UNIT-1

INTORDUTION TO PHP

INTRODUCTION TO PHP:

What is PHP

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.

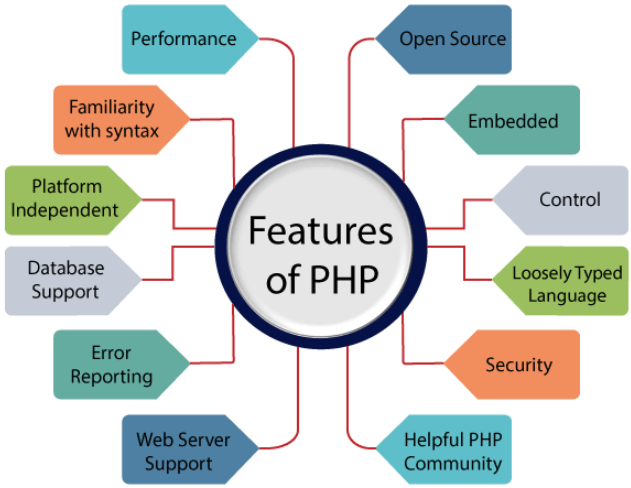
Why use PHP

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.

How to run PHP code

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
?>
```

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.

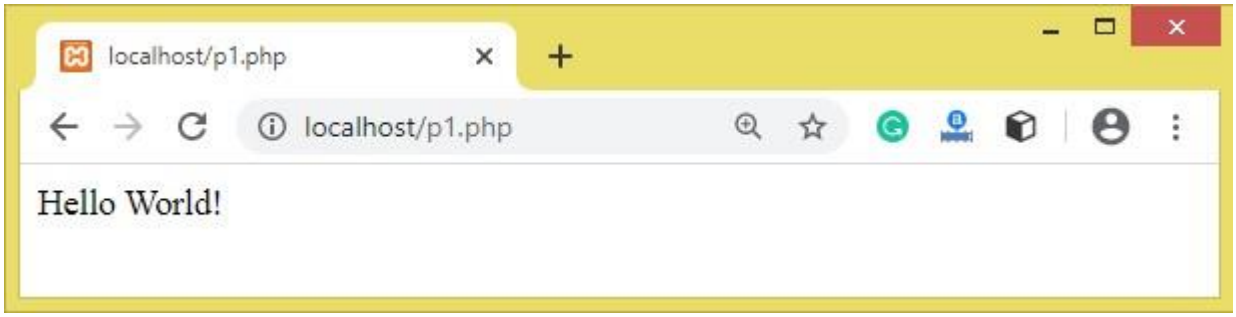
```
<?php
echo "Hello World!";
?>
```

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)

INSTALLATION AND CONFIGURATION

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.

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

Download and Install LAMP Server Click me to download LAMP server

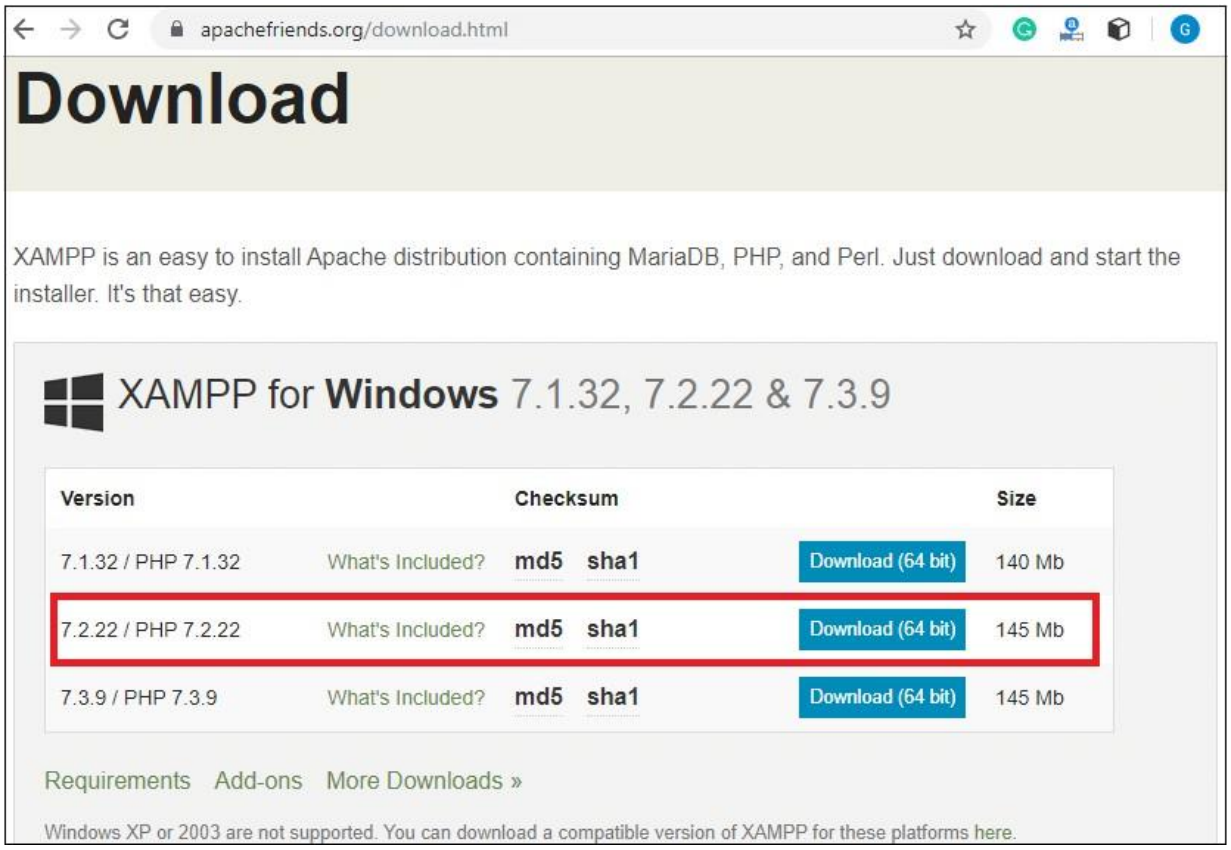
Download and Install MAMP Server Click me to download MAMP server

Download and Install XAMPP Server
