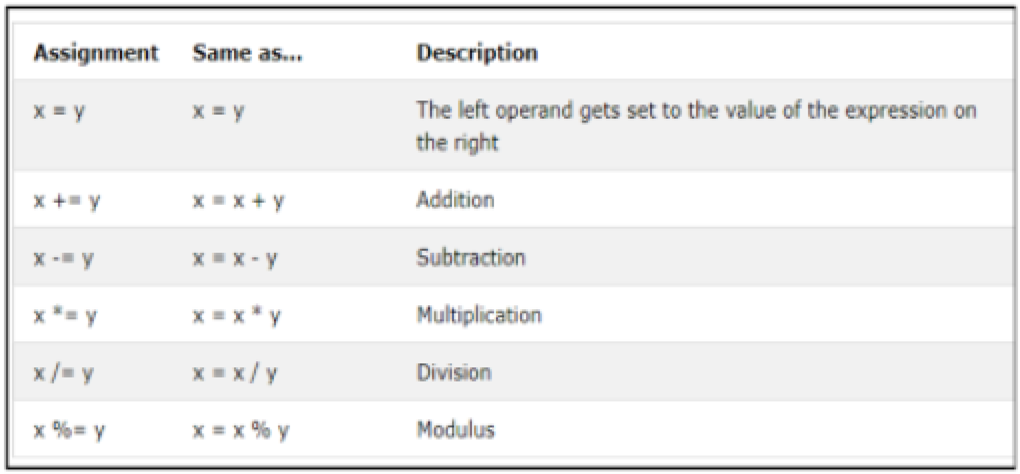
The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.



Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

Assignment Operators:

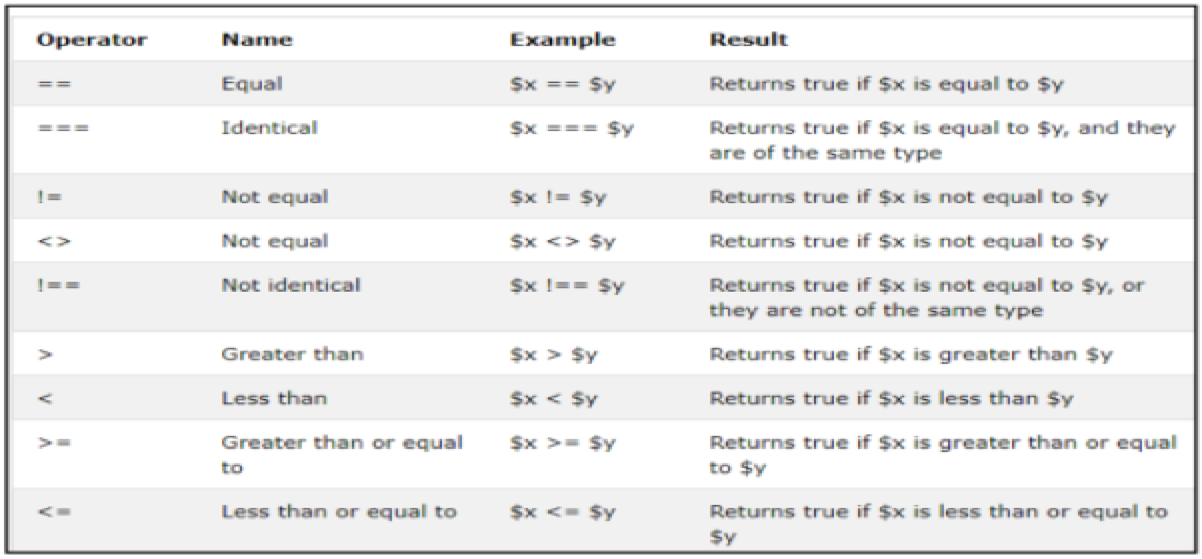
- The PHP assignment operators are used with numeric values to write a value to a variable.
- The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.



