

ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY

Kanyakumari Main Road, near Anjugramam, Palkulam, Anjugramam, Tamil Nadu 629401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATION COURSE

ON

PHP PROGRAMMING

COURSE MATERIAL

PHP PROGRAMMING

PHP is an open-source, interpreted, and object-oriented scripting language that can be executed at the server-side. PHP is well suited for web development. Therefore, it is used to develop web applications (an application that executes on the server and generates the dynamic page.).

PHP was created by Rasmus Lerdorf in 1994 but appeared in the market in 1995. PHP 7.4.0 is the latest version of PHP, which was released on 28 November. Some important points need to be noticed about PHP are as followed:

- PHP stands for Hypertext Preprocessor.
- PHP is an interpreted language, i.e., there is no need for compilation.
- PHP is faster than other scripting languages, for example, ASP and JSP.
- PHP is a server-side scripting language, which is used to manage the dynamic content of the website.
- PHP can be embedded into HTML.
- PHP is an object-oriented language.
- PHP is an open-source scripting language.
- PHP is simple and easy to learn language.



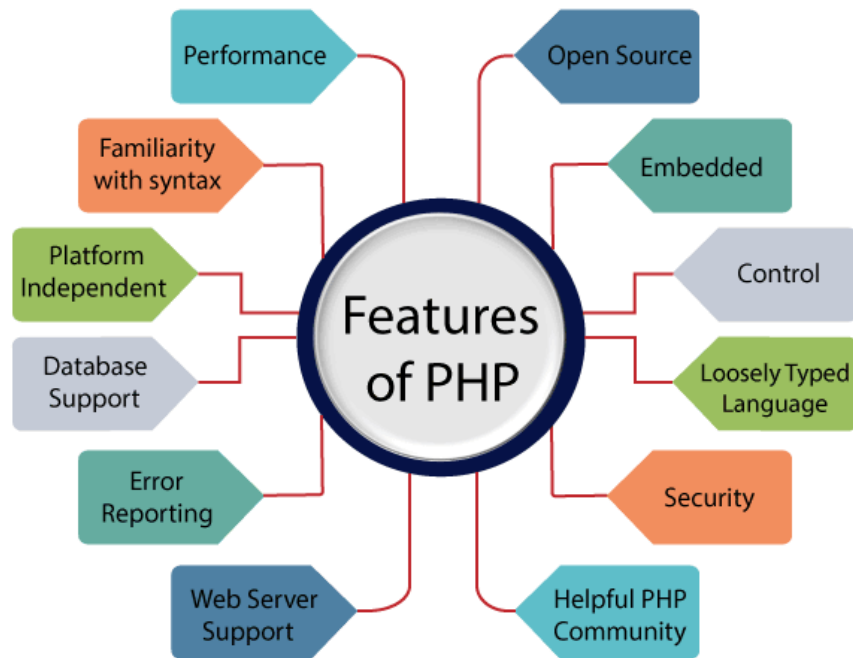
Why use PHP

PHP is a server-side scripting language, which is used to design the dynamic web applications with MySQL database.

- It handles dynamic content, database as well as session tracking for the website.
- You can create sessions in PHP.
- It can access cookies variable and also set cookies.
- It helps to encrypt the data and apply validation.
- PHP supports several protocols such as HTTP, POP3, SNMP, LDAP, IMAP, and many more.
- Using PHP language, you can control the user to access some pages of your website.
- As PHP is easy to install and set up, this is the main reason why PHP is the best language to learn.
- PHP can handle the forms, such as - collect the data from users using forms, save it into the database, and return useful information to the user. For example - Registration form.

PHP Features

PHP is very popular language because of its simplicity and open source. There are some important features of PHP given below:



Performance:

PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.

Open Source:

PHP source code and software are freely available on the web. You can develop all the versions of PHP according to your requirement without paying any cost. All its components are free to download and use.

Familiarity with syntax:

PHP has easily understandable syntax. Programmers are comfortable coding with it.

Embedded:

PHP code can be easily embedded within HTML tags and script.

Platform Independent:

PHP is available for WINDOWS, MAC, LINUX & UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.

Database Support:

PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.

Error Reporting -

PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.

Loosely Typed Language:

PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.

Web servers Support:

PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.**Security:**

PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.

Control:

Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

A Helpful PHP Community:

It has a large community of developers who regularly updates documentation, tutorials, online help, and FAQs. Learning PHP from the communities is one of the significant benefits.

Web Development

PHP is widely used in web development nowadays. PHP can develop dynamic websites easily. 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, CSS, and JavaScript. So, learn these technologies for better implementation of PHP.

HTML - HTML is used to design static webpage.CSS - CSS helps to make the webpage content more effective and attractive.JavaScript - JavaScript is used to design an interactive website. Our PHP tutorial is designed to help beginners and professionals. This PHP tutorial will help those who are unaware about the concepts of PHP but have basic knowledge of computer programming. We assure you that you will not find any problem in this PHP tutorial. But if there is any mistake or error, please post the error in the contact form.

Install PHP

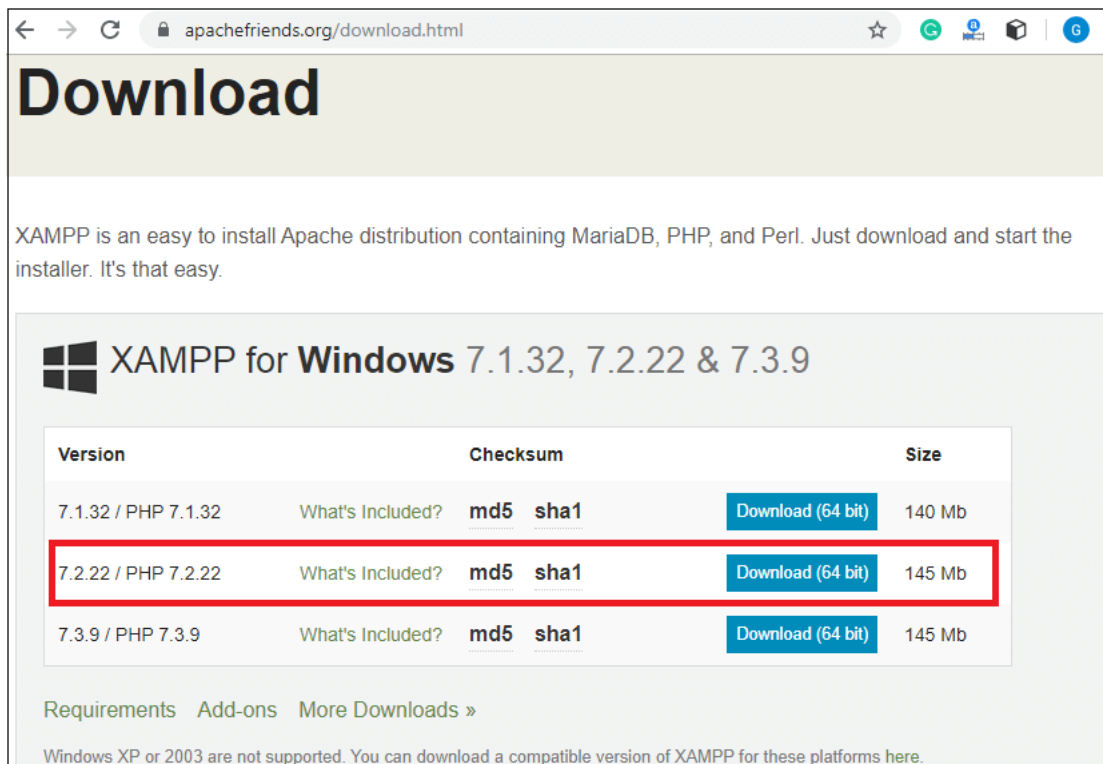
To install PHP, we will suggest you to install AMP (Apache, MySQL, PHP) software stack. It is available for all operating systems. There are many AMP options available in the market that are given below:

- 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, Mercury Mail, etc.

How to install XAMPP server on windows

We will learn how to install the XAMPP server on windows platform step by step. Follow the below steps and install the XAMPP server on your system.

Step 1: Click on the above link provided to download the XAMPP server according to your window requirement.



XAMPP is an easy to install Apache distribution containing MariaDB, PHP, and Perl. Just download and start the installer. It's that easy.

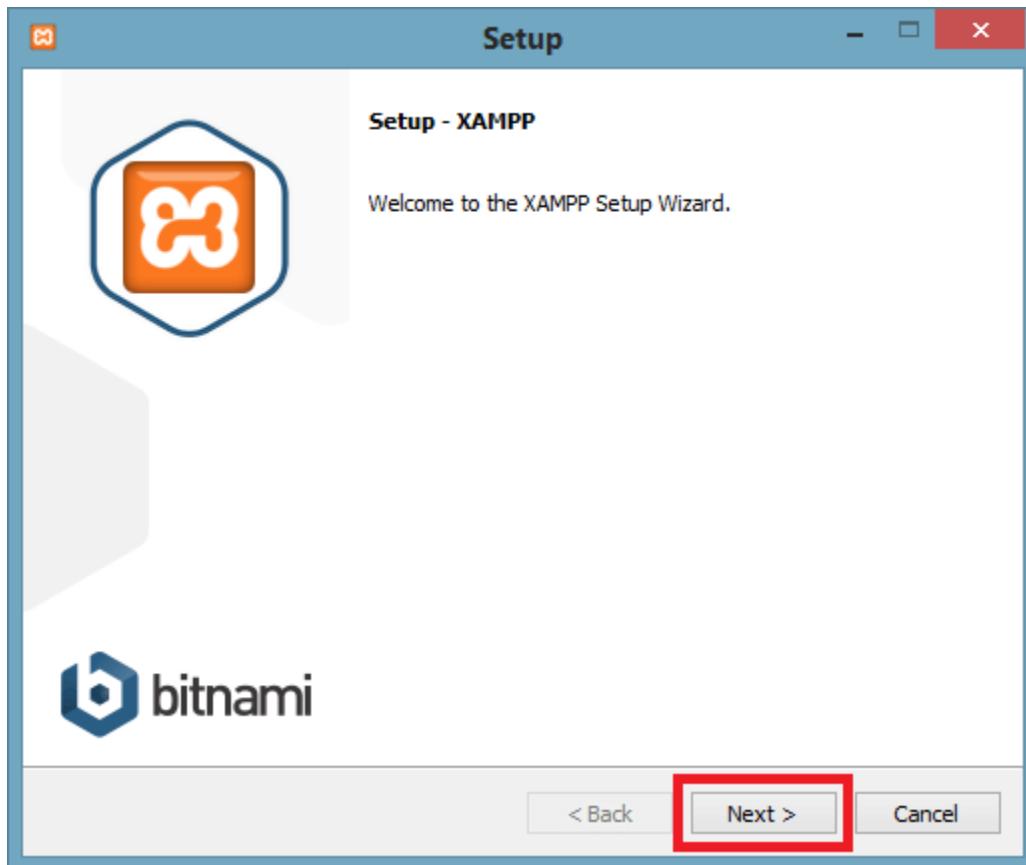
XAMPP for Windows 7.1.32, 7.2.22 & 7.3.9

Version	Checksum	Size
7.1.32 / PHP 7.1.32 What's Included?	md5 sha1	Download (64 bit) 140 Mb
7.2.22 / PHP 7.2.22 What's Included?	md5 sha1	Download (64 bit) 145 Mb
7.3.9 / PHP 7.3.9 What's Included?	md5 sha1	Download (64 bit) 145 Mb

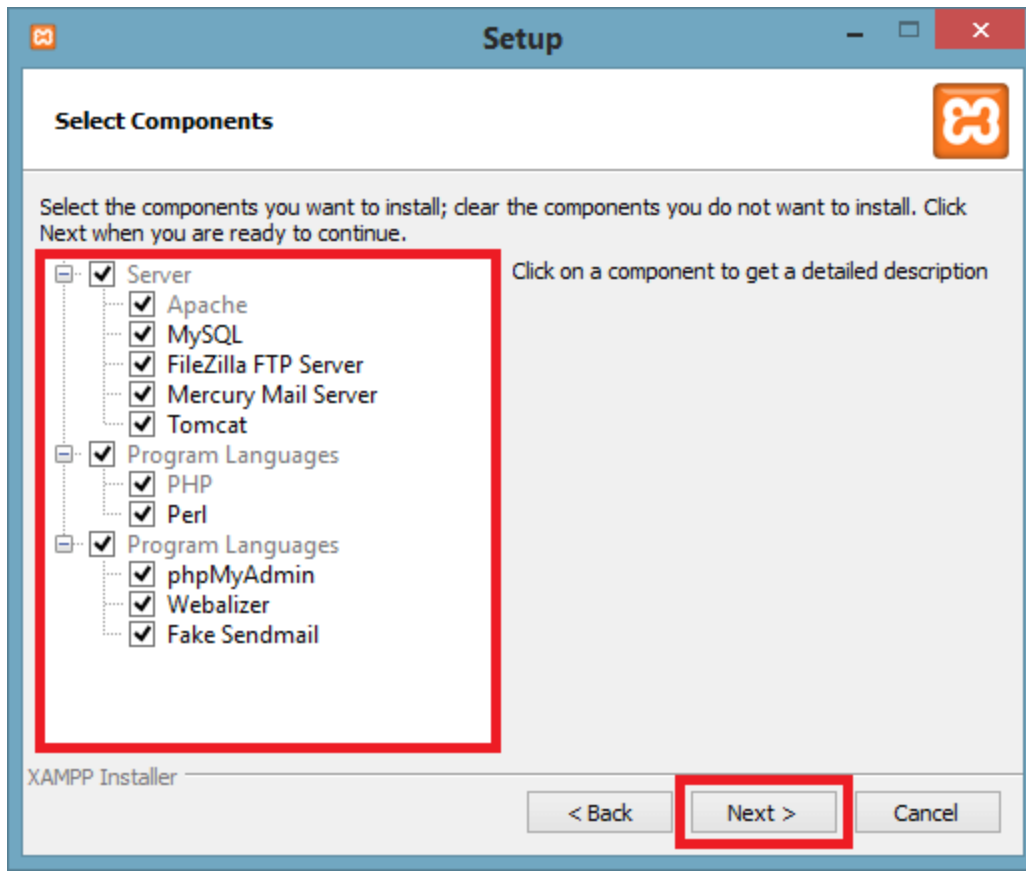
[Requirements](#) [Add-ons](#) [More Downloads »](#)

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms [here](#).