Click me to download XAMPP server

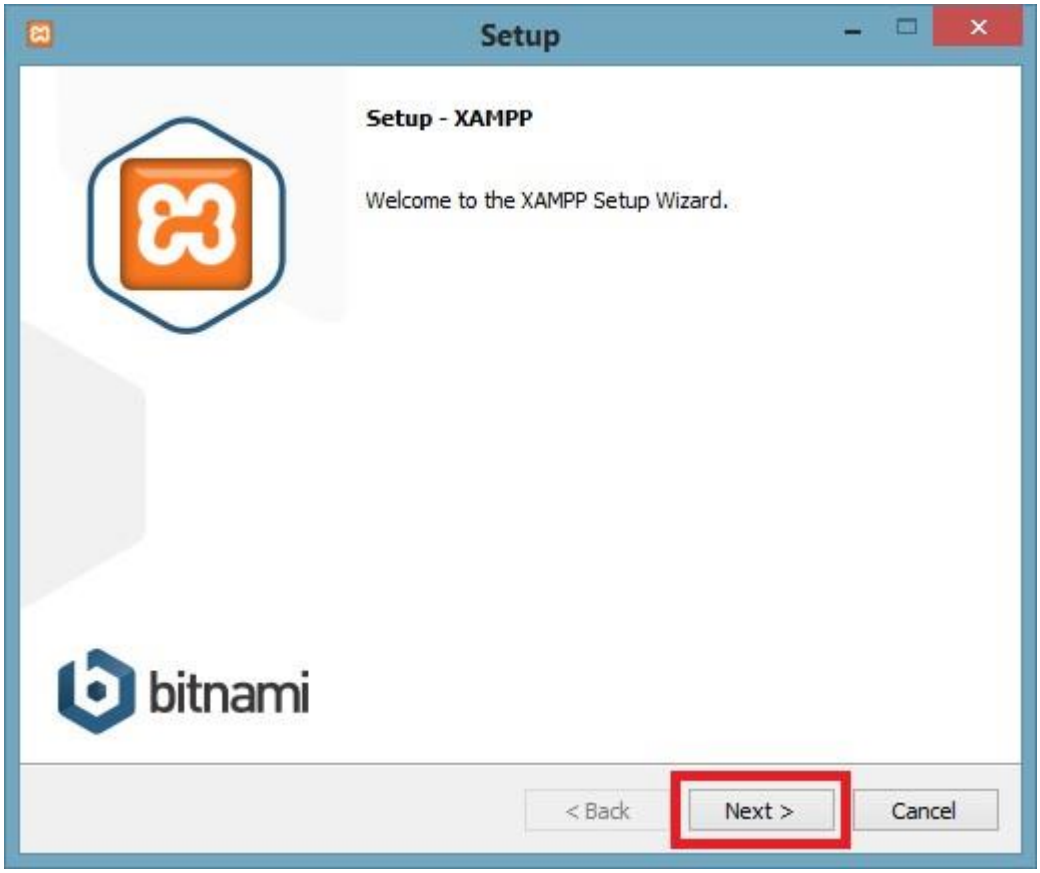
How to install XAMPP server on windows

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.

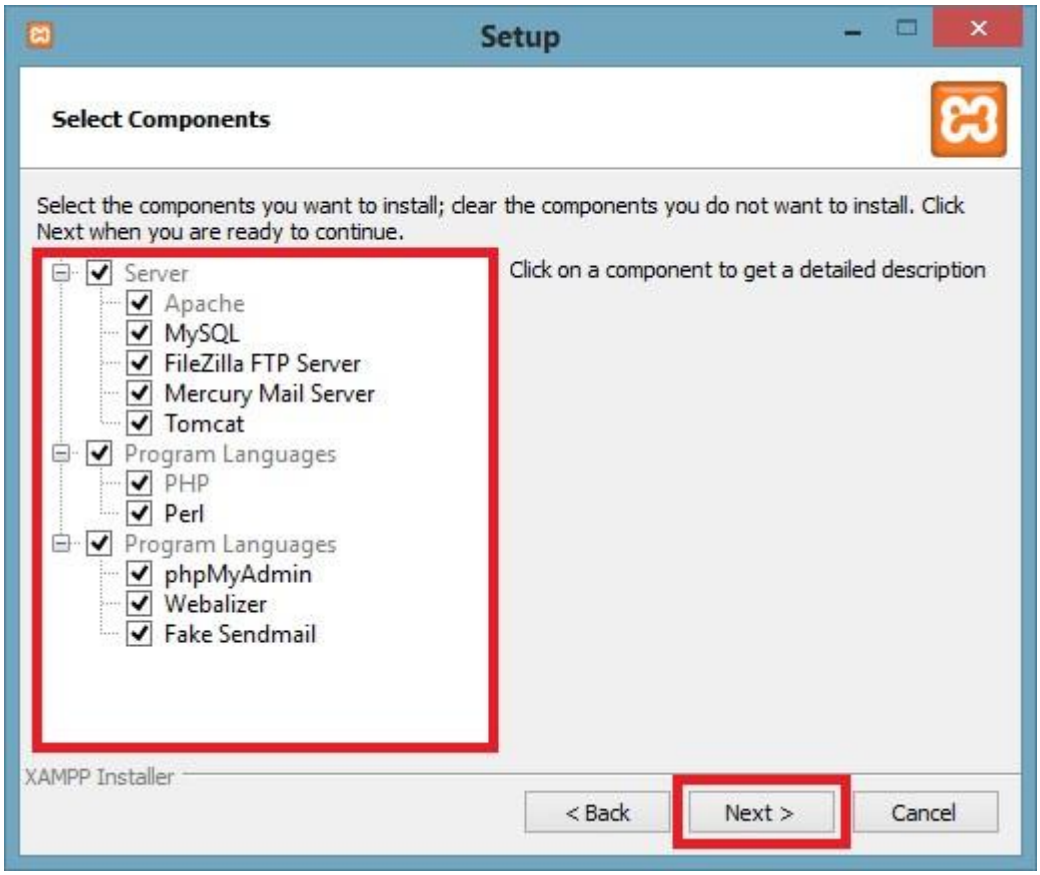
Step 1: Click on the above link provided to download the **XAMPP server** according to your window requirement.



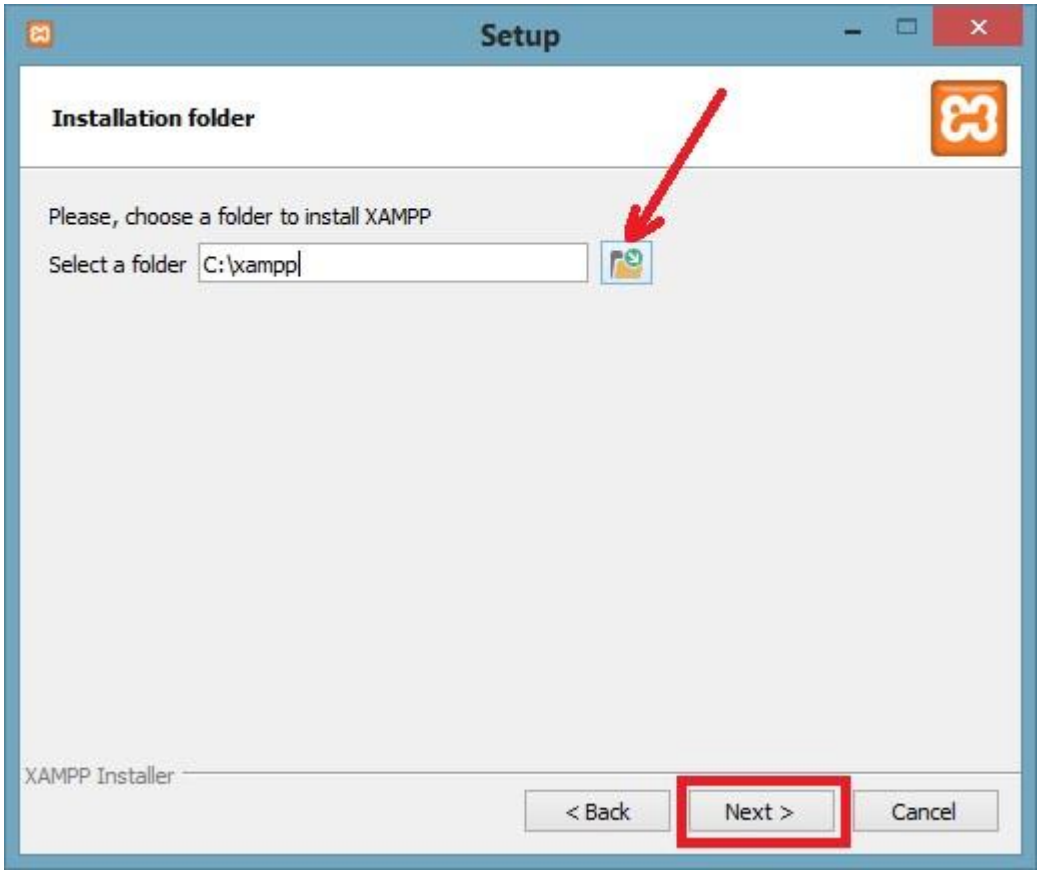
Step 2: After downloading XAMPP, double click on the downloaded file and allow XAMPP to make changes in your system. A window will pop-up, where you have to click on the **Next** button.



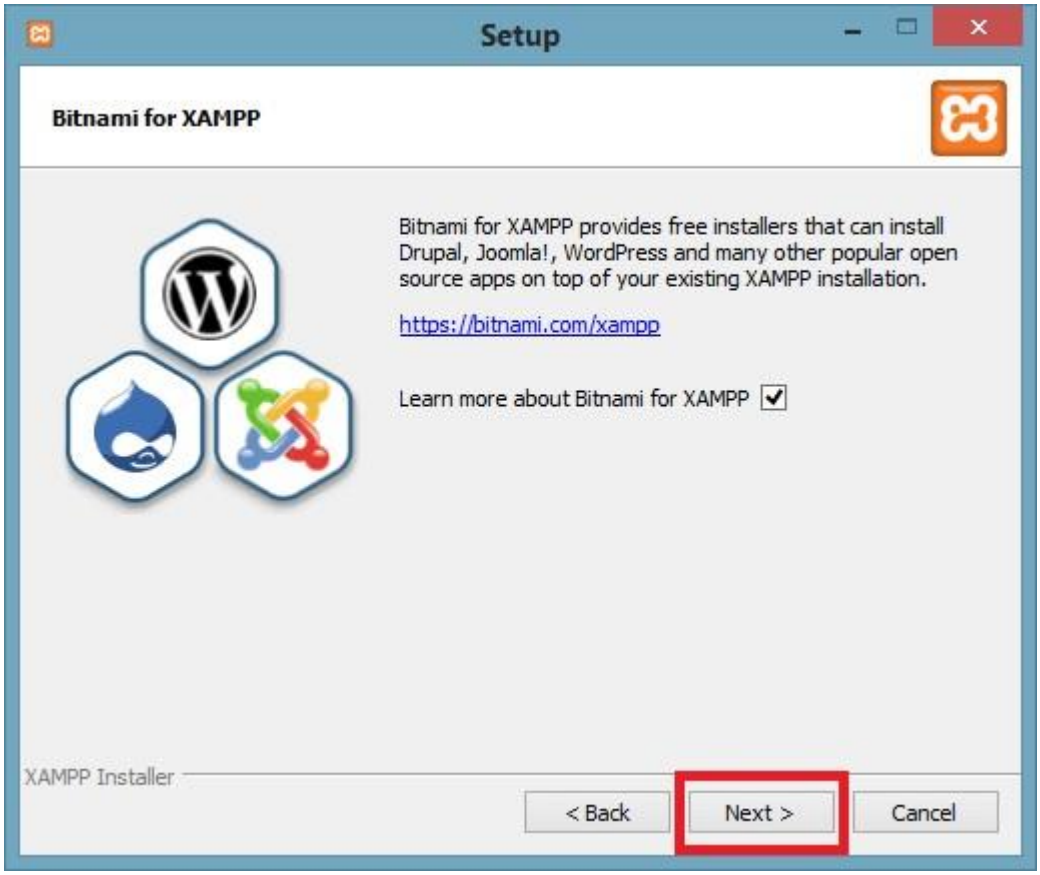
Step 3: Here, select the components, which you want to install and click **Next**.



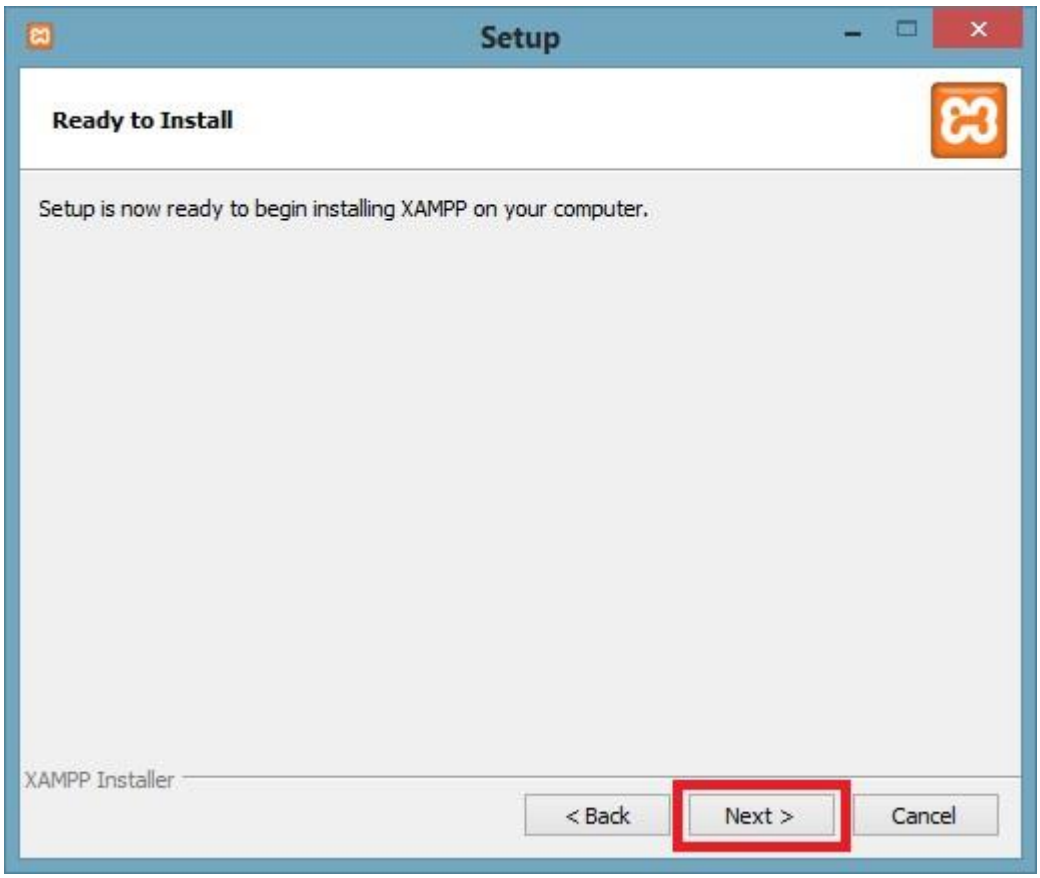
Step 4: Choose a folder where you want to install the XAMPP in your system and click **Next**.



