# What is PHP (open source- free, interpreted- line by line execution)

PHP is a open source, interpreted and object-oriented scripting language i.e. executed at server side. It is used to develop web applications (an application i.e. executed at server side and generates dynamic page).

* PHP stands for HyperText Preprocessor.
* PHP is an interpreted language, i.e. there is no need for compilation.
* PHP is a server side scripting language.
* PHP is faster than other scripting language e.g. asp and jsp.

## PHP Features

There are given many features of PHP.

* **Performance**: Script written in PHP executes much faster then those scripts written in other languages such as JSP & ASP.
* **Open Source Software**: PHP source code is free available on the web, you can developed all the version of PHP according to your requirement without paying any cost.
* **Platform Independent**: PHP are available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.
* **Compatibility**: PHP is compatible with almost all local servers used today like Apache, IIS, WAMP, XAMP etc.
* **Embedded**: PHP code can be easily embedded within HTML tags and script.

Web Development

PHP is widely used in web development now a days. Dynamic websites can be easily developed by PHP. But you must have the basic the knowledge of following technologies for web development as well.

* HTML
* CSS
* JavaScript
* AJAX
* XML and JSON
* JQuery

Prerequisite

Before learning PHP, you must have the basic knowledge of HTML.

# Install PHP

To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:

* **WAMP** for Windows
* **LAMP** for Linux
* **MAMP** for Mac
* **SAMP** for Solaris
* **FAMP** for FreeBSD
* **XAMPP** (Cross, Apache, MySQL, PHP, Perl) for Cross Platform: It includes some other components too such as FileZilla, OpenSSL, Webalizer, OpenSSL, Mercury Mail etc.

If you are on Windows and don't want Perl and other features of XAMPP, you should go for WAMP. In a similar way, you may use LAMP for Linux and MAMP for Macintosh.

Download and Install WAMP Server

[Click me to download WAMP server](http://www.wampserver.com/en/)

Download and Install LAMP Server

[Click me to download LAMP server](http://csg.sph.umich.edu/abecasis/LAMP/download/)

Download and Install MAMP Server

[Click me to download MAMP server](https://www.mamp.info/en/downloads/)

Download and Install XAMPP Server

[Click me to download XAMPP server](https://www.apachefriends.org/download.html)

# PHP Example

It is very easy to create a simple PHP example. To do so, create a file and write HTML tags + PHP code and save this file with .php extension.

All PHP code goes between php tag. A syntax of PHP tag is given below:

1. <?php
2. //your code here
3. ?>

File: first.php

1. <!DOCTYPE>
2. <html>
3. <body>
4. <?php
5. echo "<h2>Hello First PHP</h2>";
6. ?>
7. </body>
8. </html>

# PHP Echo

PHP echo is a language construct not a function, so you don't need to use parenthesis with it. But if you want to use more than one parameters, it is required to use parenthesis.

The syntax of PHP echo is given below:

1. void echo ( string $arg1 [, string $... ] )

PHP echo statement can be used to print string, multi line strings, escaping characters, variable, array etc.

## PHP echo: printing string

File: echo1.php

1. <?php
2. echo "Hello by PHP echo";
3. ?>

## PHP echo: printing multi line string

File: echo2.php

1. <?php
2. echo "Hello by PHP echo
3. this is multi line
4. text printed by
5. PHP echo statement
6. ";
7. ?>

Output:

Hello by PHP echo this is multi line text printed by PHP echo statement

## PHP echo: printing escaping characters

File: echo3.php

1. <?php
2. echo "Hello escape \"sequence\" characters";
3. ?>

Output:

Hello escape "sequence" characters

## PHP echo: printing variable value

File: echo4.php

1. <?php
2. $msg="Hello JavaTpoint PHP";
3. echo "Message is: $msg";
4. ?>

# PHP Print

Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Unlike echo, it always returns 1.

The syntax of PHP print is given below:

1. int print(string $arg)

PHP print statement can be used to print string, multi line strings, escaping characters, variable, array etc.