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

Comparison Operators:

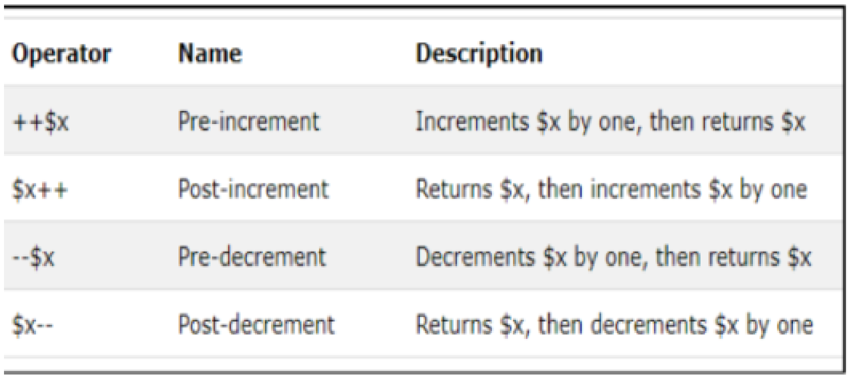
- The PHP comparison operators are used to compare two values (number or string)



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

Increment / Decrement Operators:

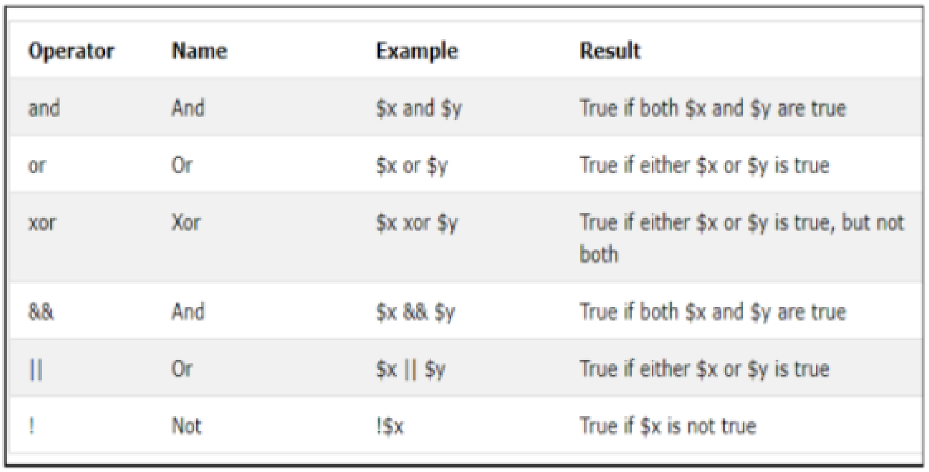
- The PHP increment operators are used to increment a variable's value.
- The PHP decrement operators are used to decrement a variable's value.



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

Logical Operators

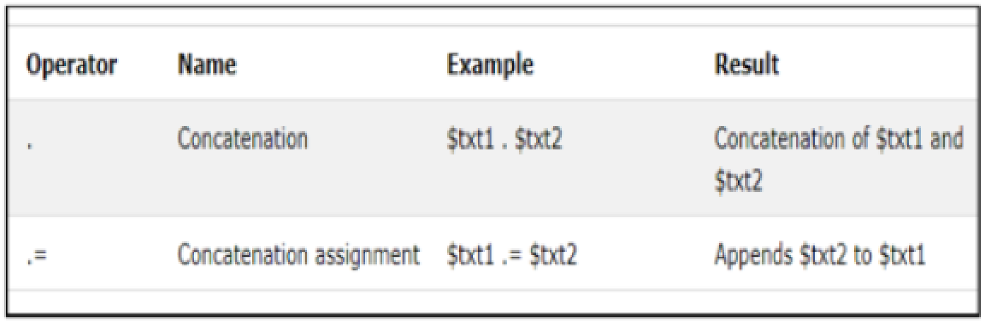
- The PHP logical operators are used to combine conditional statements.