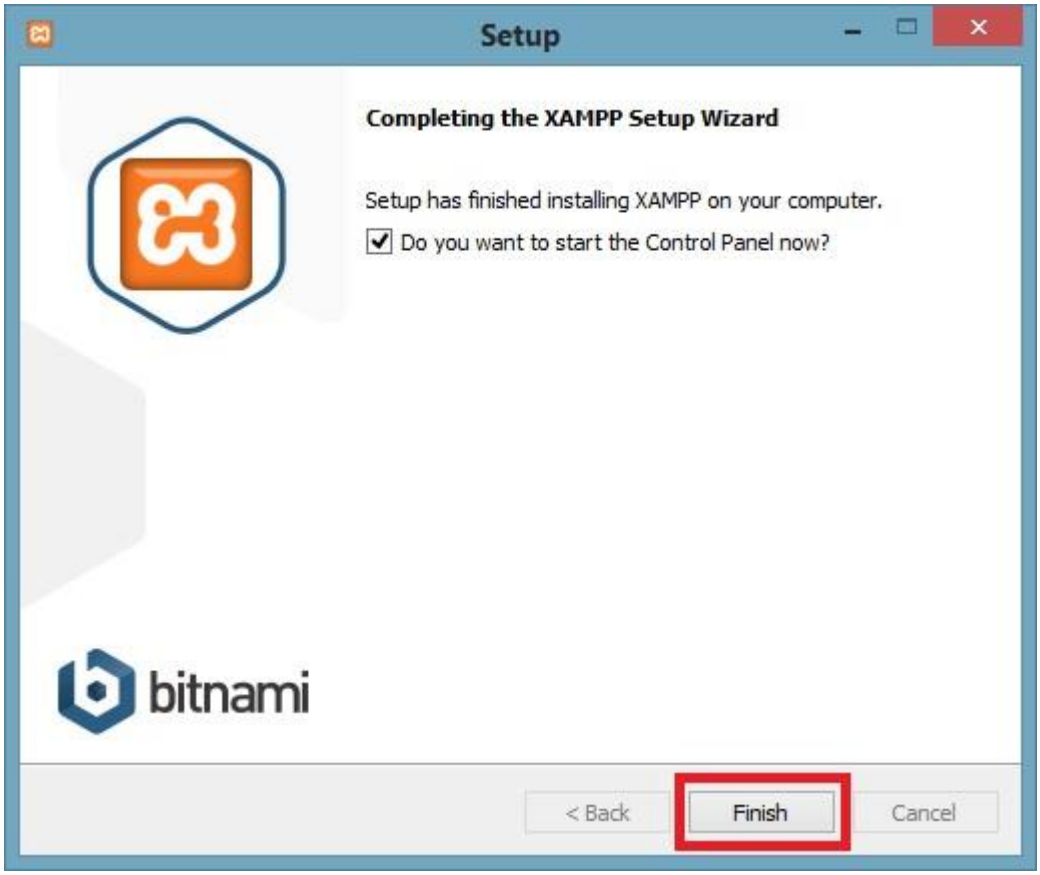
Step 5: Click **Next** and move ahead.



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



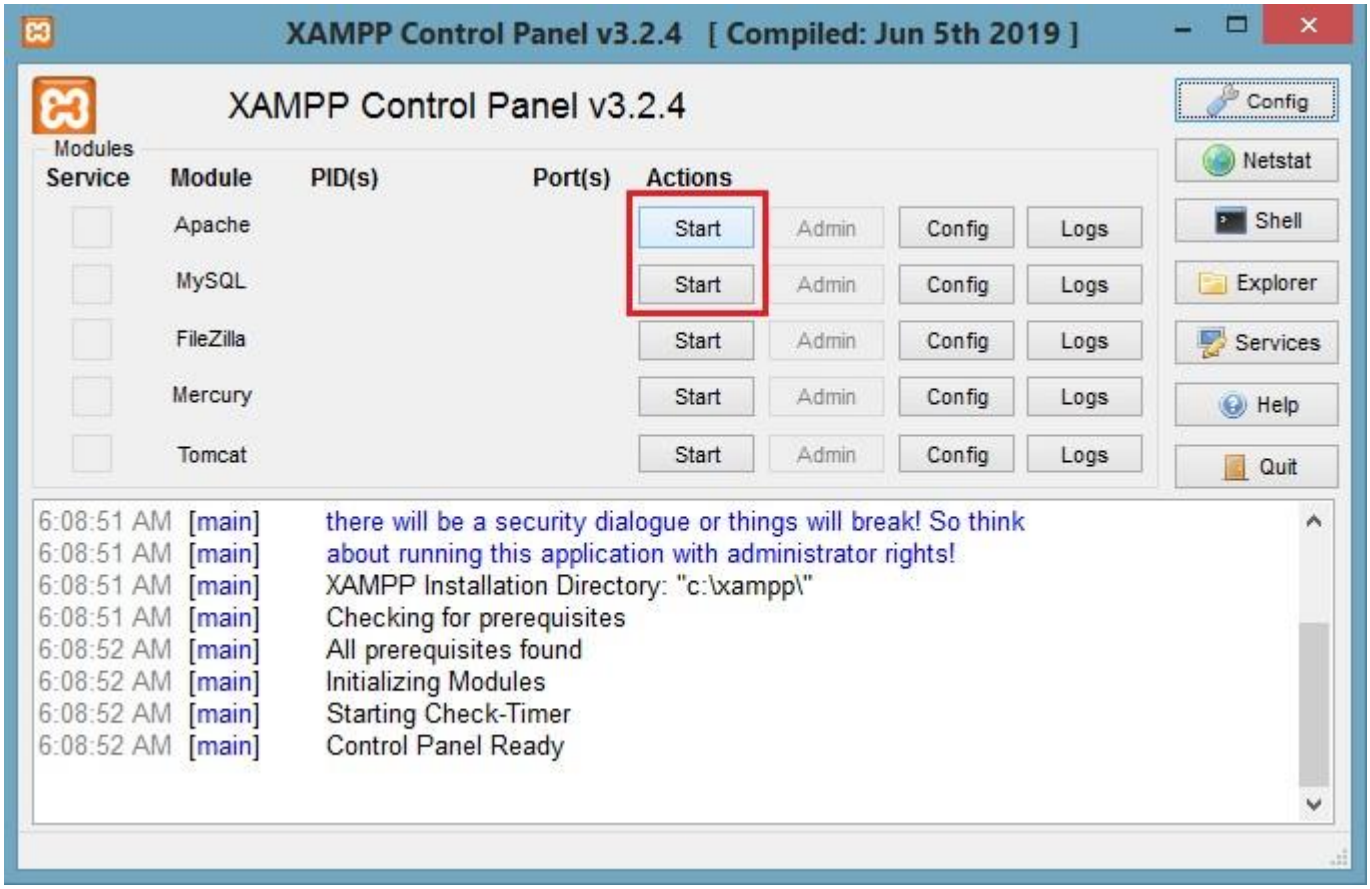
Step 7: A finish window will display after successful installation. Click on the **Finish** button.



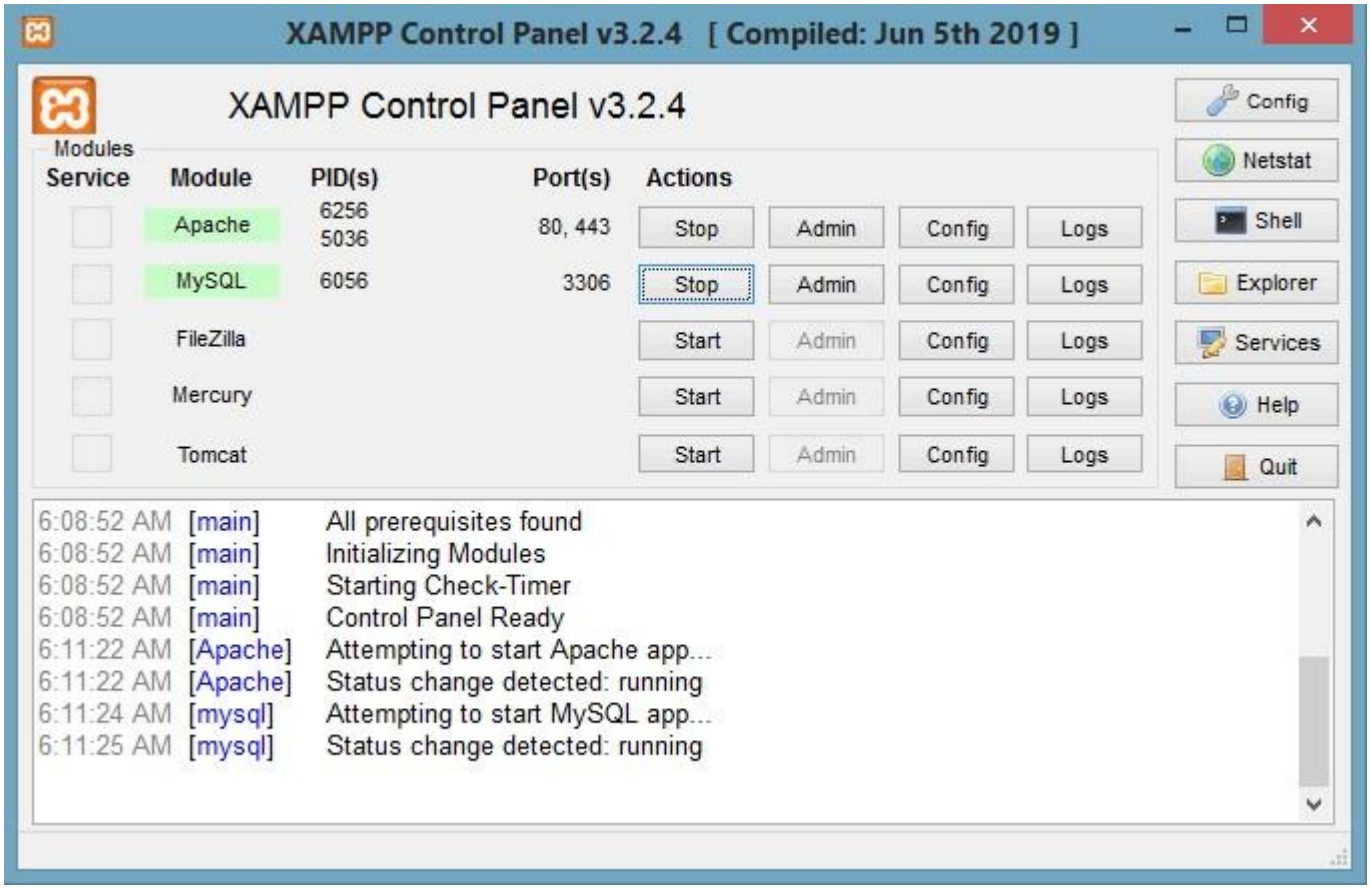
Step 8: Choose your preferred language.

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.



Embedding PHP in Web Pages Although it is possible to write and run standalone PHP programs, most PHP code is embedded in HTML or XML files. This is, after all, why it was created in the first place. Processing such documents involves replacing each chunk of PHP source code with the output it produces when executed. Because a single file contains PHP and non-PHP source code, we need a way to identify the regions of PHP code to be executed. PHP