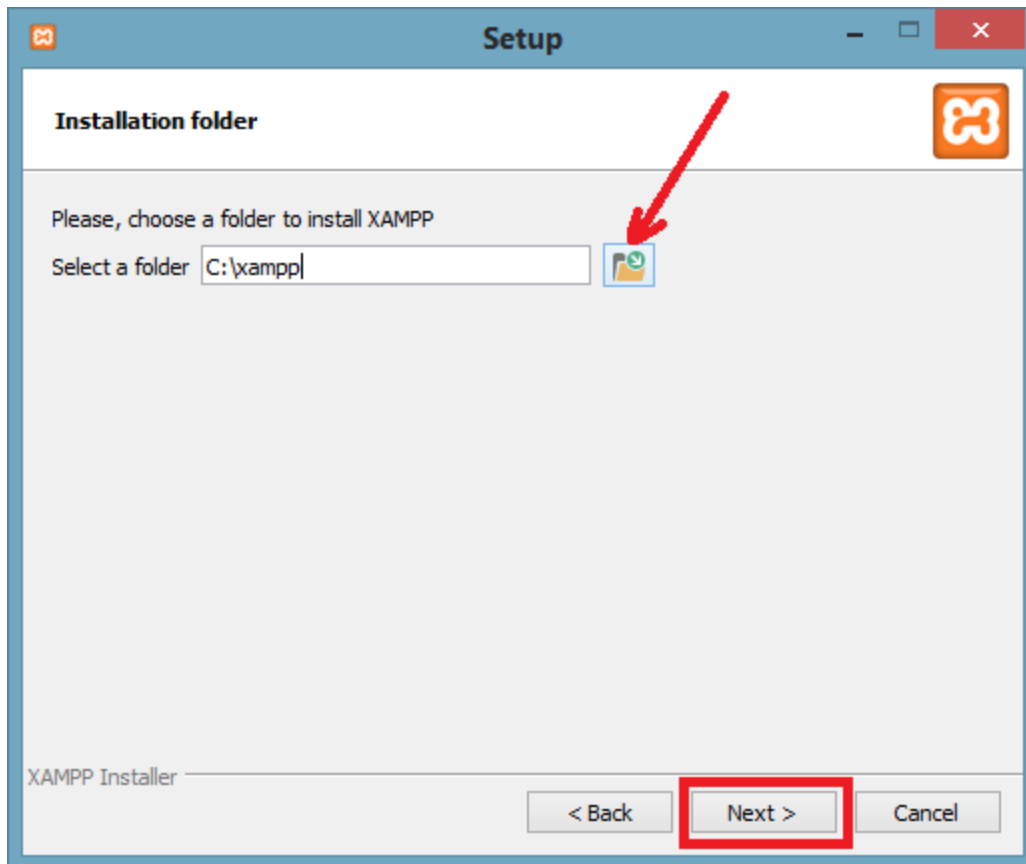
Step 2: After downloading XAMPP, double click on the downloaded file and allow XAMPP to make changes in your system. A window will pop-up, where you have to click on the Next button.



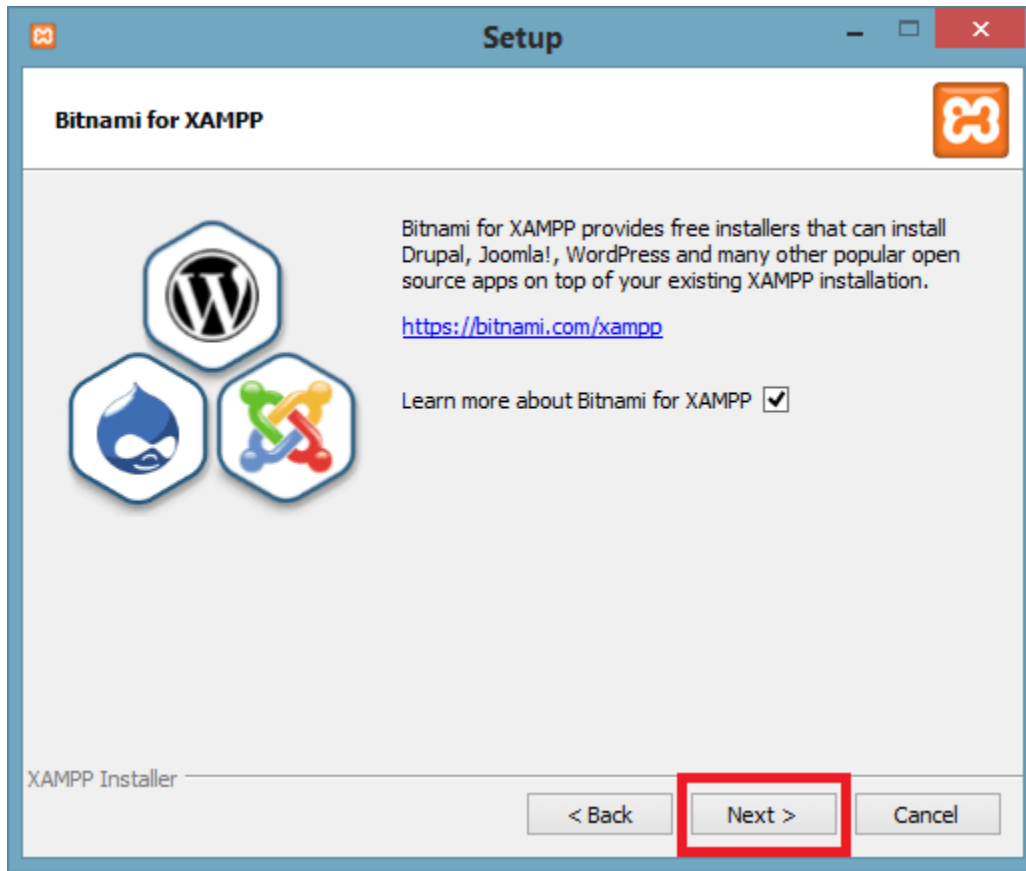
Step 3: Here, select the components, which you want to install and click Next.



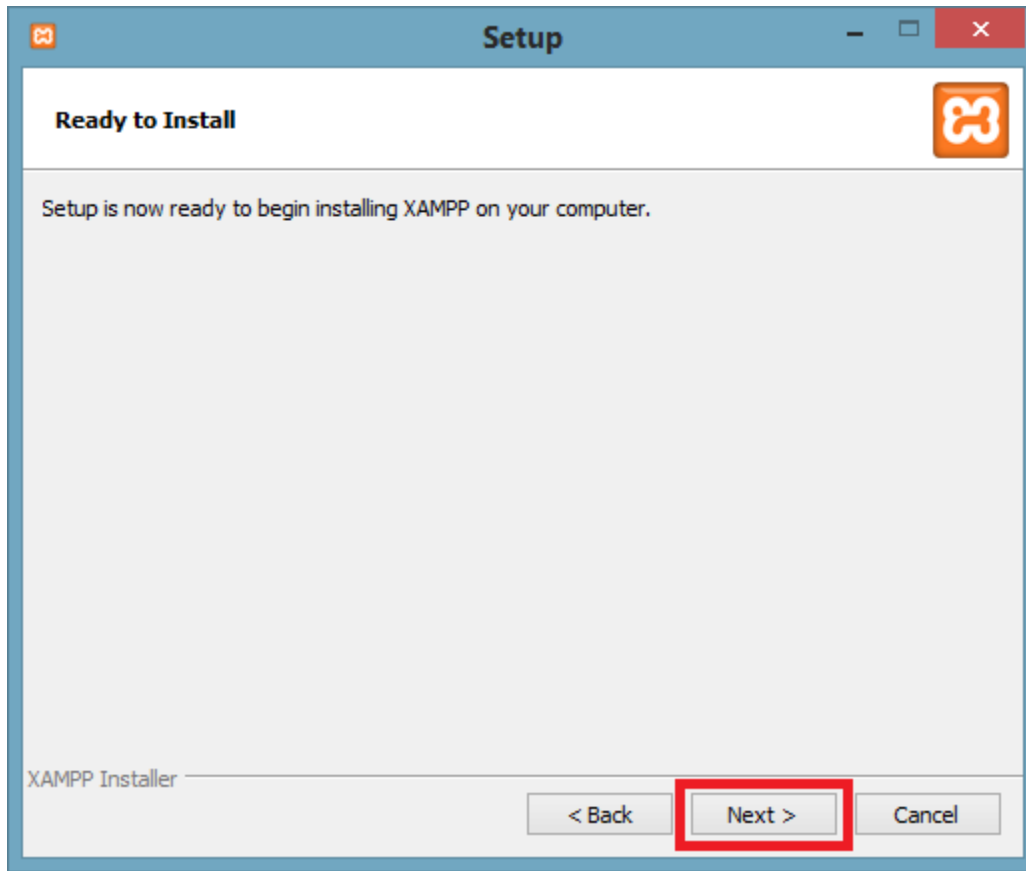
Step 4: Choose a folder where you want to install the XAMPP in your system and click Next.



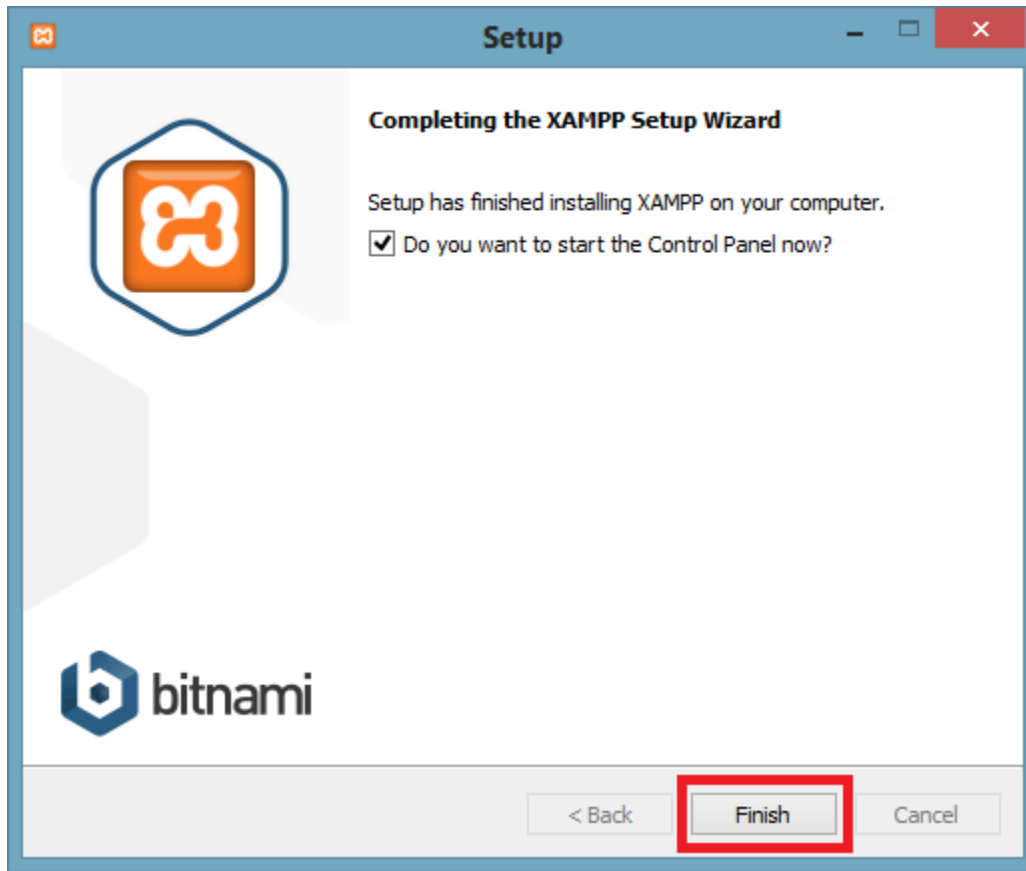
Step 5: Click Next and move ahead.