## PHP print: printing string

File: print1.php

1. <?php
2. print "Hello by PHP print ";
3. print ("Hello by PHP print()");
4. ?>

## PHP print: printing multi line string

File: print2.php

1. <?php
2. print "Hello by PHP print
3. this is multi line
4. text printed by
5. PHP print statement
6. ";
7. ?>

Output:

Hello by PHP print this is multi line text printed by PHP print statement

## PHP print: printing escaping characters

File: print3.php

1. <?php
2. print "Hello escape \"sequence\" characters by PHP print";
3. ?>

Output:

Hello escape "sequence" characters by PHP print

## PHP print: printing variable value

File: print4.php

1. <?php
2. $msg="Hello print() in PHP";
3. print "Message is: $msg";
4. ?>

Output:

Message is: Hello print() in PHP

# PHP $ and $$ Variables

The **$var** (single dollar) is a normal variable with the name var that stores any value like string, integer, float, etc.

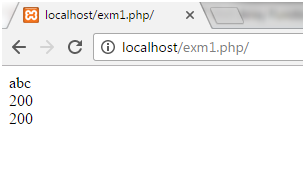
The **$$var** (double dollar) is a reference variable that stores the value of the $variable inside it.

To understand the difference better, let's see some examples.

## Example 1

1. <?php
2. $x = "abc";
3. $$x = 200;
4. echo $x."<br/>";
5. echo $$x."<br/>";
6. echo $abc;
7. ?>

Output:



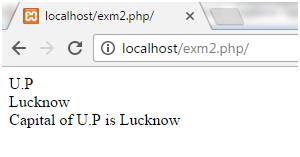
In the above example, we have assigned a value to the variable **x** as **abc**. Value of reference variable **$$x** is assigned as **200**.

Now we have printed the values **$x, $$x** and **$abc**.

## Example2

1. <?php
2. $x="U.P";
3. $$x="Lucknow";
4. echo $x. "<br>";
5. echo $$x. "<br>";
6. echo "Capital of $x is " . $$x;
7. ?>

Output:



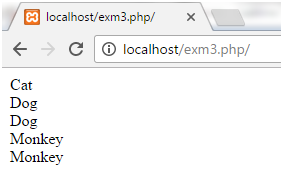
In the above example, we have assigned a value to the variable **x** as **U.P**. Value of reference variable **$$x** is assigned as **Lucknow.**

Now we have printed the values **$x, $$x** and a string.

## Example3

1. <?php
2. $name="Cat";
3. ${$name}="Dog";
4. ${${$name}}="Monkey";
5. echo $name. "<br>";
6. echo ${$name}. "<br>";
7. echo $Cat. "<br>";
8. echo ${${$name}}. "<br>";
9. echo $Dog. "<br>";
10. ?>

Output:



In the above example, we have assigned a value to the variable name **Cat**. Value of reference variable **${$name}** is assigned as **Dog** and **${${$name}}** as **Monkey**.

Now we have printed the values as **$name, ${$name}, $Cat, ${${$name}}** and **$Dog.**

# PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script. PHP constants can be defined by 2 ways:

1. Using define() function
2. Using const keyword

PHP constants follow the same PHP variable rules. For example, it can be started with letter or underscore only.

Conventionally, PHP constants should be defined in uppercase letters.

## PHP constant: define()

Let's see the syntax of define() function in PHP.

1. define(name, value, case-insensitive)
2. name: specifies the constant name
3. value: specifies the constant value
4. case-insensitive: Default value is false. It means it is case sensitive by default.

Let's see the example to define PHP constant using define().

File: constant1.php

1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP");
3. echo MESSAGE;
4. ?>

Hello JavaTpoint PHP

File: constant2.php

1. <?php
2. define("MESSAGE","Hello  PHP",true);//not case sensitive
3. echo MESSAGE;
4. echo message;
5. ?>

Output:

Hello JavaTpoint PHPHello JavaTpoint PHP

File: constant3.php

1. <?php
2. define("MESSAGE","Hello JavaTpoint PHP",false);//case sensitive
3. echo MESSAGE;
4. echo message;
5. ?>

Output:

Hello JavaTpoint PHP

Notice: Use of undefined constant message - assumed 'message'

in C:\wamp\www\vconstant3.php on line 4

message

## PHP constant: const keyword

The const keyword defines constants at compile time. It is a language construct not a function.