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

String Operators:

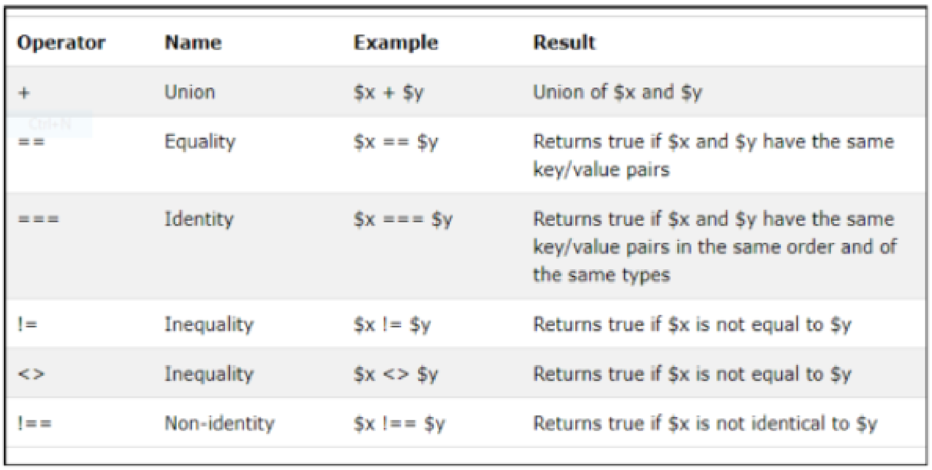
- PHP has two operators that are specially designed for strings



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

Array Operators:

- The PHP array operators are used to compare arrays.



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

Conditional Statements:

- In PHP we have the following conditional statements:
- **if statement -** executes some code if one condition is true
- **if...else statement -** executes some code if a condition is true and another code if that condition is false
- **if...elseif....else statement -** executes different codes for more than two conditions
- **switch statement -** selects one of many blocks of code to be executed.

The if Statement

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

Syntax:

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

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

Example :

```
<?php
        $t = date("H");
        if ($t < "20")
        {
                echo "Have a good day!";
        }
?>
```

The if...else Statement:

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

Syntax

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

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

Example :

```
<?php
        $t = date("H");
        if ($t < "20")
         {
                echo "Have a good day!";
        }
        else
        {
                echo "Have a good night!";
        }
?>
```

The if...elseif....else Statement:

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

  Syntax :

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

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

Example :

```
<?php
        $t = date("H");
        if ($t < "10")
        {
                echo "Have a good morning!";
        }
        elseif ($t < "20")
        {
                echo "Have a good day!";
        }
        else
        {
                echo "Have a good night!";
        }
?>
```

switch Statement:

- Use the switch statement to select one of many blocks of code to be executed.

Syntax :

```
switch (n)
 {
case label1:
        code to be executed if n=label1;
        break;
case label2:
        code to be executed if n=label2;
        break;
case label3:
        code to be executed if n=label3;
        break;
Default:
        code to be executed if n is different from all labels;
}
```

Example :

```
<?php
        $favcolor = "red";
        switch ($favcolor)
         {
        case "red":
                echo "Your favorite color is red!";
                break;
        case "blue":
                echo "Your favorite color is blue!";
                break;
        case "green":
                echo "Your favorite color is green!";
                break;
        default:
                echo "Your favorite color is neither red, blue, nor green!";
        }
?>
```

LOOPING STATEMENTS:

- In PHP, we have the following looping statements:
  - **while -** loops through a block of code as long as the specified condition is true
  - **do...while -** loops through a block of code once, and then repeats the loop as long as the specified condition is true
  - **for -** loops through a block of code a specified number of times
  - **foreach -** loops through a block of code for each element in an array

while Loop:

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

Syntax:

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

Example :

```
<?php
    $x = 1;
    while($x <= 5)
    {
        echo "The number is: $x <br>";
        $x++;
    }
?>
```

do...while Loop:

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

Syntax:

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

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

Example :

```
<?php
        $x = 1;
        do
        {
                echo "The number is: $x <br>";
                $x++;
        } while ($x <= 5);
?>
```

for Loop:

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

Syntax:

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

Parameters:

init counter: Initialize the loop counter value

test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.

increment counter: Increases the loop counter value

- The example below displays the numbers from 0 to 10:

Example :

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

foreach Loop:

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

Syntax:

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

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

Example :

```
<?php
        $colors = array("red", "green", "blue", "yellow");
        foreach ($colors as $value)
        {
                echo "$value <br>“;
        }
?>
```

PHP Break

- You have already seen the break statement used earlier. It was used to "jump out" of a switch statement.
- The break statement can also be used to jump out of a loop.
- This example jumps out of the loop when x is equal to 4

Example :

```
<?php
        for ($x = 0; $x < 10; $x++)
        {
                if ($x == 4)
                {
                        break;
                }
                echo "The number 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.
- This example skips the value of 4:

Example :

```
<?php
        for ($x = 0; $x < 10; $x++)
        {
                if ($x == 4)
                 {
                        continue;
                }
                echo "The number is: $x <br>";
        }
?>
```

PHP ARRAYS

- An array stores multiple values in one single variable

Example :

```
<?php
        $cars = array("Volvo", "BMW", "Toyota");
        echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

What is an Array?

- An array is a special variable, which can hold more than one value at a time.
- An array can hold many values under a single name, and you can access the values by referring to an index number.

Create an Array in PHP

- In PHP, the array() function is used to create an array array();
- In PHP, there are three types of arrays:
  - • Indexed arrays - Arrays with a numeric index
  - • Associative arrays - Arrays with named keys
  - • Multidimensional arrays - Arrays containing one or more arrays

PHP Indexed Arrays

- There are two ways to create indexed arrays:
- The index can be assigned automatically (index always starts at 0), like this:

```
$cars = array("Volvo", "BMW", "Toyota“,”Kia”);
```

Or

the index can be assigned manually:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
$cars[3]=“Kia”
```

- The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values

Example :

```
<?php
        $cars = array("Volvo", "BMW", "Toyota");
        echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

PHP Associative Arrays

- Associative arrays are arrays that use named keys that you assign to them.
- There are two ways to create an associative array:

```
        $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
                        or
        $age['Peter'] = "35";
        $age['Ben'] = "37";
        $age['Joe'] = "43";
```

- The named keys can then be used in a script

Example :

```
<?php
        $age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
        echo "Peter is " . $age['Peter'] . " years old.";
?>
```

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.
- The dimension of an array indicates the number of indices you need to select an element.
- For a two-dimensional array you need two indices to select an element For a three-dimensional array you need three indices to select an element

PHP - Two-dimensional Arrays

- A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

```
$cars = array (
array("Volvo",22,18),
array("BMW",15,13),
array("Saab",5,2),
array("Land Rover",17,15)
);
```

- Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column.
- To get access to the elements of the $cars array we must point to the two indices (row and column):

Example :

```
<?php
        echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
        echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
        echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
        echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

PHP - Constants

- Constants are like variables except that once they are defined they cannot be changed or undefined.
- 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)
```

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 :

Create a constant with a case-sensitive name:

```
<?php
        define("GREETING", "Welcome to SRKR !");
        echo GREETING;
?>
```

Create a constant with a case-insensitive name:

```
<?php
        define("GREETING", "Welcome to SRKR ", true);
        echo greeting;
?>
```

Example :

Create an Array constant:

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

Functions

- PHP functions are similar to other programming languages.
- A function is a piece of code which takes one or more input in the form of a parameter and does some processing and returns a value.
- You already have seen many functions like fopen() and fread() etc.
- They are built-in functions but PHP gives you option to create your own functions as well.
- There are two parts which should be clear to you
- Creating a PHP Function Calling a PHP Function In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different areas and you just need to call them according to your requirement.