provides four different ways to do this. As you'll see, the first, and preferred, method looks like XML. The second method looks like SGML. The third method is based on ASP tags. The fourth method uses the standard HTML

XML Style Because of the advent of the eXtensible Markup Language (XML) and the migration of HTML to an XML language (XHTML), the currently preferred technique for embedding PHP uses XML-compliant tags to denote PHP instructions. docstore.mik.ua/orelly/webprog/php/ch02_07.html

Look, ma! It's my first PHP program:

How cool is that?

docstore.mik.ua/orelly/webprog/php/ch02_07.html

2.7.3. ASP Style Because neither the SGML nor XML tag style is strictly legal HTML,[3] some HTML editors do not parse it correctly for color syntax highlighting, context-sensitive help, and other such niceties. Some will even go so far as to helpfully remove the "offending" code for you. [3]Mostly because you are not allowed to use a > inside your tags if you wish to

be compliant, but who wants to write code like if($a > 5)...? However, many of these same HTML editors recognize another mechanism (no more legal than PHP's) for embedding code—that of Microsoft's Active Server Pages (ASP). Like PHP, ASP is a method for embedding server-side scripts within documents. If you want to use ASP-aware tools to edit files that contain embedded PHP, you can use ASP-style tags to identify PHP regions. The ASP-style tag is the same as the SGML-style tag, but with % instead of ?: <% echo "Hello, world"; %> In all other ways, the ASP-style tag works the same as the SGML-style tag. docstore.mik.ua/orelly/webprog/php/ch02_07.html

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:

```
1. <!DOCTYPE>
2. <html>
3. <body>
4. <?php
5. echo "Hello world using echo </br>";
6. ECHO "Hello world using ECHO </br>";
7. EcHo "Hello world using EcHo </br>";
8. ?>
9. </body>
10. </html>
```

How to use PHP in Html

If we want to use PHP in the Html document, then we have to follow the steps which are given below. Using these simple steps, we can easily add the PHP code.

Step 1: Firstly, we have to type the Html code in any text editor or open the existing Html file in the text editor in which we want to use the PHP.

```
1. <!Doctype Html>
2. <Html>
3. <Head>
4. <Title>
```

5. Use a Php in Html
6. **</Title>**
7. **</Head>**
8. **<Body>**
9. **</Body>**
10. **</Html>**

Step 2: Now, we have to place the cursor in any tag of the <body> tag where we want to add the code of PHP. And, then we have to type the start and end tag of PHP.

1. **<h1>**
2. **<?php ?>**
3. **</h1>**

Step 3: After then, we have to type the code of PHP between the tags of PHP.

1. **<h1>**
2. **<?php**
3. echo "Hii User!! You are at JavaTpoint Site"
4. **?>**
5. **</h1>**

Step 4: When we successfully add the PHP code, then we have to save the Html file and then run the file in the browser.

1. <!Doctype Html>
2. **<Html>**
3. **<Head>**
4. **<Title>**
5. Use a Php in Html
6. **</Title>**
7. **</Head>**
8. **<Body>**
9. **<h1><?php** echo "Hii User!! You are at JavaTpoint Site" **?></h1>**
10. **</Body>**
11. **</Html>**

WHITE SPACE

White Space is a property defined in CSS (Cascading Style Sheets) that enable the user to control the text wrapping and white spaces inside an element of the website. It can take several types of values depending upon the user's need.

Syntax of White Space Property

The syntax of implementing the white-space property in CSS is similar to any other property that is property_name: value;

1. white-space: value;

Several values that can be passed to the white-space property are:

1. white-space: normal;
2. white-space: nowrap;
3. white-space: pre;
4. white-space: pre-wrap;
5. white-space: pre-line;
6. white space: break-spaces;

The above values assigned to the property are keyword values; they are specific to a particular element in the website.

1. white-space: inherit;
2. white-space: initial;
3. white-space: revert;
4. white-space: revert-layer;
5. white-space: unset;

The values mentioned above are global; they are used to maintain consistency among all the website elements.

White-space property is a single value, which means you can use only one value simultaneously.

Functions Performed by Different values in white-space

- **normal:** This value collapses a sequence of white spaces. The new line character is also considered white space. If the lines are broken, then the line must fill the line boxes.
- **nowrap:** Using this value, it treats white space between the element as normal, but it suppresses the line breaks or the text wrapping in the source.
- **pre:** It does not collapse the white spaces they are preserved. The lines are only discontinued by the new line characters and break tags mentioned in the website's source.

- **pre-wrap:** The sequences of white space are preserved. It is similar to pre as the sequence of white-space are preserved. The lines are either broken on the new line or due to the break tag in the source code. The only difference is that it requires filling the line boxes too.
- **pre-line:** The value collapses the white spaces in the content. The lines are broken due to the new line characters or the break tag. It is necessary to fill the line boxes.
- **break-spaces:** The behavior of break spaces is somewhat identical to pre-wrap; the key difference between both the values are as follows:
- Any sequence of preserved white space will occupy space. It also includes the end of a lin

SENDING DATA TO WEB BROWSER

PHP - GET & POST Methods

There are two ways the browser client can send information to the web server.

- The GET Method
- The POST Method

Before the browser sends the information, it encodes it using a scheme called URL encoding. In this scheme, name/value pairs are joined with equal signs and different pairs are separated by the ampersand.

name1=value1&name2=value2&name3=value3

Spaces are removed and replaced with the + character and any other nonalphanumeric characters are replaced with a hexadecimal values. After the information is encoded it is sent to the server.

The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

http://www.test.com/index.htm?name1=value1&name2=value2

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.

- The PHP provides **$_GET** associative array to access all the sent information using GET method.

Try out following example by putting the source code in test.php script.