It is bit faster than define().

It is always case sensitive.

File: constant4.php

1. <?php
2. const MESSAGE="Hello const by JavaTpoint PHP";
3. echo MESSAGE;
4. ?>

# Magic Constants

Magic constants are the predefined constants in PHP which get changed on the basis of their use. They start with double underscore (\_\_) and ends with double underscore.

They are similar to other predefined constants but as they change their values with the context, they are called magic constants.

There are eight magical constants defined in the below table. They are case-insensitive.

|  |  |
| --- | --- |
| **Name** | **Description** |
| \_\_LINE\_\_ | Represents current line number where it is used. |
| \_\_FILE\_\_ | Represents full path and file name of the file. If it is used inside an include, name of included file is returned. |
| \_\_DIR\_\_ | Represents full directory path of the file. Equivalent to dirname(\_\_file\_\_). It does not have a trailing slash unless it is a root directory. It also resolves symbolic link. |
| \_\_FUNCTION\_\_ | Represents the function name where it is used. If it is used outside of any function, then it will return blank. |
| \_\_CLASS\_\_ | Represents the function name where it is used. If it is used outside of any function, then it will return blank. |
| \_\_TRAIT\_\_ | Represents the trait name where it is used. If it is used outside of any function, then it will return blank. It includes namespace it was declared in. |
| \_\_METHOD\_\_ | Represents the name of the class method where it is used. The method name is returned as it was declared. |
| \_\_NAMESPACE\_\_ | Represents the name of the current namespace. |

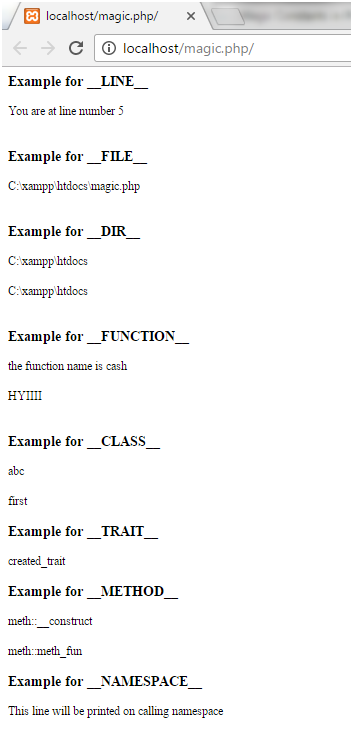
## Example

Let's see an example for each of the above magical constants.

**File Name:** magic.php

1. <?php
2. echo "<h3>Example for \_\_LINE\_\_</h3>";
3. echo "You are at line number " . \_\_LINE\_\_ . "<br><br>";// print Your current line number i.e;3
4. echo "<h3>Example for \_\_FILE\_\_</h3>";
5. echo \_\_FILE\_\_ . "<br><br>";//print full path of file with .php extension
6. echo "<h3>Example for \_\_DIR\_\_</h3>";
7. echo \_\_DIR\_\_ . "<br><br>";//print full path of directory where script will be placed
8. echo dirname(\_\_FILE\_\_) . "<br><br>"; //its output is equivalent to above one.
9. echo "<h3>Example for \_\_FUNCTION\_\_</h3>";
10. //Using magic constant inside function.
11. function cash(){
12. echo 'the function name is '. \_\_FUNCTION\_\_ . "<br><br>";//the function name is cash.
13. }
14. cash();
15. //Using magic constant outside function gives the blank output.
16. function test\_function(){
17. echo 'HYIIII';
18. }
19. test\_function();
20. echo  \_\_FUNCTION\_\_ . "<br><br>";//gives the blank output.
22. echo "<h3>Example for \_\_CLASS\_\_</h3>";
23. class abc
24. {
25. public function \_\_construct() {
26. ;
27. }
28. function abc\_method(){
29. echo \_\_CLASS\_\_ . "<br><br>";//print name of the class abc.
30. }
31. }
32. $t = new abc;
33. $t->abc\_method();
34. class first{
35. function test\_first(){
36. echo \_\_CLASS\_\_;//will always print parent class which is first here.
37. }
38. }
39. class second extends first
40. {
41. public function \_\_construct() {
42. ;
43. }
44. }
45. $t = new second;
46. $t->test\_first();
47. echo "<h3>Example for \_\_TRAIT\_\_</h3>";
48. trait created\_trait{
49. function abc(){
50. echo \_\_TRAIT\_\_;//will print name of the trait created\_trait
51. }
52. }
53. class anew{
54. use created\_trait;
55. }
56. $a = new anew;
57. $a->abc();
58. echo "<h3>Example for \_\_METHOD\_\_</h3>";
59. class meth{
60. public function \_\_construct() {
61. echo \_\_METHOD\_\_ . "<br><br>";//print meth::\_\_construct
62. }
63. public function meth\_fun(){
64. echo \_\_METHOD\_\_;//print meth::meth\_fun
65. }
66. }
67. $a = new meth;
68. $a->meth\_fun();
70. echo "<h3>Example for \_\_NAMESPACE\_\_</h3>";
71. class name{
72. public function \_\_construct() {
73. echo 'This line will be printed on calling namespace';
74. }
75. }
76. $clas\_name= \_\_NAMESPACE\_\_ .'\name';
77. $a = new $clas\_name;
78. ?>