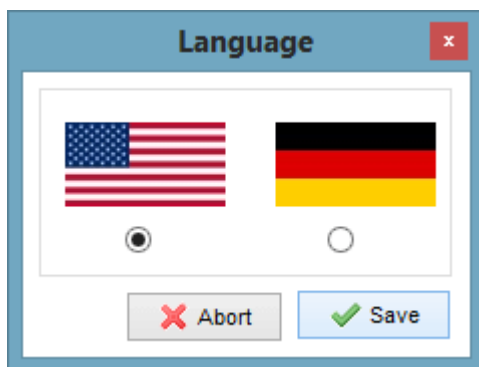
Step 6: XAMPP is ready to install, so click on the Next button and install the XAMPP.



Step 7: A finish window will display after successful installation. Click on the Finish button.

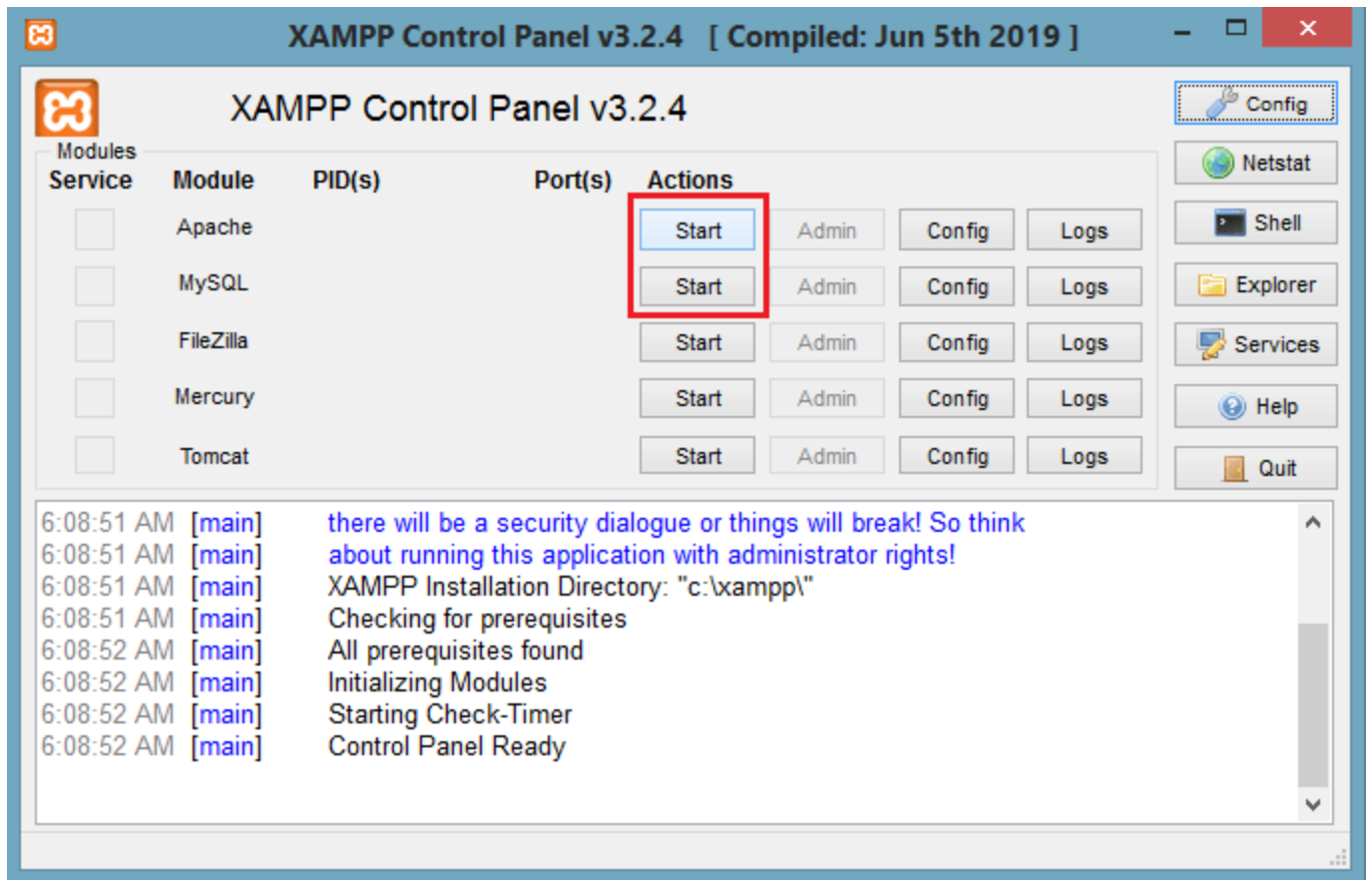


Step 8: Choose your preferred language.

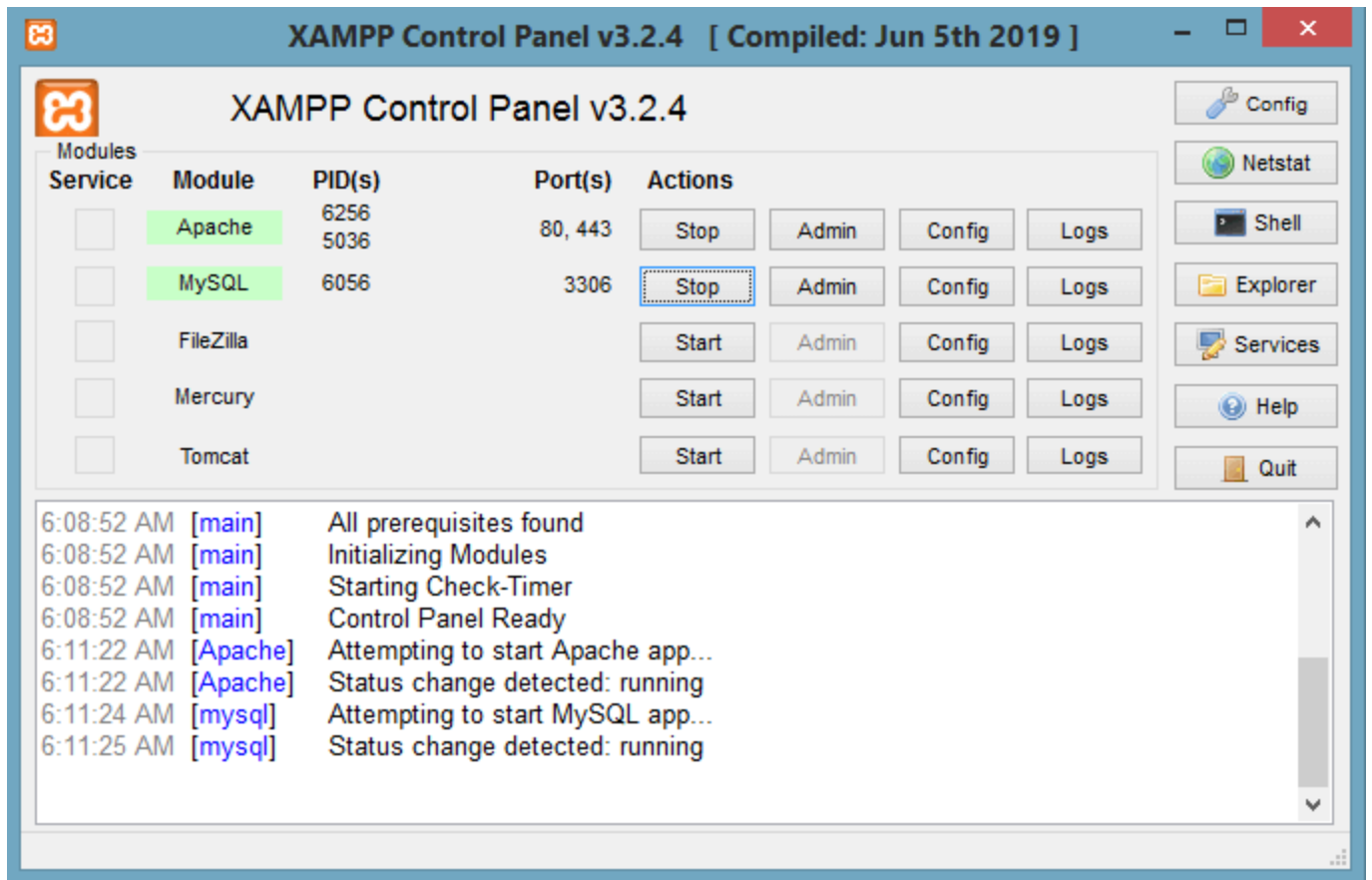


Step 9: XAMPP is ready to use. Start the Apache server and MySQL and run the php program on the localhost.

How to run PHP programs on XAMPP, see in the next tutorial.



Step 10: If no error is shown, then XAMPP is running successfully.



How to run PHP code in XAMPP.

Generally, a PHP file contains HTML tags and some PHP scripting code. 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 the php tag. It starts with `<?php` and ends with `?>`. The syntax of PHP tag is given below:

```
<?php
//your code here
?>
```

Let's see a simple PHP example where we are writing some text using PHP echo command.

File: first.php

```
<!DOCTYPE>
<html>
<body>
<?php
echo "<h2>Hello First PHP</h2>";
?>
</body>
</html>
```

How to run PHP programs in XAMPP

How to run PHP programs in XAMPP PHP is a popular backend programming language. PHP programs can be written on any editor, such as - Notepad, Notepad++, Dreamweaver, etc. These programs save with .php extension, i.e., filename.php inside the htdocs folder.

For example - p1.php.

As I'm using window, and my XAMPP server is installed in D drive. So, the path for the htdocs directory will be "D:\xampp\htdocs".PHP program runs on a web browser such as - Chrome, Internet Explorer, Firefox, etc. Below some steps are given to run the PHP programs.

Step 1: Create a simple PHP program like hello world.

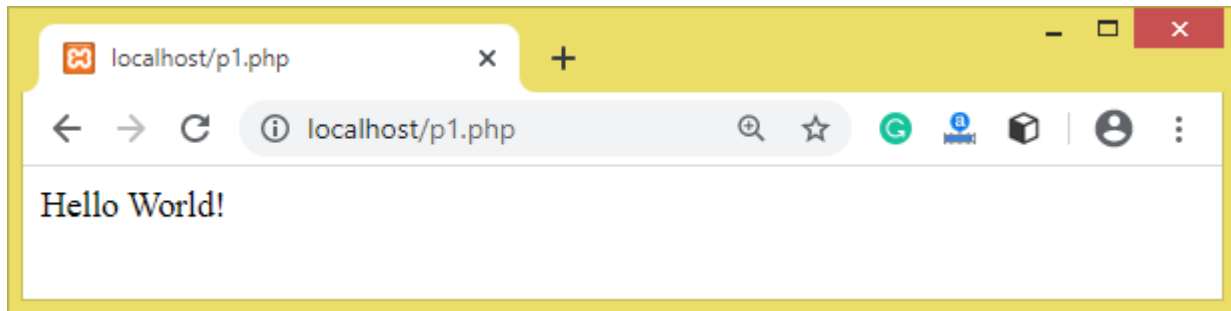
```
<?php
    echo "Hello World!";
?>
```

Step 2: Save the file with hello.php name in the htdocs folder, which resides inside the xampp folder.

Step 3: Run the XAMPP server and start the Apache and MySQL.

Step 4: Now, open the web browser and type localhost *http://localhost/hello.php* on your browser window.

Step 5: The output for the above hello.php program will be shown as the screenshot below:



Most of the time, PHP programs run as a web server module. However, PHP can also be run on CLI (Command Line Interface).

PHP Case Sensitivity

In PHP, keyword (e.g., echo, if, else, while), functions, user-defined functions, classes are not case-sensitive. However, all variable names are case-sensitive.

In the below example, you can see that all three echo statements are equal and valid:

```
<!DOCTYPE>
<html>
  <body>
    <?php
      echo "Hello world using echo </br>";
      ECHO "Hello world using ECHO </br>";
      EcHo "Hello world using EcHo </br>";
    ?>
  </body>
</html>
```

Look at the below example that the variable names are case sensitive. You can see the example below that only the second statement will display the value of the \$color variable. Because it treats \$color, \$Color, and \$COLOR as three different variables:

```

<html>
  <body>
    <?php
      $color = "black";
      echo "My car is ". $Color . "</br>";
      echo "My dog is ". $color . "</br>";
      echo "My Phone is ". $COLOR . "</br>";
    ?>
  </body>
</html>

```

Only \$color variable has printed its value, and other variables \$Color and \$COLOR are declared as undefined variables. An error has occurred in line 5 and line 7.

PHP Echo

PHP echo is a language construct, not a function. Therefore, you don't need to use parenthesis with it. But if you want to use more than one parameter, 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 the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:

- echo is a statement, which is used to display the output.
- echo can be used with or without parentheses: echo(), and echo.
- echo does not return any value.
- We can pass multiple strings separated by a comma (,) in echo.
- echo is faster than the print statement.