```
<?php if( $_GET["name"] || $_GET["age"] )
{ echo "Welcome ". $_GET['name']. "<br
/>"; echo "You are ". $_GET['age']. " years
old.";

exit();
}
?>
<html>
<body>

<form action = "<?php $_PHP_SELF ?>" method = "GET">
Name: <input type = "text" name = "name" />
Age: <input type = "text" name = "age" />
<input type = "submit" />
</form>

</body>
</html>
```

It will produce the following result −

The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using POST method.

Try out following example by putting the source code in test.php script.

```
<?php
if( $_POST["name"] || $_POST["age"] ) {
```

```
if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) { die
("invalid name and name should be alpha");
}
echo "Welcome ". $_POST['name']. "<br />";
echo "You are ". $_POST['age']. " years old.";

exit();
}
?>
<html>
<body>

<form action = "<?php $_PHP_SELF ?>" method = "POST">
Name: <input type = "text" name = "name" />
Age: <input type = "text" name = "age" />
<input type = "submit" />
</form>

</body>
</html>
```

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.

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

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.";
?>
```

Output:

```
This condition is TRUE.
```

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 2,147,483,648 and 2,147,483,647 i.e., 2^31 to 2^31.

Example:

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

Output:

```
Decimal number: 34
Octal number: 163
HexaDecimal number: 69
```

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

Output:

```
Addition of floating numbers: 73.812
```

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
<?php
$company = "Javatpoint";
```

```
3. //both single and double quote statements will treat different
4. echo "Hello $company";
5. echo "</br>";
6. echo 'Hello $company';
7. ?>
```

Output:

```
Hello Javatpoint
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:

```
1. <?php
2. $bikes = array ("Royal Enfield", "Yamaha", "KTM");
3. var_dump($bikes);   //the var_dump() function returns the datatype and values
4. echo "</br>";
5. echo "Array Element1: $bikes[0] </br>"; 6. echo "Array Element2: $bikes[1] </br>";
7. echo "Array Element3: $bikes[2] </br>";
8. ?> Output:
```

```
array(3) { [0]=> string(13) "Royal Enfield" [1]=> string(6) "Yamaha" [2]=>
string(3) "KTM" }
Array Element1: Royal Enfield
Array Element2: Yamaha Array Element3: KTM
```