PHP built-in functions

- PHP comes standard with many built-in functions. They give us an easier way to implement and repeat popular tasks throughout a program. A popular example is the echo function.

```
echo "I am built in";
```

PHP User Defined Functions

- Besides the built-in PHP functions, it is possible to create your own functions.
    - will not execute immediately when a page loads.
    - will be executed by a call to the function.
    - You may call a function from anywhere within a page.
    - Usage of PHP functions with arguments and return statements is almost similar to that of javascript functions.

The basic syntax of a PHP code is as follows:

```
<?php
function hello()
{
echo "Hello  Welcome to 3⁄4  WT Class";
echo “hi students”;
}
hello(); // call the function
hello();
?>
```

Function Arguments:

- Information can be passed to functions through arguments. An argument is just like a variable.
- Arguments are specified after the function name, inside the parentheses.
- You can add as many arguments as you want, just separate them with a comma.

Example :

```
<!DOCTYPE html>
<html>
<body>
<?php
function familyName($fname, $year,$age)
{
echo "$fname Born in $year and age is $age <br>";
}
familyName(“Ram","1993“,”XX”);
familyName(“Krishna","1994");
familyName("Bheem","1995");
?>
</body>
</html>
```

Output:

```
Ram Born in 1993
Krishna Born in 1994
Bheem Born in 1995
```

Returning values:

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

```
<!DOCTYPE html>
<html>
<body>
<?php
function sum($x, $y)
 {
$z = $x + $y;
return $z;
}
echo "5 + 10 = " .sum(5,10) . "<br>";
echo "7 + 13 = " .sum(7,13) . "<br>";
echo "2 + 4 = " .sum(2,4);
?>
</body>
</html>
```

Output:

```
5 + 10 = 15
7 + 13 = 20
2 + 4 = 6
```

Passing Arguments by Reference

- In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.
- When a function argument is passed by reference, changes to the argument also change the variable that was passed in. To turn a function argument into a reference, the & operator is used

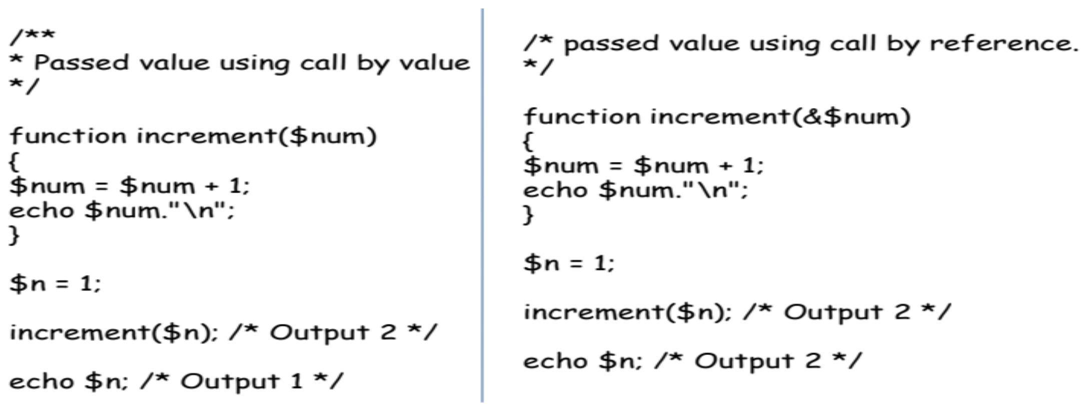
Call by Value in PHP

In call by value, the value of a variable is passed directly. It means,if the value of a variable within the function is changed, it doesn't get changed outside of the function. As we seen in our first example.

Call by Reference in PHP

In call by reference, the address of a variable (their memory location) is passed. In the case of call by reference, we prepend an ampersand (&) to the argument name in the function definition. Any change in variable value within a function can reflect the change in the original value of a variable which we have seen in our second example.