PHP echo: printing string

File: echo1.php

```
<?php
echo "Hello by PHP echo";
?>
```

PHP echo: printing multi line string

File: echo2.php

```
<?php
echo "Hello by PHP echo
this is multi line
text printed by
PHP echo statement
";
?>
```

PHP echo: printing escaping characters

File: echo3.php

```
<?php
echo "Hello escape \"sequence\" characters";
?>
```

PHP echo: printing variable value

File: echo4.php

```
<?php
$msg="Hello JavaTpoint PHP";
echo "Message is: $msg";
?>
```

PHP Print

Like PHP echo, PHP print is a language construct, so you don't need to use parenthesis with the argument list. Print statement can be used with or without parentheses: print and print(). 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 the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:

- print is a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than the echo statement.

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

```
<?php
print "Hello by PHP print
```

```
this is multi line
text printed by
PHP print statement
";
?>
```

PHP print: printing escaping characters

```
<?php
print "Hello escape \"sequence\" characters by PHP print";
?>
```

PHP print: printing variable value

File: print4.php

```
<?php
$msg="Hello print() in PHP";
print "Message is: $msg";
?>
```

PHP echo and print Statements

We frequently use the echo statement to display the output. There are two basic ways to get the output in PHP:

- echo
- print

echo and print are language constructs, and they never behave like a function. Therefore, there is no requirement for parentheses. However, both the statements can be used with or without parentheses. We can use these statements to output variables or strings.

Difference between echo and print

echo

- echo is a statement, which is used to display the output.
- echo can be used with or without parentheses.
- echo does not return any value.
- We can pass multiple strings separated by comma (,) in echo.
- echo is faster than print statement.

print

- print is also a statement, used as an alternative to echo at many times to display the output.
- print can be used with or without parentheses.
- print always returns an integer value, which is 1.
- Using print, we cannot pass multiple arguments.
- print is slower than echo statement.

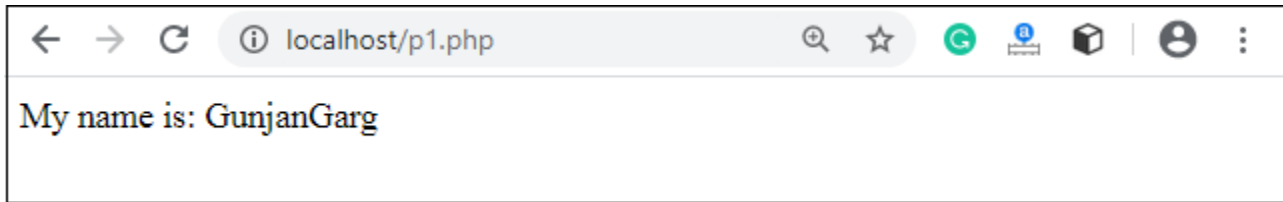
You can see the difference between echo and print statements with the help of the following programs.

For Example (Check multiple arguments)

You can pass multiple arguments separated by a comma (,) in echo. It will not generate any syntax error.

```
<?php
    $fname = "Gunjan";
    $lname = "Garg";
    echo "My name is: ".$fname,$lname;
?>
```

Output:



It will generate a syntax error because of multiple arguments in a print statement.

```
<?php
    $fname = "Gunjan";
    $lname = "Garg";
    print "My name is: ".$fname,$lname;
?>
```

Output:

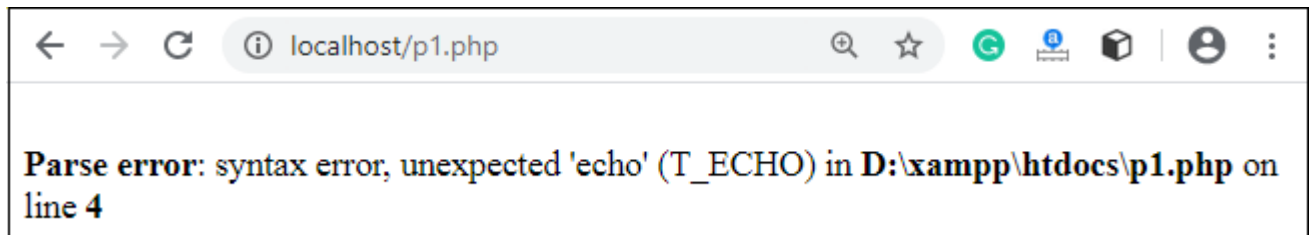


For Example (Check Return Value)

echo statement does not return any value. It will generate an error if you try to display its return value.

```
<?php
    $lang = "PHP";
    $ret = echo $lang." is a web development language.";
    echo "<br>";
    echo "Value return by print statement: ".$ret;
?>
```

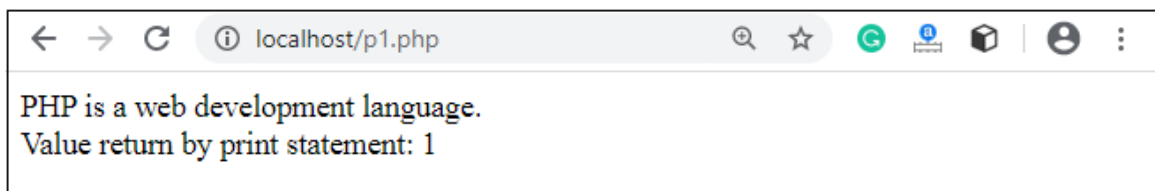
Output:



As we already discussed that print returns a value, which is always 1.

```
<?php
    $lang = "PHP";
    $ret = print $lang." is a web development language.";
    print "</br>";
print "Value return by print statement: ".$ret;
?>
```

1. Output:



2.

PHP Variables

```
<="" p="" style="color: rgb(51, 51, 51); font-family: inter-regular, system-ui, -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Helvetica, Arial, sans-serif; font-size: 16px; font-style: normal; font-variant-ligatures: normal; font-variant-caps: normal; font-weight: 400; letter-spacing: normal; orphans: 2; text-align: justify; text-indent: 0px; text-transform: none; white-space: normal; widows: 2; word-spacing: 0px; -webkit-text-stroke-width: 0px; background-color: rgb(255, 255, 255); text-decoration-thickness: initial; text-decoration-style: initial; text-decoration-color: initial;">
```

In PHP, a variable is declared using a \$ sign followed by the variable name. Here, some important points to know about variables:

- As PHP is a loosely typed language, so we do not need to declare the data types of the variables. It automatically analyzes the values and makes conversions to its correct datatype.

PHP Variable Scope

The scope of a variable is defined as its range in the program under which it can be accessed. In other words, "The scope of a variable is the portion of the program within which it is defined and can be accessed."

PHP has three types of variable scopes:

1. Local variable
2. Global variable
3. Static variable

Local variable

The variables that are declared within a function are called local variables for that function. These local variables have their scope only in that particular function in which they are declared. This means that these variables cannot be accessed outside the function, as they have local scope.

A variable declaration outside the function with the same name is completely different from the variable declared inside the function. Let's understand the local variables with the help of an example:

File: local_variable1.php

```
<?php
function local_var()
{
    $num = 45; //local variable
    echo "Local variable declared inside the function is: ". $num;
}
```

```

    local_var();
?>
<?php
    function mytest()
    {
        $lang = "PHP";
        echo "Web development language: " . $lang;
    }
    mytest();
    //using $lang (local variable) outside the function will generate an error
    echo $lang;
?>

```

Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program. To access the global variable within a function, use the GLOBAL keyword before the variable. However, these variables can be directly accessed or used outside the function without any keyword. Therefore there is no need to use any keyword to access a global variable outside the function.

Let's understand the global variables with the help of an example:

Example:

File: global_variable1.php

```

<?php
$name = "Sanaya Sharma";    //Global Variable
function global_var()
{
    global $name;
    echo "Variable inside the function: ". $name;
    echo "</br>";
}

```

```
}  
global_var();  
echo "Variable outside the function: ". $name; ?>
```

Output:

```
Variable inside the function: Sanaya Sharma  
Variable outside the function: Sanaya Sharma
```

Note: Without using the global keyword, if you try to access a global variable inside the function, it will generate an error that the variable is undefined.

Example:

File: global_variable2.php

```
<?php  
$name = "Sanaya Sharma";    //global variable  
function global_var()  
{  
    echo "Variable inside the function: ". $name;  
    echo "<br>";  
}  
global_var();  
?>
```

Using \$GLOBALS instead of global

Another way to use the global variable inside the function is predefined \$GLOBALS array.

Example:

File: global_variable3.php

```
<?php  
$num1 = 5;    //global variable  
$num2 = 13;   //global variable  
function global_var()
```

```
{
    $sum = $GLOBALS['num1'] + $GLOBALS['num2'];
    echo "Sum of global variables is: " . $sum;
}
global_var();
?>
```

If two variables, local and global, have the same name, then the local variable has higher priority than the global variable inside the function.

Example:

File: global_variable2.php

```
<?php
    $x = 5;
    function mytest()
    {
        $x = 7;
        echo "value of x: " . $x;
    }
    mytest();
?>
```

Static variable

It is a feature of PHP to delete the variable, once it completes its execution and memory is freed. Sometimes we need to store a variable even after completion of function execution. Therefore, another important feature of variable scoping is static variable. We use the static keyword before the variable to define a variable, and this variable is called as static variable. Static variables exist only in a local function, but it does not free its memory after the program execution leaves the scope. Understand it with the help of an example:

Example:

File: static_variable.php

```
<?php
function static_var()
{
    static $num1 = 3;    //static variable
    $num2 = 6;         //Non-static variable
    //increment in non-static variable
    $num1++;
    //increment in static variable
    $num2++;
    echo "Static: " . $num1 . "<br>";
    echo "Non-static: " . $num2 . "<br>";
}

//first function call
static_var();

//second function call
static_var();
?>
```

You have to notice that \$num1 regularly increments after each function call, whereas \$num2 does not. This is why because \$num1 is not a static variable, so it freed its memory after the execution of each function call.

PHP \$ and \$\$ Variables

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

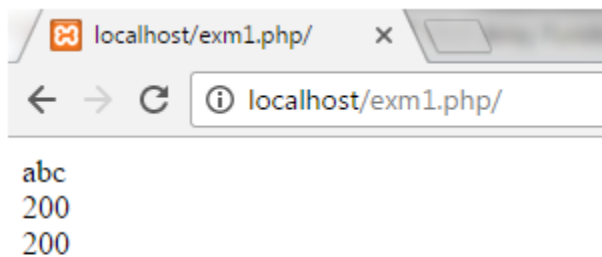
The \$\$var (double dollar) is a reference variable that stores the value of the \$variable inside it.

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

Example 1

```
<?php
$x = "abc";
$$x = 200;
echo $x."<br/>";
echo $$x."<br/>";
echo $abc;
?>
```

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.

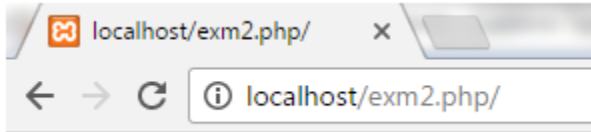
Now we have printed the values \$x, \$\$x and \$abc.

Example2

```
<?php
$x="U.P";
$$x="Lucknow";
echo $x. "<br>";
echo $$x. "<br>";
echo "Capital of $x is " . $$x;
```

?>

Output:



U.P
Lucknow
Capital of U.P is Lucknow

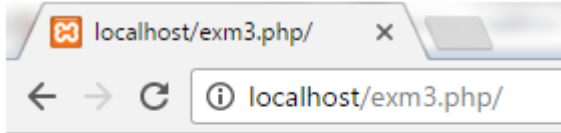
In the above example, we have assigned a value to the variable x as U.P. Value of reference variable \$\$x is assigned as Lucknow.

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

Example3

```
<?php
$name="Cat";
${$name}="Dog";
${${$name}}="Monkey";
echo $name. "<br>";
echo ${$name}. "<br>";
echo $Cat. "<br>";
echo ${${$name}}. "<br>";
echo $Dog. "<br>";
?>
```

Output:



Cat
Dog
Dog
Monkey
Monkey

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 except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:

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

Constants are similar to the variable except once they defined, they can never be undefined or changed. They remain constant across the entire program. PHP constants follow the same PHP variable rules. For example, it can be started with a letter or underscore only.

Conventionally, PHP constants should be defined in uppercase letters.

PHP constant: `define()`

Use the `define()` function to create a constant. It defines constant at run time. Let's see the syntax of `define()` function in PHP.

1. `define(name, value, case-insensitive)`

1. `name`: It specifies the constant name.

2. value: It specifies the constant value.
3. case-insensitive: Specifies whether a constant is 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

```
<?php
define("MESSAGE","Hello JavaTpoint PHP");
echo MESSAGE;
?>
```

Output:

```
Hello JavaTpoint PHP
```

Create a constant with case-insensitive name:

File: constant2.php

```
<?php
define("MESSAGE","Hello JavaTpoint PHP",true);//not case sensitive
echo MESSAGE, "</br>";
echo message;
?>
```

Output:

File: constant3.php

```
<?php
define("MESSAGE","Hello JavaTpoint PHP",false);//case sensitive
echo MESSAGE;
echo message; ?>
```

Output:

PHP constant: const keyword

PHP introduced a keyword const to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function. The constant defined using const keyword are case-sensitive.

File: constant4.php

```
<?php
const MESSAGE="Hello const by JavaTpoint PHP";
echo MESSAGE;
?>
```

Constant() function:

There is another way to print the value of constants using constant() function instead of using the echo statement.