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:

```
1. <?php
2. class bike {
3. function model() {
4. $model_name = "Royal Enfield";
```

```
5. echo "Bike Model: " .$model_name;
6. }
7. }
8. $obj = new bike();
9. $obj -> model();
10. ?>
```

Output:

```
Bike Model: Royal Enfield
```

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:

```
1. <?php
2. $nl = NULL;
3. echo $nl;   //it will not give any output
4. ?>
```

PHP KEYWORDS

PHP has a set of keywords that are reserved words which cannot be used as function names, class names or method names. Prior to PHP 7, these keywords could not be used as class property names either:

Keyword	Description
abstract	Declare a class as abstract
and	A logical operator
as	Used in the foreach loop
break	Break out of loops and switch statements
callable	A data type which can be executed as a function
case	Used in the switch conditional
catch	Used in the try..catch statement
class	Declare a class
clone	Create a copy of an object
const	Define a class constant

continue	Jump to the next iteration of a loop

declare	Set directives for a block of code
default	Used in the switch statement
do	Create a do...while loop
echo	Output text
else	Used in conditional statements
elseif	Used in conditional statements
empty	Check if an expression is empty
enddeclare	End a declare block
endfor	End a for block
endforeach	End a foreach block
endif	End an if or elseif block

endswitch	End a switch block

endwhile	End a while block
extends	Extends a class or interface
final	Declare a class, property or method as final
finally	Used in the try...catch statement
fn	Declare an arrow function
for	Create a for loop
foreach	Create a foreach loop
function	Create a function
global	Import variables from the global scope
goto	Jump to a line of code
if	Create a conditional statement

implements	Implement an interface

include	Embed code from another file
include_once	Embed code from another file
instanceof	Test an object's class
insteadof	Resolve conflicts with traits
interface	Declare an interface
isset	Check if a variable exists and is not null
list	Assigns array elements into variables
namespace	Declares a namespace
new	Creates an object
or	A logical operator
print	Output text

private	Declare a property, method or constant as private

protected	Declare a property, method or constant as protected
public	Declare a property, method or constant as public
require	Embed code from another file
require_once	Embed code from another file
return	Exit a function and return a value
static	Declare a property or method as static
switch	Create a switch block
throw	Throw an exception
trait	Declare a trait
try	Create a try...catch structure
unset	Delete a variable or array element

use	Use a namespace

var	Declare a variable
while	Create a while loop or end a do...while loop
xor	A logical operator
yield	Used in generator functions
yield from	Used in generator functions

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;
?>
```

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)
```

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.

PHP VARIABLES

n 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.
- 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/>";
?>
```

Output:

```
string is: hello string
integer is: 200 float
is: 44.6
```

PHP Variable: Sum of two variables

File: variable2.php

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

Output:

```
11
```

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>";
?>
```

Output:

```
My car is red
Notice: Undefined variable: COLOR in C:\wamp\www\variable.php on line 4
My house is
Notice: Undefined variable: coLOR in C:\wamp\www\variable.php on line 5 My boat is
```

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)   3. $_b="hello";//underscore (valid)

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

Output:

```
hello  hello
```

File: variableinvalid.php

```
<?php
$4c="hello";//number (invalid)   3. $*d="hello";//special symbol (invalid)

```

5. echo "$4c
 $*d";
6. ?>

Output:

```
Parse error: syntax error, unexpected '4' (T_LNUMBER), expecting variable
(T_VARIABLE)
 or '$' in C:\wamp\www\variableinvalid.php on line 2
```

PHP EXPRESSION

Introduction

Almost everything in a PHP script is an expression. Anything that has a value is an expression. In a typical assignment statement ($x=100), a literal value, a function or operands processed by operators is an expression, anything that appears to the right of assignment operator (=)

Syntax

$x=100; //100 is an expression
$a=$b+$c; //b+$c is an expression
$c=add($a,$b); //add($a,$b) is an expresson
$val=sqrt(100); //sqrt(100) is an expression

$var=$x!=$y; //$x!=$y is an expression

expression with ++ and -- operators

These operators are called increment and decrement operators respectively. They are unary operators, needing just one operand and can be used in prefix or postfix manner, although with different effect on value of expression

Both prefix and postfix ++ operators increment value of operand by 1 (whereas -- operator decrements by 1). However, when used in assignment expression, prefix makesincremnt/decrement first and then followed by assignment. In case of postfix, assignment is done before increment/decrement

Uses postfix ++ operator

```
<?php
$x=10;
$y=$x++; //equivalent to $y=$x followed by $x=$x+1 echo "x = $x y = $y";



?>
```

Expression with Ternary conditional operator

Ternary operator has three operands. First one is a logical expression. If it is TRU, second operand expression is evaluated otherwise third one is evaluated

```
<?php
$marks=60;
$result= $marks<50 ? "fail" : "pass"; echo $result; ?>
```

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;

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

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

Operator	Name	Example	Explanation

=	Assign	$a = $b	The value of right operand is assigned to the left operand.

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

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 "=".

Bitwise Operators

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

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.

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

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

Incrementing/Decrementing Operators

Operator	Name	Example	Explanation
and	And	$a and $b	Return TRUE if both $a and $b are tru
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 b
!	Not	! $a	Return TRUE if $a is not true
&&	And	$a && $b	Return TRUE if either $a and $b are tru
||	Or	$a || $b	Return TRUE if either $a or $b is true

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

Logical Operators

Operator	Name	Example	Explanation

.	Concatenation	$a . $b	Concatenate both $a and $b

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
+	Union	$a + $y	Union of $a and $b
==	Equality	$a == $b	Return TRUE if $a and $b have same key/value pai
!=	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 pai
!==	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

String Operators

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

Array Operators

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.
.=	Concatenation and Assignment	$a .= $b	First concatenate $a and $b, then assign the concatenated string to $a, e.g. $a = $a . $b

Operator	Name	Example
@	at	@file ('non_existent_file')

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

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.

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.

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.