Output:



# PHP Data Types

PHP data types are used to hold different types of data or values. PHP supports 8 primitive data types that can be categorized further in 3 types:

1. Scalar Types
2. Compound Types
3. Special Types

## PHP Data Types: Scalar Types

There are 4 scalar data types in PHP.

1. boolean
2. integer
3. float
4. string

## PHP Data Types: Compound Types

There are 2 compound data types in PHP.

1. array
2. object

## PHP Data Types: Special Types

There are 2 special data types in PHP.

1. resource
2. NULL

# PHP Operators

PHP Operator is a symbol i.e used to perform operations on operands. For example:

1. $num=10+20;//+ is the operator and 10,20 are operands

In the above example, + is the binary + operator, 10 and 20 are operands and $num is variable.

PHP Operators can be categorized in following forms:

* Arithmetic Operators
* Comparison Operators
* Bitwise Operators
* Logical Operators
* String Operators
* Incrementing/Decrementing Operators
* Array Operators
* Type Operators
* Execution Operators
* Error Control Operators
* Assignment Operators

We can also categorize operators on behalf of operands. They can be categorized in 3 forms:

* Unary Operators: works on single operands such as ++, -- etc.
* Binary Operators: works on two operands such as binary +, -, \*, / etc.
* Ternary Operators: works on three operands such as "?:".

## PHP Operators Precedence

Let's see the precedence of PHP operators with associativity.

|  |  |  |
| --- | --- | --- |
| **Operators** | **Additional Information** | **Associativity** |
| clone new | clone and new | non-associative |
| [ | array() | left |
| \*\* | arithmetic | right |
| ++ -- ~ (int) (float) (string) (array) (object) (bool) @ | increment/decrement and types | right |
| Instanceof | types | non-associative |
| ! | logical (negation) | right |
| \* / % | arithmetic | left |
| + - . | arithmetic and string concatenation | left |
| << >> | bitwise (shift) | left |
| < <= > >= | comparison | non-associative |
| == != === !== <> | comparison | non-associative |
| & | bitwise AND | left |
| ^ | bitwise XOR | left |
| | | bitwise OR | left |
| && | logical AND | left |
| || | logical OR | left |
| ?: | ternary | left |
| = += -= \*= \*\*= /= .= %= &= |= ^= <<= >>= => | assignment | right |
| And | logical | left |
| Xor | logical | left |
| Or | logical | left |
| , | many uses (comma) | left |

# PHP Comments

PHP comments can be used to describe any line of code so that other developer can understand the code easily. It can also be used to hide any code.

PHP supports single line and multi line comments. These comments are similar to C/C++ and Perl style (Unix shell style) comments.

## PHP Single Line Comments

There are two ways to use single line comments in PHP.