Syntax

The syntax for the following constant function:

```
constant (name)
```

File: constant5.php

```
<?php
define("MSG", "JavaTpoint");
echo MSG, "</br>";
echo constant("MSG");
//both are similar
?>
```

Output:

JavaTpoint

JavaTpoint

Constant vs Variables

Constant	Variables
Once the constant is defined, it can never be redefined.	A variable can be undefined as well as redefined easily.
A constant can only be defined using define() function. It cannot be defined by any simple assignment.	A variable can be defined by simple assignment (=) operator.
There is no need to use the dollar (\$) sign before constant during the assignment.	To declare a variable, always use the dollar (\$) sign before the variable.
Constants do not follow any variable scoping rules, and they can be defined and accessed anywhere.	Variables can be declared anywhere in the program, but they follow variable scoping rules.
Constants are the variables whose values can't be changed throughout the program.	The value of the variable can be changed.
By default, constants are global.	Variables can be local, global, or static.

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 nine magic constants in PHP. In which eight magic constants start and end with double underscores (___).

___LINE__

___FILE__

___DIR__

___FUNCTION__

___CLASS__

___TRAIT__

___METHOD__

___NAMESPACE__

ClassName::class

All of the constants are resolved at compile-time instead of run time, unlike the regular constant. Magic constants are case-insensitive.

Changelog

Version	Description
5.3.0	Added ___DIR__ and ___NAMESPACE__ magic constant
5.4.0	Added ___TRAIT__ magic constant
5.5.0	Added ::class magic constant

All the constants are defined below with the example code:

1. `__LINE__`

It returns the current line number of the file, where this constant is used.

Example:

```
<?php
echo "<h3>Example for __LINE__</h3>";
// print Your current line number i.e;4
echo "You are at line number " . __LINE__ . "<br><br>";
?>
```

Output: Example for `__LINE__`

You are at line number " . `__LINE__`

2. `__FILE__`:

This magic constant returns the full path of the executed file, where the file is stored. If it is used inside the include, the name of the included file is returned.

Example:

```
<?php
echo "<h3>Example for __FILE__</h3>";
//print full path of file with .php extension
echo __FILE__ . "<br><br>";
?>
```

Output: Example for `__FILE__`

D:\xampp\htdocs\program\magic.php3. `__DIR__`:

It returns the full directory path of the executed file. The path returned by this magic constant is equivalent to `dirname(__FILE__)`. This magic constant does not have a trailing slash unless it is a root directory.

Example:

```
<?php
echo "<h3>Example for __DIR__</h3>";
//print full path of directory where script will be placed
echo __DIR__ . "<br><br>";
//below output will equivalent to above one.
echo dirname(__FILE__) . "<br><br>";
?>
```

4. **__FUNCTION__**:

This magic constant returns the function name, where this constant is used. It will return blank if it is used outside of any function.

Example:

```
<?php
echo "<h3>Example for __FUNCTION__</h3>";
//Using magic constant inside function.
function test(){
    //print the function name i.e; test.
    echo "The function name is ' . __FUNCTION__ . "<br><br>";
}
test();

//Magic constant used outside function gives the blank output.
function test_function(){
    echo 'Hie';
}
```

```
test_function();
```

```
//give the blank output.
```

```
    echo __FUNCTION__ . "<br><br>"; ?>
```

5. __CLASS__:

It returns the class name, where this magic constant is used. __CLASS__ constant also works in traits.

Example:

```
<?php
```

```
echo "<h3>Example for __CLASS__</h3>";
```

```
class JTP
```

```
{
```

```
    public function __construct() {
```

```
        ;
```

```
    }
```

```
function getClassName(){
```

```
    //print name of the class JTP.
```

```
    echo __CLASS__ . "<br><br>";
```

```
    }
```

```
}
```

```
$t = new JTP;
```

```
$t->getClassName();
```

```
//in case of multiple classes
```

```
class base
```

```
{
```

```
function test_first(){
```

```
    //will always print parent class which is base here.
```

```
    echo __CLASS__;
```

```
    }
```

```

}
class child extends base
{
    public function __construct() {
        ;
    }
}
$t = new child;
$t->test_first();
?>

```

6. **__TRAIT__**:

This magic constant returns the trait name, where it is used.

Example:

```

<?php
echo "<h3>Example for __TRAIT__</h3>";
trait created_trait {
    function jtp(){
        //will print name of the trait i.e; created_trait
        echo __TRAIT__;
    }
}
class Company {
    use created_trait;
}
$a = new Company;
$a->jtp();
?>

```


7. `__METHOD__`:

It returns the name of the class method where this magic constant is included. The method name is returned the same as it was declared.

Example:

```
<?php
echo "<h3>Example for __METHOD__</h3>";
class method {
    public function __construct() {
        //print method::__construct
        echo __METHOD__ . "<br><br>";
    }
    public function meth_fun(){
        //print method::meth_fun
        echo __METHOD__;
    }
}
$a = new method;
$a->meth_fun();
?>
```

8. `__NAMESPACE__`:

It returns the current namespace where it is used.

Example:

```
<?php
echo "<h3>Example for __NAMESPACE__</h3>";
class name {
    public function __construct() {
        echo 'This line will print on calling namespace.';
    }
}
```

```
    }  
}  
$class_name = __NAMESPACE__ . '\name';  
$a = new class_name;  
?>
```

9. ClassName::class:

This magic constant does not start and end with the double underscore (__). It returns the fully qualified name of the ClassName. ClassName::class is added in PHP 5.5.0. It is useful with namespaced classes.

Example:

```
<?php  
namespace Technical_Portal;  
echo "<h3>Example for CLASSNAME::CLASS </h3>";  
class javatpoint {  
}  
echo javatpoint::class; //ClassName::class  
?>
```

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 (predefined)
2. Compound Types (user-defined)
3. Special Types

PHP Data Types: Scalar Types

It holds only single value. There are 4 scalar data types in PHP.

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

PHP Data Types: Compound Types

It can hold multiple values. 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 Boolean

Booleans are the simplest data type works like switch. It holds only two values: TRUE (1) or FALSE (0). It is often used with conditional statements. If the condition is correct, it returns TRUE otherwise FALSE.

Example:

```
<?php
if (TRUE)
    echo "This condition is TRUE.";
if (FALSE)
    echo "This condition is FALSE.";
```

?>

PHP Integer

Integer means numeric data with a negative or positive sign. It holds only whole numbers, i.e., numbers without fractional part or decimal points.

Rules for integer:

- An integer can be either positive or negative.
- An integer must not contain decimal point.
- Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).
- The range of an integer must be lie between -2^{31} and 2^{31} i.e., -2^{31} to 2^{31} .

Example:

```
<?php
    $dec1 = 34;
    $oct1 = 0243;
    $hexa1 = 0x45;
    echo "Decimal number: " . $dec1. "<br>";
    echo "Octal number: " . $oct1. "<br>";
    echo "HexaDecimal number: " . $hexa1. "<br>";
?>
```

PHP Float

A floating-point number is a number with a decimal point. Unlike integer, it can hold numbers with a fractional or decimal point, including a negative or positive sign.

Example:

```
<?php
```

```
$n1 = 19.34;
$n2 = 54.472;
$sum = $n1 + $n2;
echo "Addition of floating numbers: " . $sum;
?>
```

PHP String

A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.

String values must be enclosed either within single quotes or in double quotes. But both are treated differently. To clarify this, see the example below:

Example:

```
<?php
$company = "Javatpoint";
//both single and double quote statements will treat different
echo "Hello $company";
echo "</br>";
echo 'Hello $company';
?>
```

PHP Array

An array is a compound data type. It can store multiple values of same data type in a single variable.

Example:

```
<?php
$bikes = array ("Royal Enfield", "Yamaha", "KTM");
var_dump($bikes); //the var_dump() function returns the datatype and values
echo "</br>";
```

```
echo "Array Element1: $bikes[0] </br>";  
echo "Array Element2: $bikes[1] </br>";  
echo "Array Element3: $bikes[2] </br>";  
?>
```

You will learn more about array in later chapters of this tutorial.

PHP object

Objects are the instances of user-defined classes that can store both values and functions. They must be explicitly declared.

Example:

```
<?php  
class bike {  
    function model() {  
        $model_name = "Royal Enfield";  
        echo "Bike Model: " . $model_name;  
    }  
}  
$obj = new bike();  
$obj -> model();  
?>
```

This is an advanced topic of PHP, which we will discuss later in detail.

PHP Resource

Resources are not the exact data type in PHP. Basically, these are used to store some function calls or references to external PHP resources. For example - a database call. It is an external resource.

This is an advanced topic of PHP, so we will discuss it later in detail with examples.

PHP Null

Null is a special data type that has only one value: NULL. There is a convention of writing it in capital letters as it is case sensitive.

The special type of data type NULL defined a variable with no value.

Example:

```
<?php
    $nl = NULL;
    echo $nl; //it will not give any output
?>
```

PHP Operators

PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values. 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
- Assignment Operators
- Bitwise Operators
- Comparison Operators
- Incrementing/Decrementing Operators
- Logical Operators
- String Operators
- Array Operators
- Type Operators

- Execution Operators
- Error Control 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 "?:".

Arithmetic Operators

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

Operator	Name	Example	Explanation
+	Addition	\$a + \$b	Sum of operands
-	Subtraction	\$a - \$b	Difference of operands
*	Multiplication	\$a * \$b	Product of operands
/	Division	\$a / \$b	Quotient of operands
%	Modulus	\$a % \$b	Remainder of operands
**	Exponentiation	\$a ** \$b	\$a raised to the power \$b

The exponentiation (**) operator has been introduced in PHP 5.6.

Assignment Operators

The assignment operators are used to assign value to different variables. The basic assignment operator is "=".

Operator	Name	Example	Explanation
=	Assign	$\$a = \b	The value of right operand is assigned to the left operand.
+=	Add then Assign	$\$a += \b	Addition same as $\$a = \$a + \$b$
-=	Subtract then Assign	$\$a -= \b	Subtraction same as $\$a = \$a - \$b$
*=	Multiply then Assign	$\$a *= \b	Multiplication same as $\$a = \$a * \$b$
/=	Divide then Assign (quotient)	$\$a /= \b	Find quotient same as $\$a = \$a / \$b$
%=	Divide then Assign (remainder)	$\$a \% = \b	Find remainder same as $\$a = \$a \% \$b$

Bitwise Operators

The bitwise operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
&	And	\$a & \$b	Bits that are 1 in both \$a and \$b are set to 1, otherwise 0.
	Or (Inclusive or)	\$a \$b	Bits that are 1 in either \$a or \$b are set to 1
^	Xor (Exclusive or)	\$a ^ \$b	Bits that are 1 in either \$a or \$b are set to 0.
~	Not	~\$a	Bits that are 1 set to 0 and bits that are 0 are set to 1
<<	Shift left	\$a << \$b	Left shift the bits of operand \$a \$b steps
>>	Shift right	\$a >> \$b	Right shift the bits of \$a operand by \$b number of places

Comparison Operators

Comparison operators allow comparing two values, such as number or string. Below the list of comparison operators are given:

Operator	Name	Example	Explanation
==	Equal	\$a == \$b	Return TRUE if \$a is equal to \$b
===	Identical	\$a === \$b	Return TRUE if \$a is equal to \$b, and they are of same data type
!==	Not identical	\$a !== \$b	Return TRUE if \$a is not equal to \$b, and they are not of same data type

!=	Not equal	\$a != \$b	Return TRUE if \$a is not equal to \$b
<>	Not equal	\$a <> \$b	Return TRUE if \$a is not equal to \$b
<	Less than	\$a < \$b	Return TRUE if \$a is less than \$b
>	Greater than	\$a > \$b	Return TRUE if \$a is greater than \$b
<=	Less than or equal to	\$a <= \$b	Return TRUE if \$a is less than or equal \$b
>=	Greater than or equal to	\$a >= \$b	Return TRUE if \$a is greater than or equal \$b
<=>	Spaceship	\$a <=>\$b	Return -1 if \$a is less than \$b Return 0 if \$a is equal \$b Return 1 if \$a is greater than \$b

Incrementing/Decrementing Operators

The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Example	Explanation
++	Increment	++\$a	Increment the value of \$a by one, then return \$a
		\$a++	Return \$a, then increment the value of \$a by one

--	decrement	--\$a	Decrement the value of \$a by one, then return \$a
		\$a--	Return \$a, then decrement the value of \$a by one

Logical Operators

The logical operators are used to perform bit-level operations on operands. These operators allow the evaluation and manipulation of specific bits within the integer.