Example :



```
/**
* Passed value using call by value
*/

function increment($num)
{
$num = $num + 1;
echo $num."\n";
}

$n = 1;

increment($n); /* Output 2 */

echo $n; /* Output 1 */
```

```
/* passed value using call by reference.
*/

function increment(&$num)
{
$num = $num + 1;
echo $num."\n";
}

$n = 1;

increment($n); /* Output 2 */

echo $n; /* Output 2 */
```

Strings

- Strings are used for holding characters and text
- A string is a sequence of characters, like "Hello world!".
- **strlen() - Return the Length of a String**

```
<!DOCTYPE html>
        <html>
                <body>
                        <?php
                                echo strlen("Hello world!");
                        ?>
                </body>
        </html>
```

- **str_word_count() - Count Words in a String**

```
<?php
          echo str_word_count("Hello world!");
?>
```

- **strrev() - Reverse a String**

```
<?php
          echo strrev("Hello world!");
?>
```

- **strpos() - Search For a Text Within a String**

The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

Example :

- Search for the text "world" in the string "Hello world!":

```
<?php

          echo strpos("Hello world!", "world"); // outputs 6

?>
```

- **strtolower()-** Converts a string to lowercase Letters
- **strtoupper()-** Converts a string to uppercase letters
- **trim() -** Removes whitespace or other characters from both sides of a string
- **str_replace() -** Replace Text Within a String

  The PHP str_replace() function replaces some characters with some other characters in a string.

Example :

Replace the text "world" with "Everyone":

```
<?php
echo str_replace("world", "Everyone", "Hello world!");  //outputs Hello Everyone
?>
```

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;
?>
```

NUMBERS

PHP Math

- PHP has a set of math functions that allows you to perform mathematical tasks on numbers.
- PHP pi() Function
- The pi() function returns the value of PI:

Example :

```
<?php
echo(pi()); // returns 3.1415926535898
?>
```

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 abs() Function

- The abs() function returns the absolute (positive) value of a number

Example :

```
<?php
        echo(abs(-6.7)); // returns 6.7
?>
```

PHP sqrt() Function

- The sqrt() function returns the square root of a number:

Example :

```
<?php
        echo(sqrt(64)); // returns 8
?>
```

PHP round() Function

- The round() function rounds a floating-point number to its nearest integer

Example :

```
<?php
echo(round(0.60)); // returns 1
echo(round(0.49)); // returns 0
?>
```

Random Numbers

- The rand() function generates a random number

Example :

```
<?php
        echo(rand());
?>
```

- To get more control over the random number, you can add the optional min and max parameters to specify the lowest integer and the highest integer to be returned.
- For example, if you want a random integer between 10 and 100 (inclusive), use rand(10, 100)

Example :

```
<?php
        echo(rand(10, 100));
?>
```

DATE & TIME

- The PHP date() function is used to format a date and/or a time.

The PHP Date() Function

- The PHP date() function formats a timestamp to a more readable date and time.

Syntax:

```
date(format,timestamp)
```

Format - Required. Specifies the format of the

Timestamp - Optional. Specifies a timestamp. Default is the current date and time.

Get a Date

- The required format parameter of the date() function specifies how to format the date (or time).
- Here are some characters that are commonly used for dates:
  - d - Represents the day of the month (01 to 31)
  - m - Represents a month (01 to 12)

    - Y - Represents a year (in four digits)
    - l (lowercase 'L') - Represents the day of the week
- Other characters, like"/", ".", or "-" can also be inserted between the characters to add additional formatting.
- The example below formats today's date in three different ways

Example :

```
<?php
        echo "Today is " . date("Y/m/d") . "<br>";
        echo "Today is " . date("Y.m.d") . "<br>";
        echo "Today is " . date("Y-m-d") . "<br>";
        echo "Today is " . date("l");
?>
```

Get a Time

- Here are some characters that are commonly used for times:
    - H - 24-hour format of an hour (00 to 23)
    - h - 12-hour format of an hour with leading zeros (01 to 12)
    - i - Minutes with leading zeros (00 to 59)
    - s - Seconds with leading zeros (00 to 59)
    - a - Lowercase Ante meridiem and Post meridiem (am or pm)
- The example below outputs the current time in the specified format:

Example :

```
<?php
        echo "The time is " . date("h:i:sa");
?>
```