* // (C++ style single line comment)
* # (Unix Shell style single line comment)

1. <?php
2. // this is C++ style single line comment
3. # this is Unix Shell style single line comment
4. echo "Welcome to PHP single line comments";
5. ?>

Output:

Welcome to PHP single line comments

## PHP Multi Line Comments

In PHP, we can comments multiple lines also. To do so, we need to enclose all lines within /\* \*/. Let's see a simple example of PHP multiple line comment.

1. <?php
2. /\*
3. Anything placed
4. within comment
5. will not be displayed
6. on the browser;
7. \*/
8. echo "Welcome to PHP multi line comment";
9. ?>

# PHP If Else

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

* if
* if-else
* if-else-if
* nested if

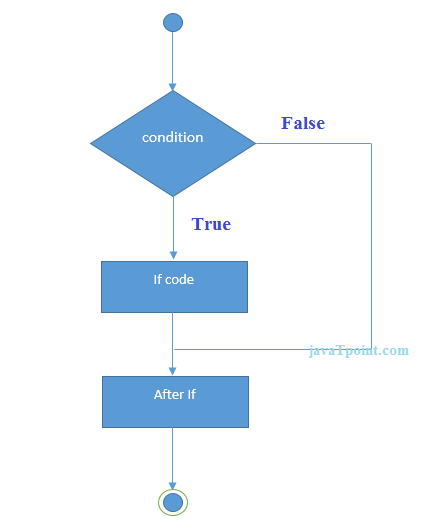
## PHP If Statement

PHP if statement is executed if condition is true.

**Syntax**

1. if(condition){
2. //code to be executed
3. }

**Flowchart**



**Example**

1. <?php
2. $num=12;
3. if($num<100){
4. echo "$num is less than 100";
5. }
6. ?>

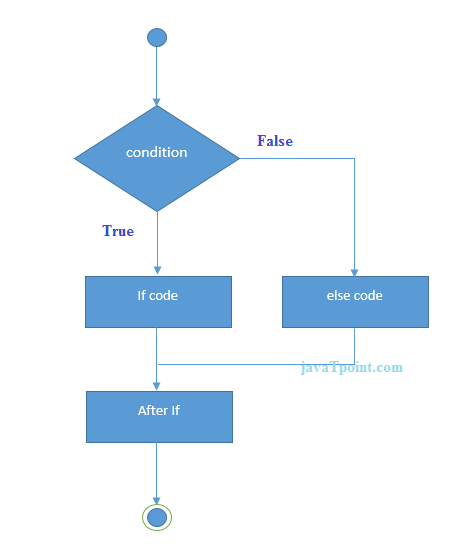
## PHP If-else Statement

PHP if-else statement is executed whether condition is true or false.

**Syntax**

1. if(condition){
2. //code to be executed if true
3. }else{
4. //code to be executed if false
5. }

**Flowchart**



**Example**

1. <?php
2. $num=12;
3. if($num%2==0){
4. echo "$num is even number";
5. }else{
6. echo "$num is odd number";
7. }
8. ?>

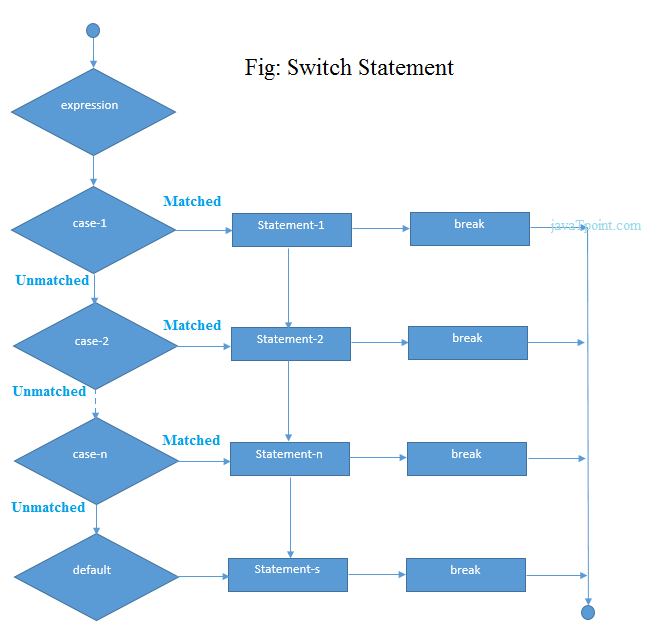
# PHP Switch

PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.