Operator	Name	Example	Explanation
and	And	\$a and \$b	Return TRUE if both \$a and \$b are true
Or	Or	\$a or \$b	Return TRUE if either \$a or \$b is true
xor	Xor	\$a xor \$b	Return TRUE if either \$ or \$b is true but not both
!	Not	! \$a	Return TRUE if \$a is not true
&&	And	\$a && \$b	Return TRUE if either \$a and \$b are true
	Or	\$a \$b	Return TRUE if either \$a or \$b is true

String Operators

The string operators are used to perform the operation on strings. There are two string operators in PHP, which are given below:

Operator	Name	Example	Explanation
.	Concatenation	$\$a . \b	Concatenate both $\$a$ and $\$b$
.=	Concatenation and Assignment	$\$a .= \b	First concatenate $\$a$ and $\$b$, then assign the concatenated string to $\$a$, e.g. $\$a = \$a . \$b$

Array Operators

The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

Operator	Name	Example	Explanation
+	Union	$\$a + \y	Union of $\$a$ and $\$b$
==	Equality	$\$a == \b	Return TRUE if $\$a$ and $\$b$ have same key/value pair
!=	Inequality	$\$a != \b	Return TRUE if $\$a$ is not equal to $\$b$
===	Identity	$\$a === \b	Return TRUE if $\$a$ and $\$b$ have same key/value pair of same type in same order
!==	Non-Identity	$\$a !== \b	Return TRUE if $\$a$ is not identical to $\$b$
<>	Inequality	$\$a <> \b	Return TRUE if $\$a$ is not equal to $\$b$

Type Operators

The type operator `instanceof` is used to determine whether an object, its parent and its derived class are the same type or not. Basically, this operator determines which certain class the object belongs to. It is used in object-oriented programming.

```
<?php
//class declaration
class Developer
{
class Programmer
{
//creating an object of type Developer
$charu = new Developer();

//testing the type of object
if( $charu instanceof Developer)
{
    echo "Charu is a developer.";
}
else
{
    echo "Charu is a programmer.";
}
echo "</br>";
var_dump($charu instanceof Developer);    //It will return true.
var_dump($charu instanceof Programmer);    //It will return false.
?>
```

Execution Operators

PHP has an execution operator backticks (`). PHP executes the content of backticks as a shell command. Execution operator and shell_exec() give the same result.

Operator	Name	Example	Explanation
`	backticks	echo `dir`;	Execute the shell command and return the result. Here, it will show the directories available in current folder.

Error Control Operators

PHP has one error control operator, i.e., at (@) symbol. Whenever it is used with an expression, any error message will be ignored that might be generated by that expression.

Operator	Name	Example	Explanation
@	at	@file ('non_existent_file')	Intentional file error

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

<code> </code>	logical OR	left
<code>?:</code>	ternary	left
<code>= += -= *= **= /= .= %= &= = ^= <<= >>= ==</code>	assignment	right
<code>and</code>	logical	left
<code>xor</code>	logical	left
<code>or</code>	logical	left
<code>,</code>	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)

```
<?php
```

```
// this is C++ style single line comment
```

```
# this is Unix Shell style single line comment
```

```
echo "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.

```
<?php
/*
Anything placed
within comment
will not be displayed
on the browser;
*/
echo "Welcome to PHP multi line comment";
?>
```

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

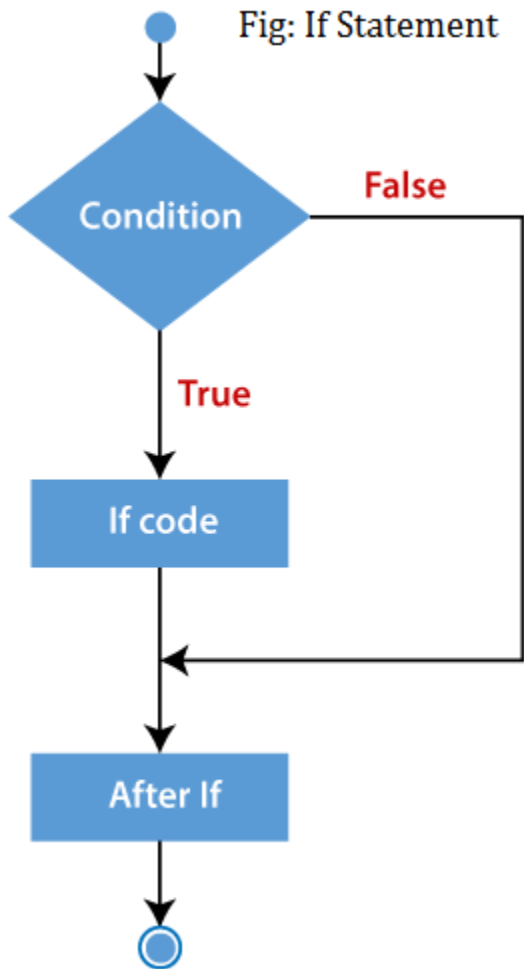
PHP if statement allows conditional execution of code. It is executed if condition is true.

If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.

Syntax

```
if(condition){  
//code to be executed  
}
```

Flowchart



Example

```
<?php  
$num=12;  
if($num<100){  
echo "$num is less than 100";  
}
```

?>

PHP If-else Statement

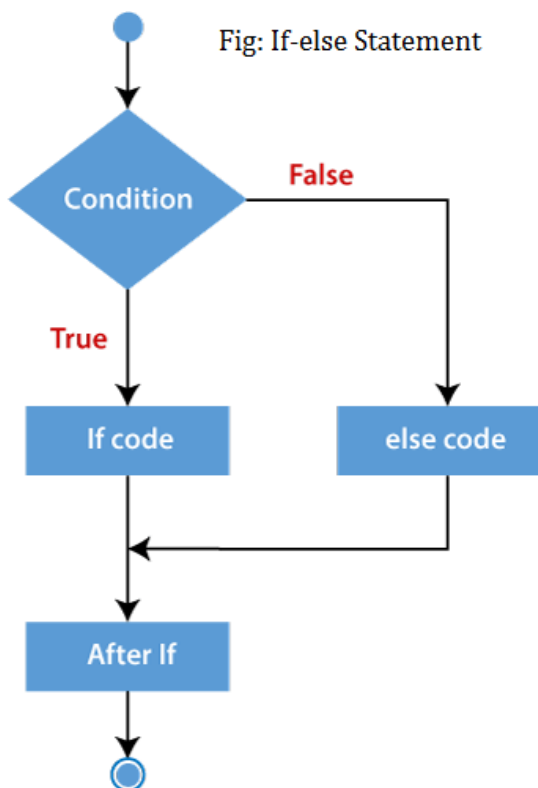
PHP if-else statement is executed whether condition is true or false.

If-else statement is slightly different from if statement. It executes one block of code if the specified condition is true and another block of code if the condition is false.

Syntax

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

Flowchart



Example

```
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

PHP If-else-if Statement

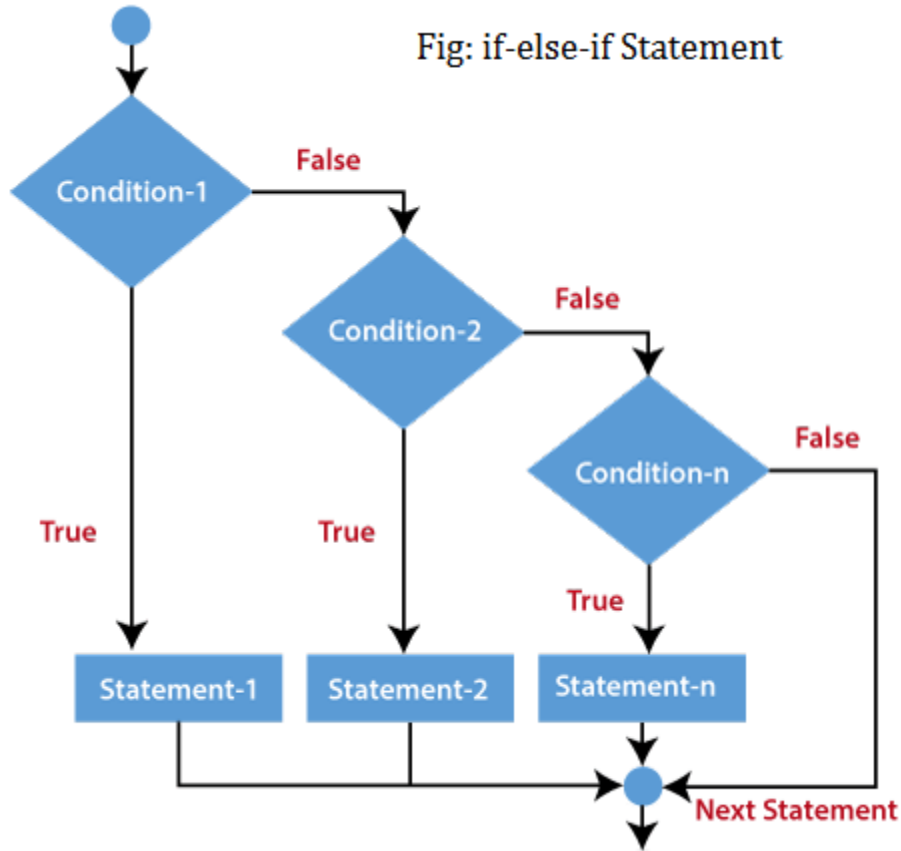
The PHP if-else-if is a special statement used to combine multiple if?.else statements. So, we can check multiple conditions using this statement.

Syntax

```
if (condition1){
//code to be executed if condition1 is true
} elseif (condition2){
//code to be executed if condition2 is true
} elseif (condition3){
//code to be executed if condition3 is true
....
} else{
//code to be executed if all given conditions are false
}
```

Flowchart

Fig: if-else-if Statement



Example

```
<?php
$marks=69;
if ($marks<33){
    echo "fail";
}
else if ($marks>=34 && $marks<50) {
    echo "D grade";
}
else if ($marks>=50 && $marks<65) {
    echo "C grade";
}
else if ($marks>=65 && $marks<80) {
    echo "B grade";
}
```

```
}  
else if ($marks>=80 && $marks<90) {  
    echo "A grade";  
}  
else if ($marks>=90 && $marks<100) {  
    echo "A+ grade";  
}  
else {  
    echo "Invalid input";  
}  
?>
```

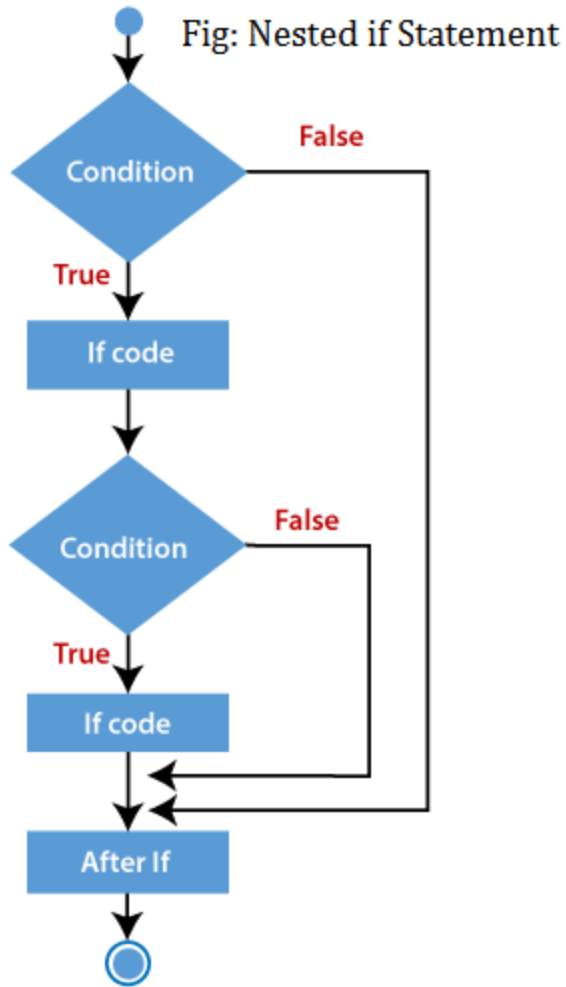
PHP nested if Statement

The nested if statement contains the if block inside another if block. The inner if statement executes only when specified condition in outer if statement is true.