**Syntax**

1. switch(expression){
2. case value1:
3. //code to be executed
4. break;
5. case value2:
6. //code to be executed
7. break;
8. ......
9. default:
10. code to be executed if all cases are not matched;
11. }

**PHP Switch Flowchart**



**PHP Switch Example**

1. <?php
2. $num=20;
3. switch($num){
4. case 10:
5. echo("number is equals to 10");
6. break;
7. case 20:
8. echo("number is equal to 20");
9. break;
10. case 30:
11. echo("number is equal to 30");
12. break;
13. default:
14. echo("number is not equal to 10, 20 or 30");
15. }
16. ?>

# PHP For Loop

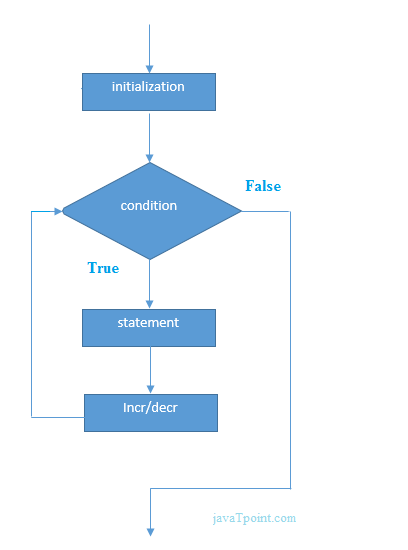
PHP for loop can be used to traverse set of code for the specified number of times.

It should be used if number of iteration is known otherwise use while loop.

**Syntax**

1. for(initialization; condition; increment/decrement){
2. //code to be executed
3. }

**Flowchart**



**Example**

1. <?php
2. for($n=1;$n<=10;$n++){
3. echo "$n<br/>";
4. }
5. ?>

Output:

1

2

3

4

5

6

7

8

9

10

## PHP Nested For Loop

We can use for loop inside for loop in PHP, it is known as nested for loop.

In case of inner or nested for loop, nested for loop is executed fully for one outer for loop. If outer for loop is to be executed for 3 times and inner for loop for 3 times, inner for loop will be executed 9 times (3 times for 1st outer loop, 3 times for 2nd outer loop and 3 times for 3rd outer loop).

**Example**

1. <?php
2. for($i=1;$i<=3;$i++){
3. for($j=1;$j<=3;$j++){
4. echo "$i   $j<br/>";
5. }
6. }
7. ?>

Output:

1 1

1 2

1 3

2 1

2 2

2 3

3 1

3 2

3 3

## PHP For Each Loop

PHP for each loop is used to traverse array elements.

**Syntax**

1. foreach( $array as $var ){
2. //code to be executed
3. }
4. ?>

**Example**

1. <?php
2. $season=array("summer","winter","spring","autumn");
3. foreach( $season as $arr ){
4. echo "Season is: $arr<br />";
5. }
6. ?>

# PHP While Loop

PHP while loop can be used to traverse set of code like for loop.

It should be used if number of iteration is not known.

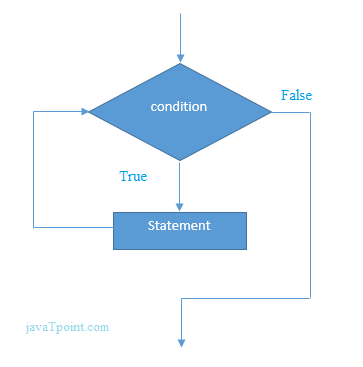
**Syntax**

1. while(condition){
2. //code to be executed
3. }

**Alternative Syntax**

1. while(condition):
2. //code to be executed
4. endwhile;

**PHP While Loop Flowchart**



**PHP While Loop Example**

1. <?php
2. $n=1;
3. while($n<=10){
4. echo "$n<br/>";
5. $n++;
6. }
7. ?>

Output:

1

2

3

4

5

6

7

8

9

10

**Alternative Example**

1. <?php
2. $n=1;
3. while($n<=10):
4. echo "$n<br/>";
5. $n++;
6. endwhile;
7. ?>

Output:

1

2

3

4

5

6

7

8

9

10

## PHP Nested While Loop

We can use while loop inside another while loop in PHP, it is known as nested while loop.

In case of inner or nested while loop, nested while loop is executed fully for one outer while loop. If outer while loop is to be executed for 3 times and nested while loop for 3 times, nested while loop will be executed 9 times (3 times for 1st outer loop, 3 times for 2nd outer loop and 3 times for 3rd outer loop).

**Example**

1. <?php
2. $i=1;
3. while($i<=3){
4. $j=1;
5. while($j<=3){
6. echo "$i   $j<br/>";
7. $j++;
8. }
9. $i++;
10. }
11. ?>

# PHP do while loop

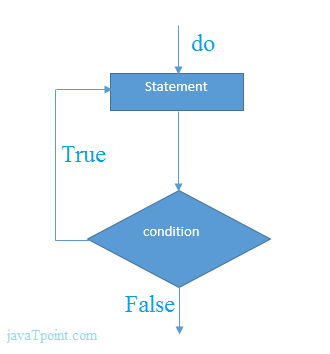
PHP do while loop can be used to traverse set of code like php while loop. The PHP do-while loop is guaranteed to run at least once.

It executes the code at least one time always because condition is checked after executing the code.

**Syntax**

1. do{
2. //code to be executed
3. }while(condition);

**Flowchart**



**Example**

1. <?php
2. $n=1;
3. do{
4. echo "$n<br/>";
5. $n++;
6. }while($n<=10);
7. ?>

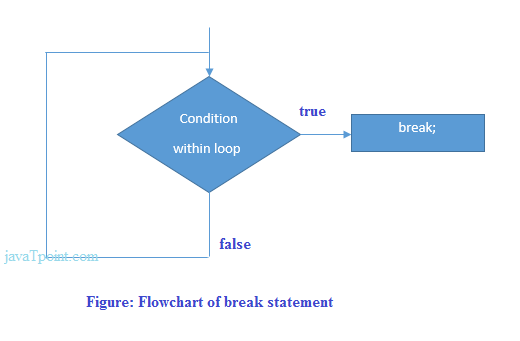
# PHP Break

PHP break statement breaks the execution of current for, while, do-while, switch and for-each loop. If you use break inside inner loop, it breaks the execution of inner loop only.

**Syntax**

1. jump statement;
2. break;

**Flowchart**



## PHP Break: inside loop

Let's see a simple example to break the execution of for loop if value of i is equal to 5.

1. <?php
2. for($i=1;$i<=10;$i++){
3. echo "$i <br/>";
4. if($i==5){
5. break;
6. }
7. }
8. ?>

Output:

1

2

3

4

5

## PHP Break: inside inner loop

The PHP break statement breaks the execution of inner loop only.

1. <?php
2. for($i=1;$i<=3;$i++){
3. for($j=1;$j<=3;$j++){
4. echo "$i   $j<br/>";
5. if($i==2 && $j==2){
6. break;
7. }
8. }
9. }
10. ?>

Output:

1 1

1 2

1 3

2 1

2 2

3 1

3 2

3 3

## PHP Break: inside switch statement

The PHP break statement breaks the flow of switch case also.

1. <?php
2. $num=200;
3. switch($num){
4. case 100:
5. echo("number is equals to 100");
6. break;
7. case 200:
8. echo("number is equal to 200");
9. break;
10. case 50:
11. echo("number is equal to 300");
12. break;
13. default:
14. echo("number is not equal to 100, 200 or 500");
15. }
16. ?>

Questions:

# PHP Programs

PHP programs are frequently asked in the interview. These programs can be asked from basics, control statements, array, string, oops, file handling etc. Let's see the list of top PHP programs.

## 1) Sum of Digits

Write a PHP program to print sum of digits.