Syntax

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

Flowchart



Example

```
<?php
    $age = 23;
    $nationality = "Indian";
    //applying conditions on nationality and age
    if ($nationality == "Indian")
    {
        if ($age >= 18) {
            echo "Eligible to give vote";
        }
        else {
```



```
        echo "Not eligible to give vote";
    }
}
?>
```

PHP Switch Example

```
<?php
    $a = 34; $b = 56; $c = 45;
    if ($a < $b) {
        if ($a < $c) {
            echo "$a is smaller than $b and $c";
        }
    }
?>
```

PHP Switch

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

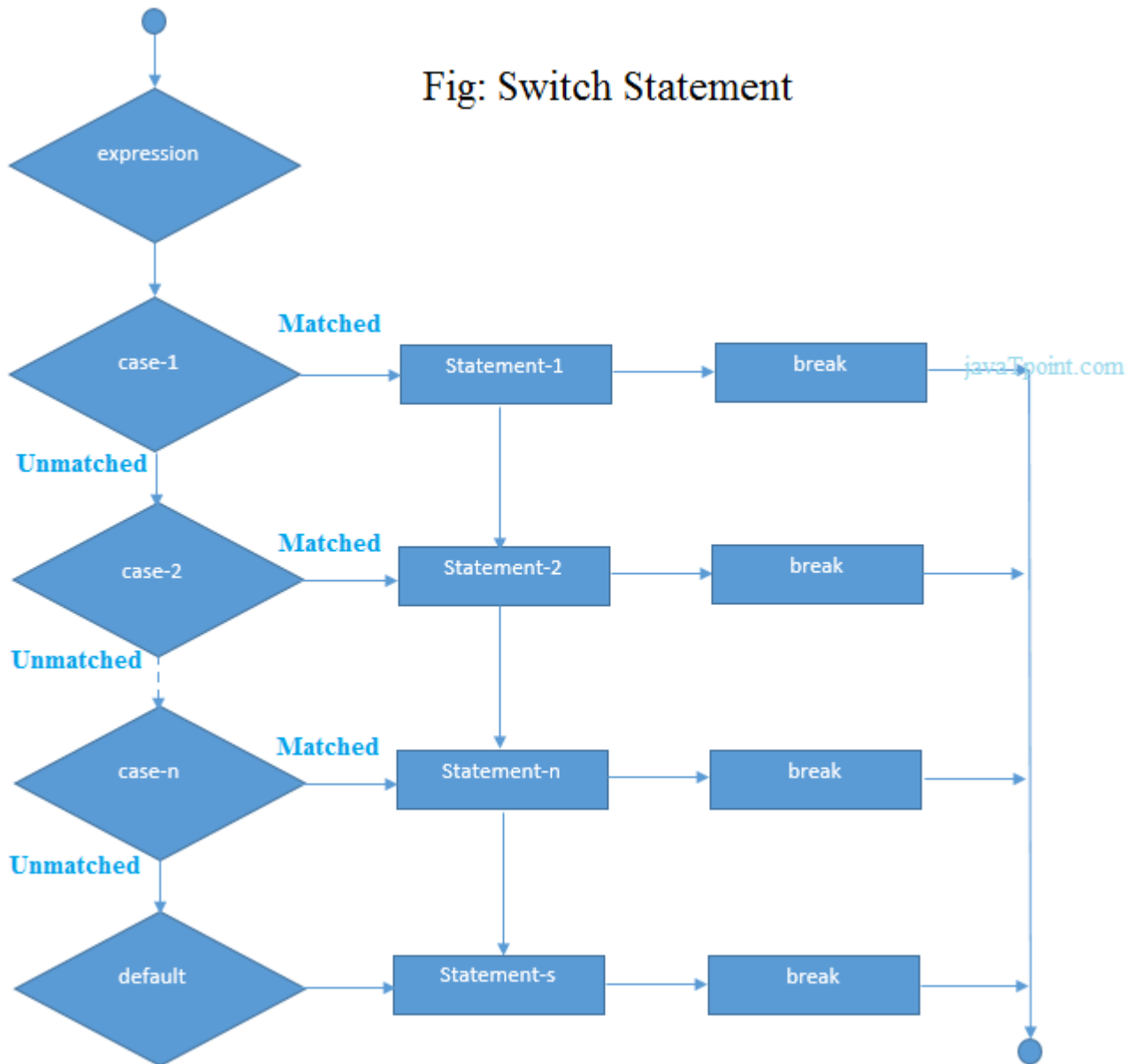
Syntax

```
switch(expression){
case value1:
    //code to be executed
    break;
case value2:
    //code to be executed
    break;
.....
default:
    code to be executed if all cases are not matched;
}
```

Important points to be noticed about switch case:

1. The default is an optional statement. Even it is not important, that default must always be the last statement.
2. There can be only one default in a switch statement. More than one default may lead to a Fatal error.
3. Each case can have a break statement, which is used to terminate the sequence of statement.
4. The break statement is optional to use in switch. If break is not used, all the statements will execute after finding matched case value.
5. PHP allows you to use number, character, string, as well as functions in switch expression.
6. Nesting of switch statements is allowed, but it makes the program more complex and less readable.
7. You can use semicolon (;) instead of colon (:). It will not generate any error.

PHP Switch Flowchart



PHP Switch Example

```
<?php
$num=20;
switch($num){
case 10:
echo("number is equals to 10");
```

```
break;
case 20:
echo("number is equal to 20");
break;
case 30:
echo("number is equal to 30");
break;
default:
echo("number is not equal to 10, 20 or 30");
}
?>
```

PHP switch statement with character

Program to check Vowel and consonant

We will pass a character in switch expression to check whether it is vowel or constant. If the passed character is A, E, I, O, or U, it will be vowel otherwise consonant.

```
<?php
$ch = 'U';
switch ($ch)
{
case 'a':
    echo "Given character is vowel";
    break;
case 'e':
    echo "Given character is vowel";
    break;
case 'i':
    echo "Given character is vowel";
    break;
```

```
case 'o':
    echo "Given character is vowel";
    break;
case 'u':
    echo "Given character is vowel";
    break;
case 'A':
    echo "Given character is vowel";
    break;
case 'E':
    echo "Given character is vowel";
    break;
case 'I':
    echo "Given character is vowel";
    break;
case 'O':
    echo "Given character is vowel";
    break;
case 'U':
    echo "Given character is vowel";
    break;
default:
    echo "Given character is consonant";
    break;
}
?>
```

PHP switch statement with String

PHP allows to pass string in switch expression. Let's see the below example of course duration by passing string in switch case statement.

```
<?php
    $sch = "B.Tech";
    switch ($sch)
    {
        case "BCA":
            echo "BCA is 3 years course";
            break;
        case "Bsc":
            echo "Bsc is 3 years course";
            break;
        case "B.Tech":
            echo "B.Tech is 4 years course";
            break;
        case "B.Arch":
            echo "B.Arch is 5 years course";
            break;
        default:
            echo "Wrong Choice";
            break;
    }
?>
```

PHP switch statement is fall-through

PHP switch statement is fall-through. It means it will execute all statements after getting the first match, if break statement is not found.

```
<?php
    $sch = 'c';
    switch ($sch)
    {
        case 'a':
```

```
        echo "Choice a";
        break;
    case 'b':
        echo "Choice b";
        break;
    case 'c':
        echo "Choice c";
        echo "</br>";
    case 'd':
        echo "Choice d";
        echo "</br>";
    default:
        echo "case a, b, c, and d is not found";
    }
?>
```

PHP nested switch statement

Nested switch statement means switch statement inside another switch statement. Sometimes it leads to confusion.

```
<?php
$car = "Hyundai";
$model = "Tucson";
switch( $car )
{
    case "Honda":
        switch( $model )
        {
            case "Amaze":
                echo "Honda Amaze price is 5.93 - 9.79 Lakh.";
                break;
        }
    }
}
```

```
        case "City":
            echo "Honda City price is 9.91 - 14.31 Lakh.";
            break;
    }
    break;
case "Renault":
    switch( $model )
    {
        case "Duster":
            echo "Renault Duster price is 9.15 - 14.83 L.";
            break;
        case "Kwid":
            echo "Renault Kwid price is 3.15 - 5.44 L.";
            break;
    }
    break;
case "Hyundai":
    switch( $model )
    {
        case "Creta":
            echo "Hyundai Creta price is 11.42 - 18.73 L.";
            break;
    }
case "Tucson":
    echo "Hyundai Tucson price is 22.39 - 32.07 L.";
    break;
    case "Xcent":
        echo "Hyundai Xcent price is 6.5 - 10.05 L.";
        break;
    }
    break;
} ?>
```


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 the number of iterations is known otherwise use while loop. This means for loop is used when you already know how many times you want to execute a block of code. It allows users to put all the loop related statements in one place. See in the syntax given below:

Syntax

```
for(initialization; condition; increment/decrement){  
//code to be executed  
}
```

Parameters

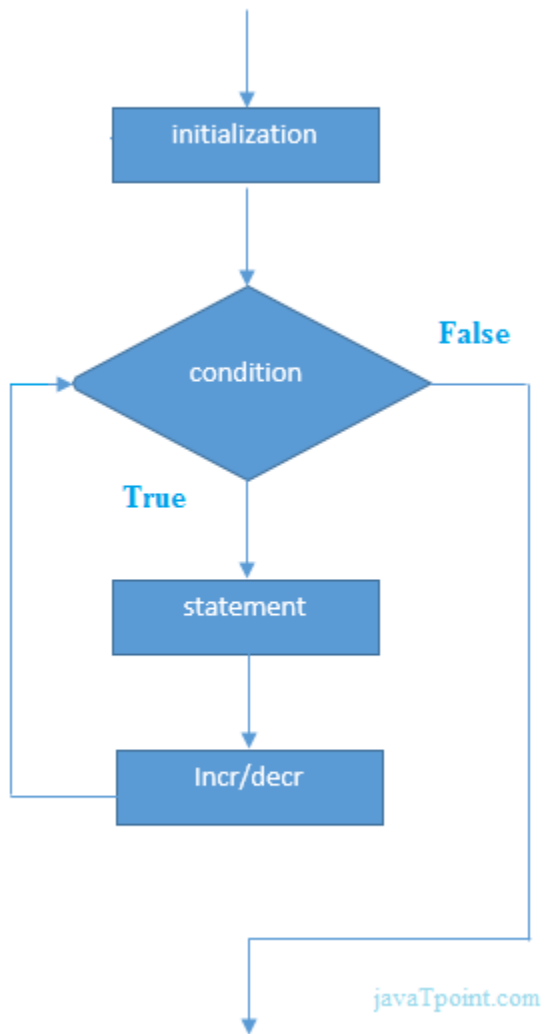
The php for loop is similar to the java/C/C++ for loop. The parameters of for loop have the following meanings:

initialization - Initialize the loop counter value. The initial value of the for loop is done only once. This parameter is optional.

condition - Evaluate each iteration value. The loop continuously executes until the condition is false. If TRUE, the loop execution continues, otherwise the execution of the loop ends.

Increment/decrement - It increments or decrements the value of the variable.

Flowchart



Example

```
<?php
for($n=1;$n<=10;$n++){
echo "$n<br/>";
}
?>
```

Example

All three parameters are optional, but semicolon (;) is must to pass in for loop. If we don't pass parameters, it will execute infinite.

```
<?php
    $i = 1;
    //infinite loop
    for (;;) {
        echo $i++;
        echo "</br>";
    }
?>
```

Example

Below is the example of printing numbers from 1 to 9 in four different ways using for loop.

```
<?php
    /* example 1 */

    for ($i = 1; $i <= 9; $i++) {
        echo $i;
    }
    echo "</br>";

    /* example 2 */

    for ($i = 1; ; $i++) {
        if ($i > 9) {
            break;
        }
        echo $i;
    }
    echo "</br>";

    /* example 3 */
```

```
$i = 1;
for (; ) {
    if ($i > 9) {
        break;
    }
    echo $i;
    $i++;
}
echo "<br>";
```

```
/* example 4 */
```

```
for ($i = 1, $j = 0; $i <= 9; $j += $i, print $i, $i++);
?>
```

PHP Nested For Loop

We can use for loop inside for loop in PHP, it is known as nested for loop. The inner for loop executes only when the outer for loop condition is found true.

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

```
<?php
for($i=1;$i<=3;$i++){
for($j=1;$j<=3;$j++){
echo "$i $j<br>";
}
}
?>
```

PHP For Each Loop

PHP for each loop is used to traverse array elements.

Syntax

```
foreach( $array as $var ){  
    //code to be executed  
}  
?>
```

Example

```
<?php  
$season=array("summer","winter","spring","autumn");  
foreach( $season as $arr ){  
    echo "Season is: $arr<br />";  
}  
?>
```

PHP foreach loop

The foreach loop is used to traverse the array elements. It works only on array and object. It will issue an error if you try to use it with the variables of different datatype.

The foreach loop works on elements basis rather than index. It provides an easiest way to iterate the elements of an array.

In foreach loop, we don't need to increment the value.

Syntax

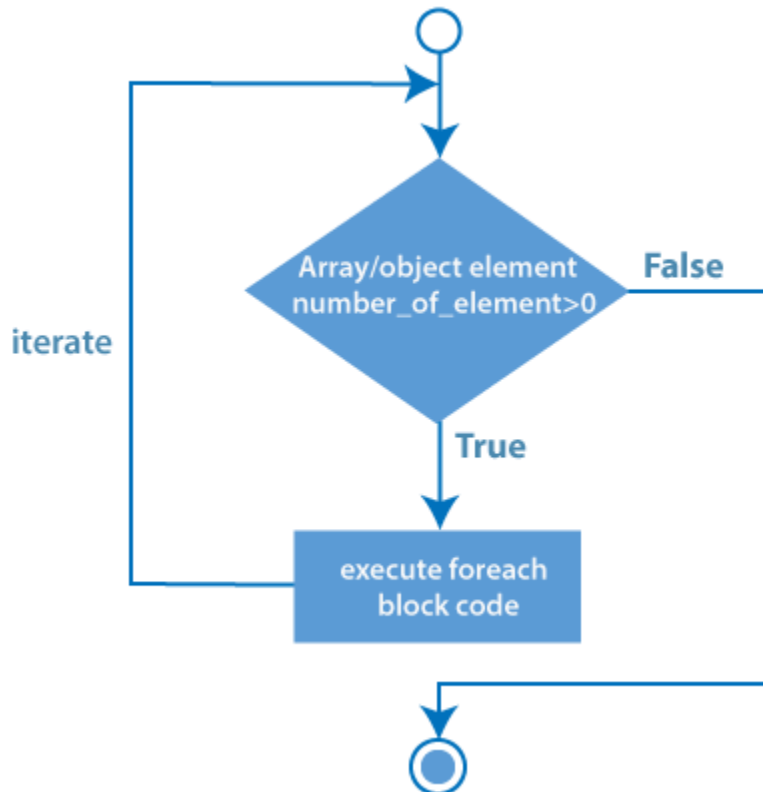
```
foreach ($array as $value) {  
    //code to be executed  
}
```

There is one more syntax of foreach loop.

Syntax

```
foreach ($array as $key => $element) {  
    //code to be executed  
}
```

Flowchart



Example 1:

PHP program to print array elements using foreach loop.

```
<?php  
    //declare array  
    $season = array ("Summer", "Winter", "Autumn", "Rainy");  
  
    //access array elements using foreach loop  
    foreach ($season as $element) {
```

```
        echo "$element";
        echo "</br>";
    }
?>
```

Example 2:

PHP program to print associative array elements using foreach loop.

```
<?php
//declare array
$employee = array (
    "Name" => "Alex",
    "Email" => "alex_jtp@gmail.com",
    "Age" => 21,
    "Gender" => "Male"
);

//display associative array element through foreach loop
foreach ($employee as $key => $element) {
    echo $key . " : " . $element;
    echo "</br>";
}
?>
```

Example 3:

Multi-dimensional array

```
<?php
//declare multi-dimensional array
$a = array();
$a[0][0] = "Alex";
$a[0][1] = "Bob";
```

```
$a[1][0] = "Camila";  
$a[1][1] = "Denial";
```

```
//display multi-dimensional array elements through foreach loop
```

```
foreach ($a as $e1) {  
    foreach ($e1 as $e2) {  
        echo "$e2\n";  
    }  
}  
?>
```

Example 4:

Dynamic array

```
<?php  
    //dynamic array  
    foreach (array ('j', 'a', 'v', 'a', 't', 'p', 'o', 'i', 'n', 't') as $elements) {  
        echo "$elements\n";  
    }  
?>
```

PHP While Loop

PHP while loop can be used to traverse set of code like for loop. The while loop executes a block of code repeatedly until the condition is FALSE. Once the condition gets FALSE, it exits from the body of loop.

It should be used if the number of iterations is not known.

The while loop is also called an Entry control loop because the condition is checked before entering the loop body. This means that first the condition is checked. If the condition is true, the block of code will be executed.

Syntax

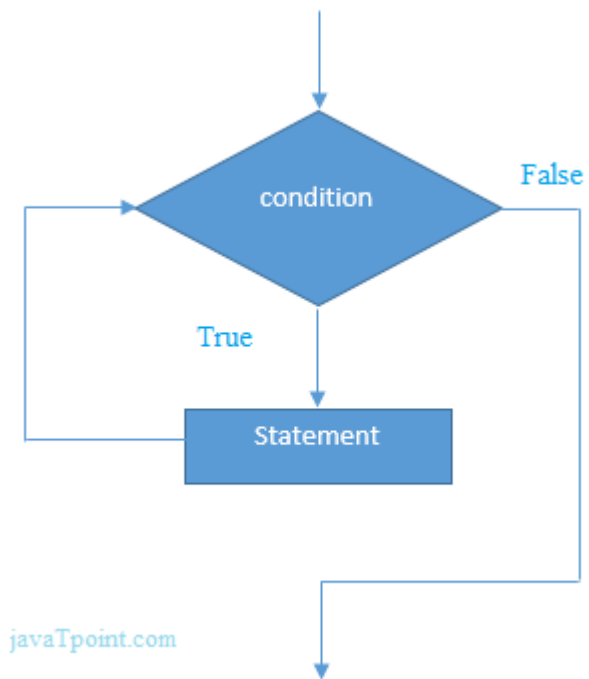
```
while(condition){  
  //code to be executed  
}
```

Alternative Syntax

```
while(condition):  
  //code to be executed
```

```
endwhile;
```

PHP While Loop Flowchart



PHP While Loop Example

```
<?php  
$n=1;  
while($n<=10){  
  echo "$n<br/>";  
  $n++;  
} ?>
```

Alternative Example

```
<?php
$n=1;
while($n<=10):
echo "$n<br/>";
$n++;
endwhile;
?>
```

Example

Below is the example of printing alphabets using while loop.

```
<?php
$i = 'A';
while ($i < 'H') {
    echo $i;
    $i++;
    echo "</br>";
}
?>
```

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

```
<?php
```

```
$i=1;
while($i<=3){
    $j=1;
    while($j<=3){
        echo "$i $j<br/>";
        $j++;
    }
    $i++;
}
?>
```

PHP Infinite While Loop

If we pass TRUE in while loop, it will be an infinite loop.

Syntax

```
while(true) {
    //code to be executed
}
```

Example

```
<?php
while (true) {
    echo "Hello Javatpoint!";
    echo "</br>";
}
}
```

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.

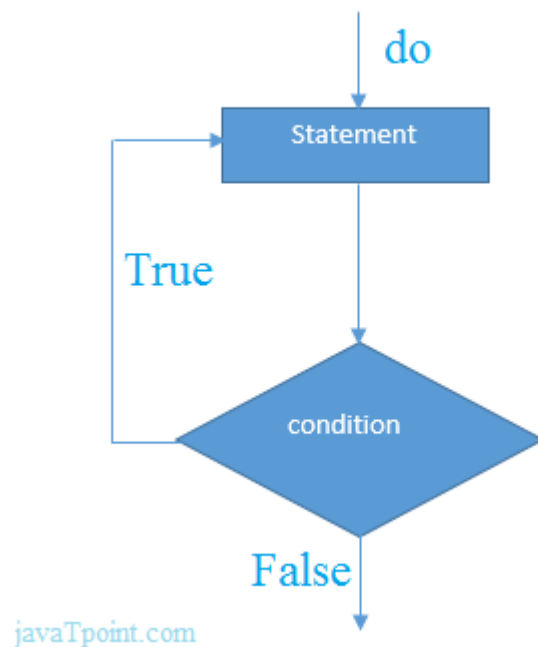
The PHP do-while loop is used to execute a set of code of the program several times. If you have to execute the loop at least once and the number of iterations is not even fixed, it is recommended to use the do-while loop. It executes the code at least one time always because the condition is checked after executing the code.

The do-while loop is very much similar to the while loop except the condition check. The main difference between both loops is that while loop checks the condition at the beginning, whereas do-while loop checks the condition at the end of the loop.

Syntax

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

Flowchart



Example

```
<?php  
$n=1;  
do{
```

```
echo "$n<br/>";
$n++;
}while($n<=10);
?>
```

Example

A semicolon is used to terminate the do-while loop. If you don't use a semicolon after the do-while loop, it is must that the program should not contain any other statements after the do-while loop. In this case, it will not generate any error.

```
<?php
    $x = 5;
    do {
        echo "Welcome to javatpoint! </br>";
        $x++;
    } while ($x < 10);
?>
```

Example

The following example will increment the value of \$x at least once. Because the given condition is false.

```
<?php
    $x = 1;
    do {
        echo "1 is not greater than 10.";
        echo "</br>";
        $x++;
    } while ($x > 10);
    echo $x;
?>
```

Difference between while and do-while loop

while Loop	do-while loop
The while loop is also named as entry control loop.	The do-while loop is also named as exit control loop.
The body of the loop does not execute if the condition is false.	The body of the loop executes at least once, even if the condition is false.
Condition checks first, and then block of statements executes.	Block of statements executes first and then condition checks.
This loop does not use a semicolon to terminate the loop.	Do-while loop use semicolon to terminate the loop

PHP Break

PHP break statement breaks the execution of the 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.

The break keyword immediately ends the execution of the loop or switch structure. It breaks the current flow of the program at the specified condition and program control resumes at the next statements outside the loop.

The break statement can be used in all types of loops such as while, do-while, for, foreach loop, and also with switch case.

Syntax

```
jump statement;  
break;
```

Flowchart

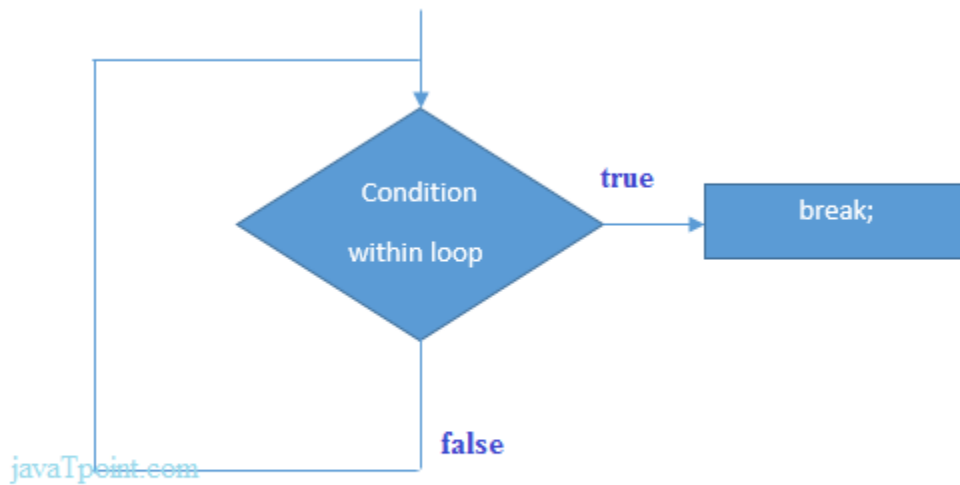


Figure: Flowchart of break statement

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.

```
<?php
for($i=1;$i<=10;$i++){
echo "$i <br/>";
if($i==5){
break;
}
}
?>
```

PHP Break: inside inner loop

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

```
<?php
```

```
for($i=1;$i<=3;$i++){  
  for($j=1;$j<=3;$j++){  
    echo "$i $j<br/>";  
    if($i==2 && $j==2){  
      break;  
    }  
  }  
}  
?>
```

PHP Break: inside switch statement

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

```
<?php  
$num=200;  
switch($num){  
  case 100:  
    echo("number is equals to 100");  
    break;  
  case 200:  
    echo("number is equal to 200");  
    break;  
  case 50:  
    echo("number is equal to 300");  
    break;  
  default:  
    echo("number is not equal to 100, 200 or 500");  
  }  
?>
```


PHP Break: with array of string

```
<?php
//declare an array of string
$number = array ("One", "Two", "Three", "Stop", "Four");
foreach ($number as $element) {
if ($element == "Stop") {
break;
}
echo "$element </br>";
}
?>
```

You can see in the above output, after getting the specified condition true, break statement immediately ends the loop and control is came out from the loop.

PHP Break: switch statement without break

It is not essential to break out of all cases of a switch statement. But if you want that only one case to be executed, you have to use break statement.

```
<?php
$car = 'Mercedes Benz';
switch ($car) {
default:
echo '$car is not Mercedes Benz<br>';
case 'Orange':
echo '$car is Mercedes Benz';
}
?>
```

PHP Break: using optional argument

The break accepts an optional numeric argument, which describes how many nested structures it will exit. The default value is 1, which immediately exits from the enclosing structure.

```
<?php
$i = 0;
while (++$i) {
    switch ($i) {
        case 5:
            echo "At matched condition i = 5<br />\n";
            break 1; // Exit only from the switch.
        case 10:
            echo "At matched condition i = 10; quitting<br />\n";
            break 2; // Exit from the switch and the while.
        default:
            break;
    }
}
}??>
```

- After declaring a variable, it can be reused throughout the code.
- Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP is given below:

1. \$variablename=value;

Rules for declaring PHP variable:

- A variable must start with a dollar (\$) sign, followed by the variable name.
- It can only contain alpha-numeric character and underscore (A-z, 0-9, _).
- A variable name must start with a letter or underscore (_) character.
- A PHP variable name cannot contain spaces.
- One thing to be kept in mind that the variable name cannot start with a number or special symbols.

- PHP variables are case-sensitive, so \$name and \$NAME both are treated as different variable.

PHP Variable: Declaring string, integer, and float

Let's see the example to store string, integer, and float values in PHP variables.

File: variable1.php

```
<?php
$str="hello string";
$x=200;
$y=44.6;
echo "string is: $str <br/>";
echo "integer is: $x <br/>";
echo "float is: $y <br/>";
?>
```

PHP Variable: Sum of two variables

File: variable2.php

```
<?php
$x=5;
$y=6;
$z=$x+$y;
echo $z;
?>
```

PHP Variable: case sensitive

In PHP, variable names are case sensitive. So variable name "color" is different from Color, COLOR, COLor etc.

File: variable3.php

```
<?php
$color="red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>
```

PHP Variable: Rules

PHP variables must start with letter or underscore only.

PHP variable can't be start with numbers and special symbols.

File: variablevalid.php

```
<?php
$a="hello";//letter (valid)
$_b="hello";//underscore (valid)

echo "$a <br/> $_b";
?>

<?php
$4c="hello";//number (invalid)
$*d="hello";//special symbol (invalid)

echo "$4c <br/> $*d";
?>
```

PHP: Loosely typed language

PHP is a loosely typed language, it means PHP automatically converts the variable to its correct data type.