**Input:** 23

**Output:** 5

**Input:** 624

**Output:** 12

## 2) Even or odd number

**Input:** 23

**Output:** odd number

**Input:** 12

**Output:** even number

## 3) Prime number

Write a PHP program to check prime number.

**Input:** 17

**Output:** not prime number

**Input:** 57

**Output:** prime number

## 4) Table of number

Write a PHP program to print table of a number.

**Input:** 2

**Output:** 2 4 6 8 10 12 14 16 18 20

**Input:** 5

**Output:** 5 10 15 20 25 30 35 40 45 50

## 5) Factorial

Write a PHP program to print factorial of a number.

**Input:** 5

**Output:** 120

**Input:** 6

**Output:** 720

## 6) Armstrong number

Write a PHP program to check armstrong number.

**Input:** 371

**Output:** armstrong

**Input:** 342

**Output:** not armstrong

## 7) Palindrome number

Write a PHP program to check palindrome number.

**Input:** 121

**Output:** not palindrome number

**Input:** 113

**Output:** palindrome number

## 8) Fibonacci Series

Write a PHP program to print fibonacci series without using recursion and using recursion.

**Input:** 10

**Output:** 0 1 1 2 3 5 8 13 21 34

## 9) Reverse Number

Write a PHP program to reverse given number.

**Input:** 234

**Output:** 432

## 10) Reverse String

Write a PHP program to reverse given string.

**Input:** amit

**Output:** tima

## 11) Swap two numbers

Write a PHP program to swap two numbers with and without using third variable.

**Input:** a=5 b=10

**Output:** a=10 b=5

## 12) Adding Two Numbers

Write a PHP program to add two numbers.

**First Input:** 10

**Second Input:** 20

**Output:** 30

## 13) Subtracting Two Numbers

Write a PHP program to subtract two numbers.

**First Input:** 50

**Second Input:** 10

**Output:** 40

## 14) Area of Triangle

Write a PHP program to find area of triangle.

**Base Input:** 10

**Height Input:** 15

**Output:** 75

## 15) Area of rectangle

Write a PHP program to find the area of rectangle.

**Length Input:** 10

**Width Input:** 20

**Output:** 200

## 16) Leap Year

Write a PHP program to find if the given year is leap year or not.

**Input:** 2000

**Output:** Leap Year

**Input:** 2001

**Output:** Not Leap Year

## 17) Alphabet Triangle using PHP method

Write a PHP program to print alphabet triangle.

**Output:**

A

ABA

ABCBA

ABCDCBA

ABCDEDCBA

## 18) Alphabet Triangle Pattern

Write a PHP program to print alphabet triangle.

**Output:**

A

ABA

ABCBA

ABCDCBA

ABCDEDCBA

## 19) Number Triangle

Write a PHP program to print number triangle.

**Output:**

enter the range= 6

1

121

12321

1234321

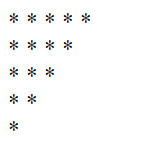
123454321

12345654321

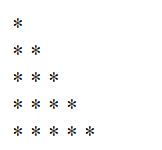
## 20) Star Triangle

Write a PHP programs to print star triangle.

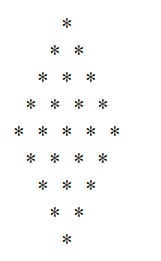
**Output:**



**Output:**



**Output:**



**Output:**



**Output:**



# Sum of Digits

To find sum of digits of a number just add all the digits.

For example,

1. 14597 = 1 + 4 + 5 + 9 + 7
2. 14597 = 26

**Logic:**

* Take the number.
* Divide the number by 10.
* Add the remainder to a variable.
* Repeat the process until remainder is 0.

**Example:**

Given program shows the sum of digits of 14597.

1. <?php
2. $num = 14597;
3. $sum=0; $rem=0;
4. for ($i =0; $i<=strlen($num);$i++)
5. {
6. $rem=$num%10;
7. $sum = $sum + $rem;
8. $num=$num/10;
9. }
10. echo "Sum of digits 14597 is $sum";
11. ?>

Output:

