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 8.1.4** is the latest version of PHP, which was released on **17 March 2022**. 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.

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.

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.

- o HTML
- o CSS
- o 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.

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.

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.
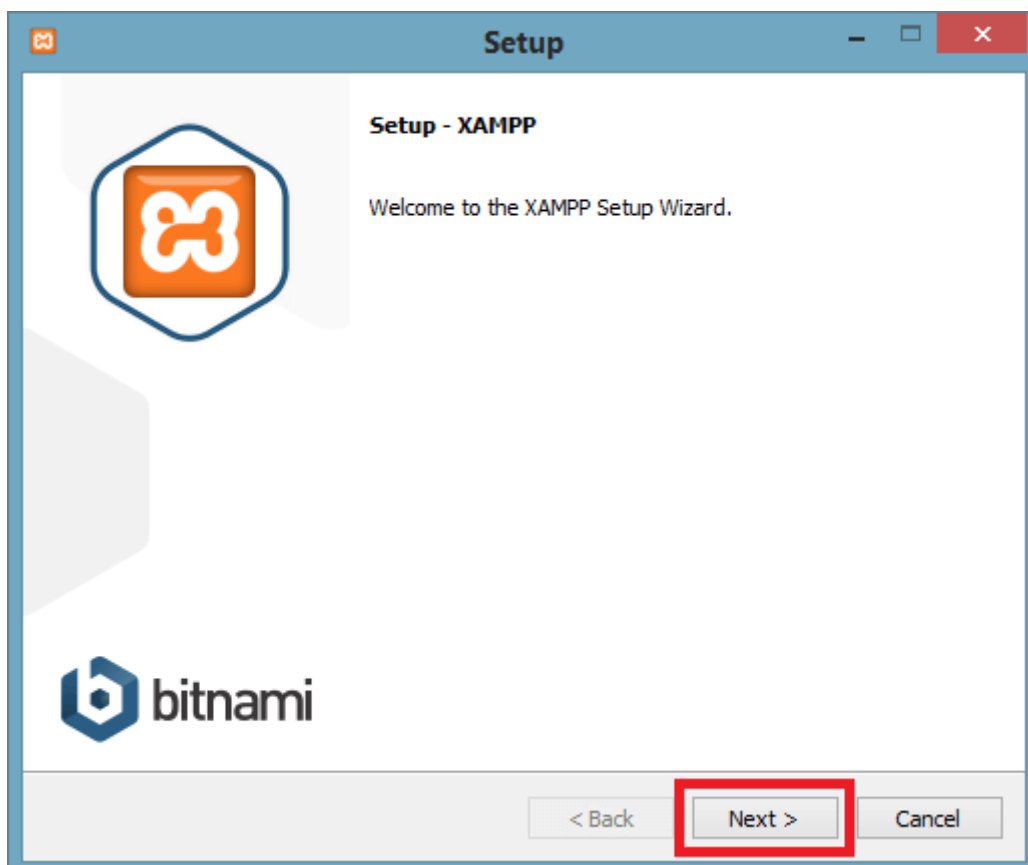
Step 3: Here, select the components, which you want to install and click **Next**.
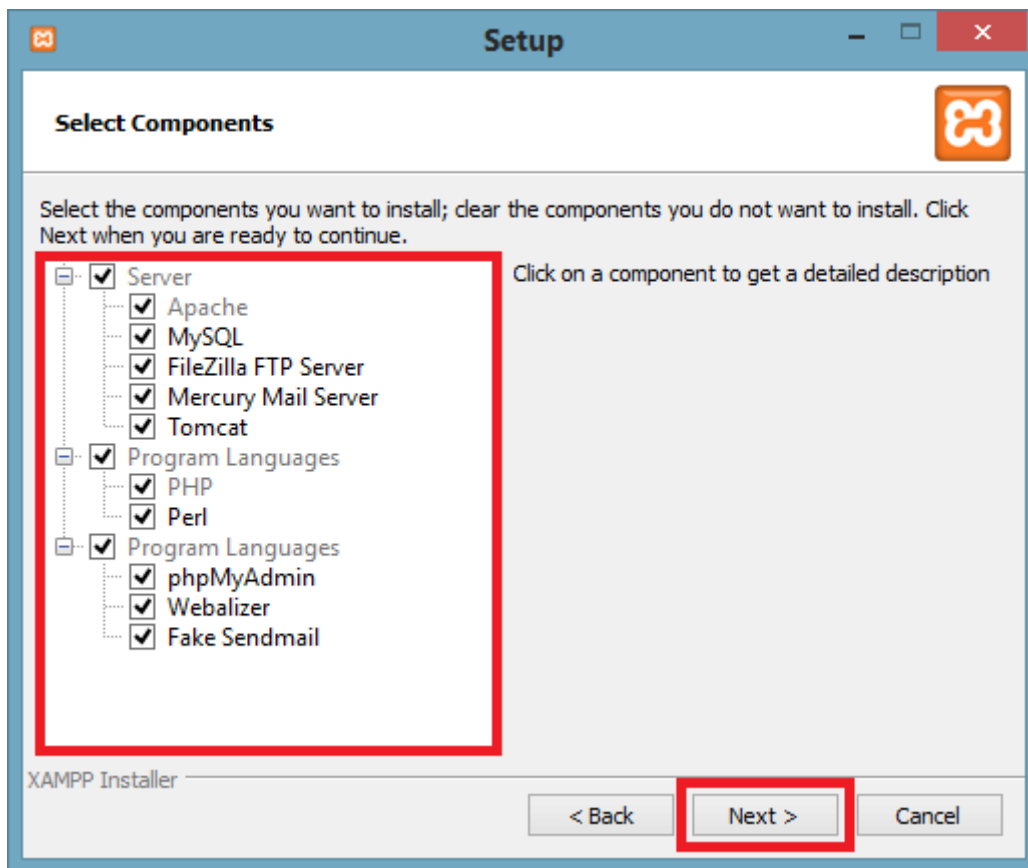
Step 4: Choose a folder where you want to install the XAMPP in your system and click **Next**.
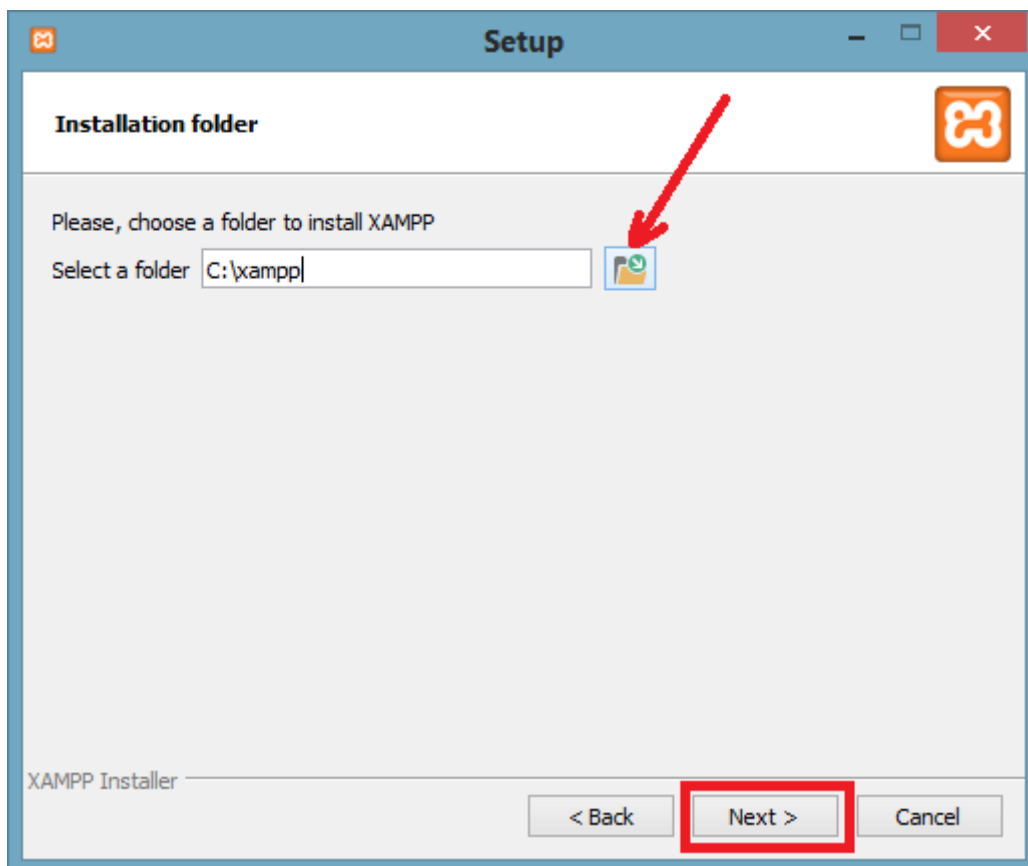
Step 5: Click **Next** and move ahead.
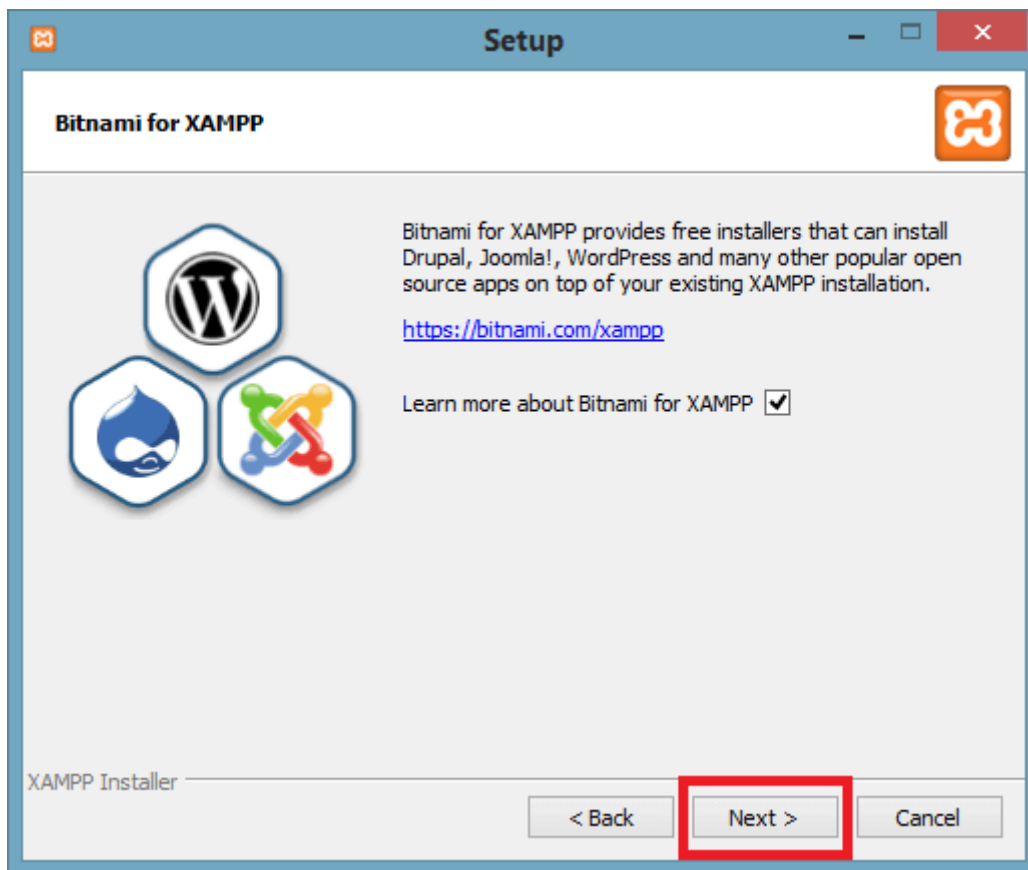
Step 6: XAMPP is ready to install, so click on the **Next** button and install the XAMPP.
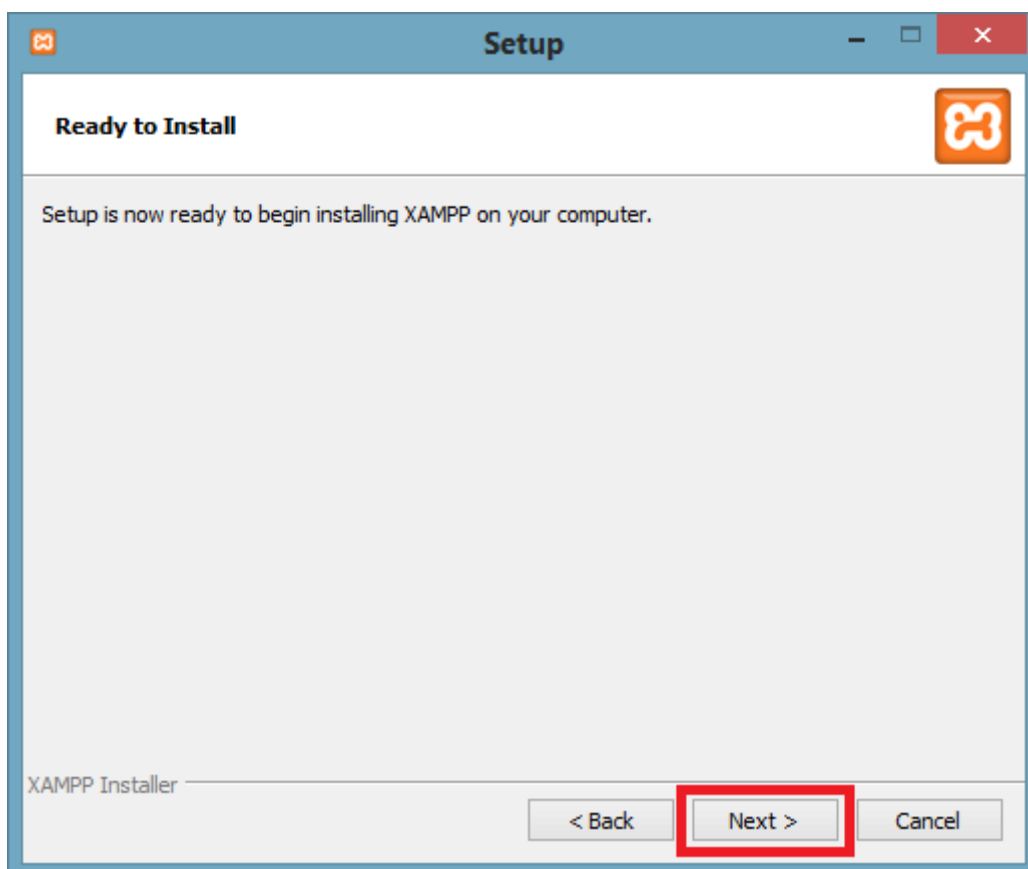
Step 7: A finish window will display after successful installation. Click on the **Finish** button.
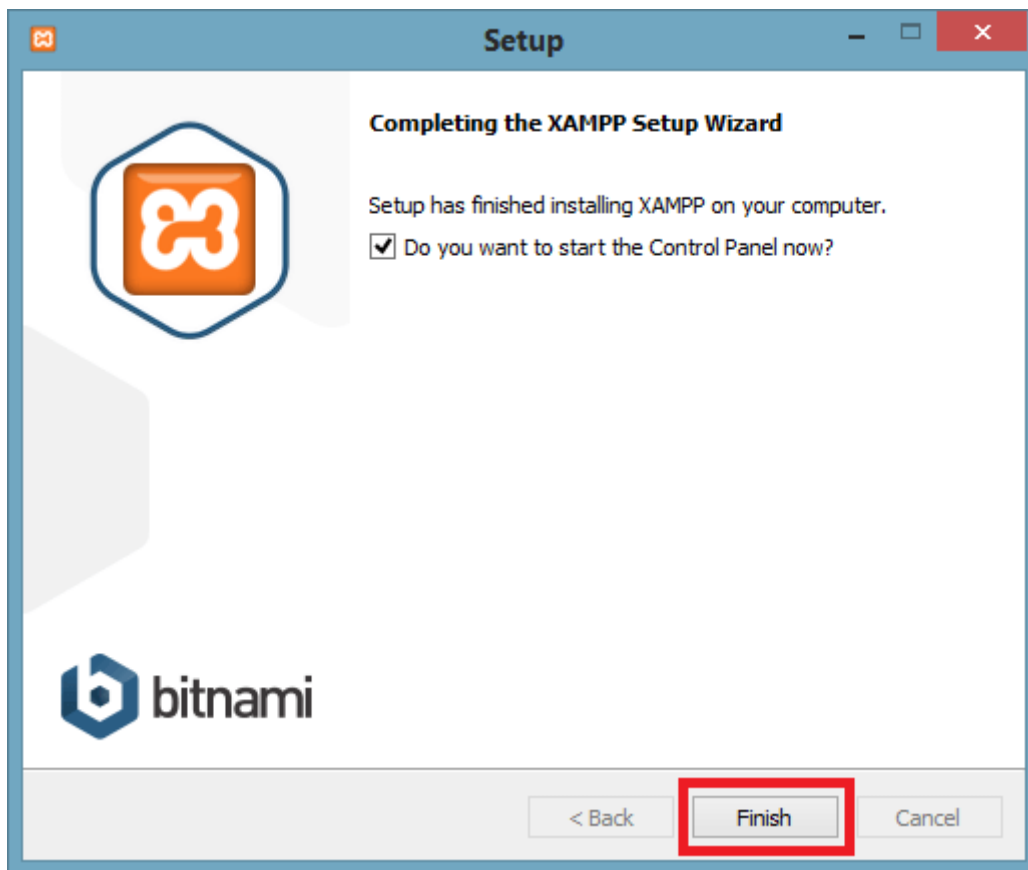
Step 8: Choose your preferred language.
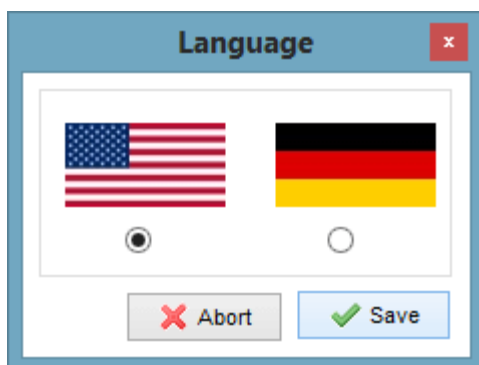
Step 9: XAMPP is ready to use. Start the Apache server and MySQL and run the php program on the localhost.

How to run PHP programs on XAMPP, see in the next tutorial.
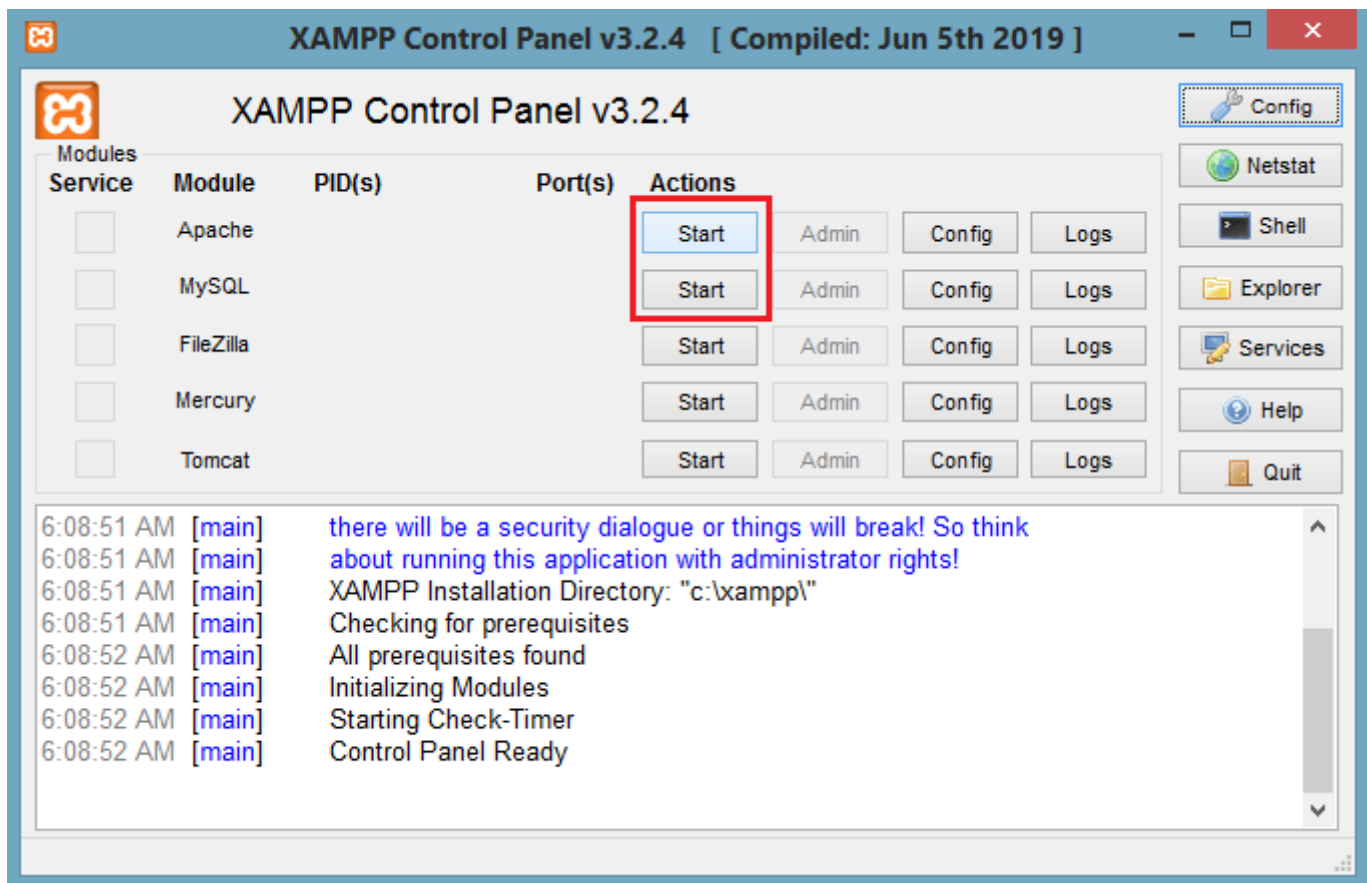
Step 10: If no error is shown, then XAMPP is running successfully.
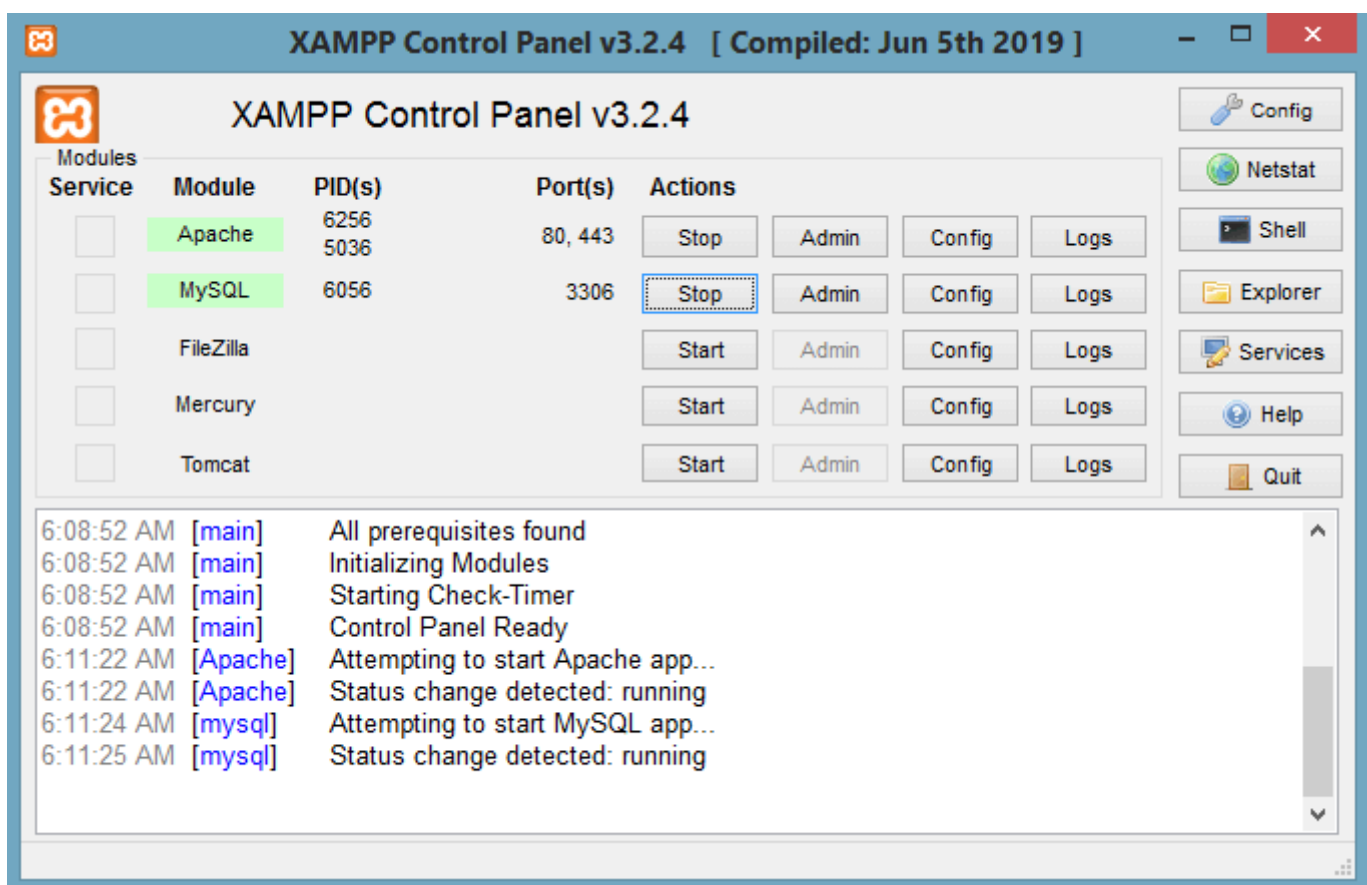
How to run PHP code in XAMPP

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.

Note: PHP statements ends with semicolon (;).

All PHP code goes between the phptag. It starts with <?php and ends with ?>. The syntax of PHP tag is given below:

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

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

File: first.php

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

Output:

Hello First PHP

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

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

Note: PHP program must be saved in the htdocs folder, which resides inside the xampp folder, where you installed the XAMPP. Otherwise it will generate an error - Object not found.

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

Step 4: Now, open the web browser and type ***localhost/mrec/hello.php***

on your browser window.

Step 5: The output for the above **hello.php** program will be shown as the screenshot below:
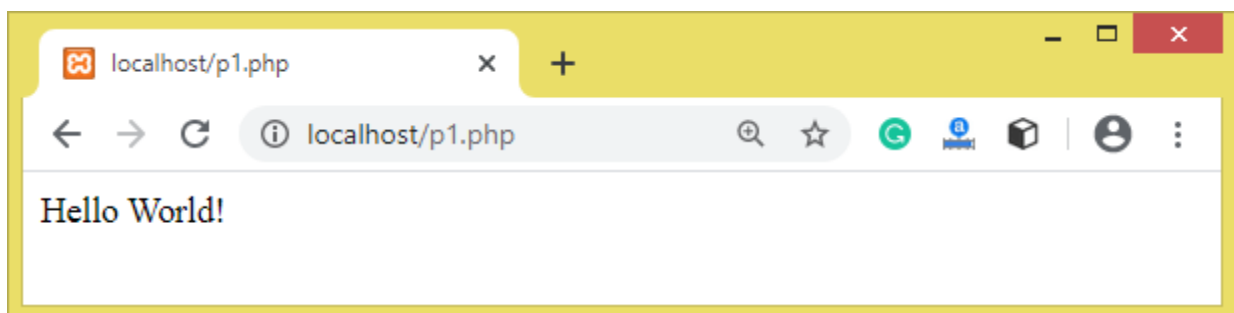
Most of the time, PHP programs run as a web server module. However, PHP can also be run on CLI (Command Line Interface).

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:

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

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

Output:

```
Hello world using echo
Hello world using ECHO
Hello world using EcHo
```

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:

```php
<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>
```

Output:

```
Notice: Undefined variable: ColoR in D:\xampp\htdocs\program\p2.php on line 8
My car is
My dog is black

Notice: Undefined variable: COLOR in D:\xampp\htdocs\program\p2.php on line 10
My Phone is
```

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

Output:

Hello by PHP echo

PHP echo: printing multi line string

File: echo2.php

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

Output:

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

PHP echo: printing escaping characters

File: echo3.php

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

Output:

Hello escape "sequence" characters

PHP echo: printing variable value

File: echo4.php

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

Output:

Message is: Hello JavaTpoint PHP

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

```php
<?php
```

```php
print "Hello by PHP print ";
print ("Hello by PHP print()");
?>
```

Output:

Hello by PHP print Hello by PHP print()

PHP print: printing multi line string

File: print2.php

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

Output:

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

PHP print: printing escaping characters

File: print3.php

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

Output:

Hello escape "sequence" characters by PHP print

PHP print: printing variable value

File: print4.php

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

Output:

Message is: Hello print() in PHP

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:

- o echo
- o 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

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

print

- o print is also a statement, used as an alternative to echo at many times to display the output.
- o print can be used with or without parentheses.
- o print always returns an integer value, which is 1.
- o Using print, we cannot pass multiple arguments.
- o 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
<?php
    $fname = "MREC";
    $lname = "CSE";
    echo "My name is: ".$fname,$lname;
```

```php
?>
```

Output:

My name is: GunjanGarg
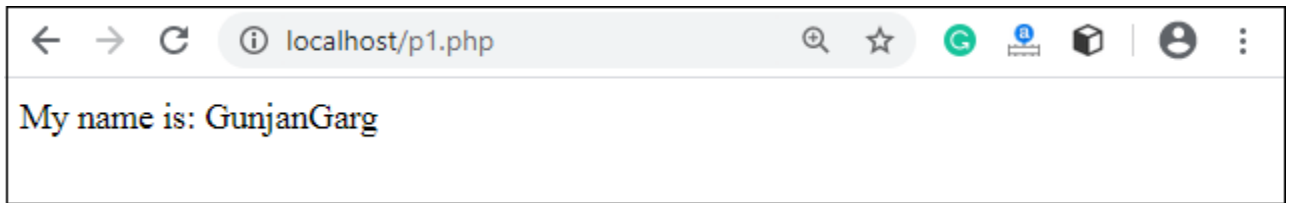
It will generate a **syntax error** because of multiple arguments in a print statement.

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

Output:

Parse error: syntax error, unexpected ',' in **D:\xampp\htdocs\p1.php** on line **6**
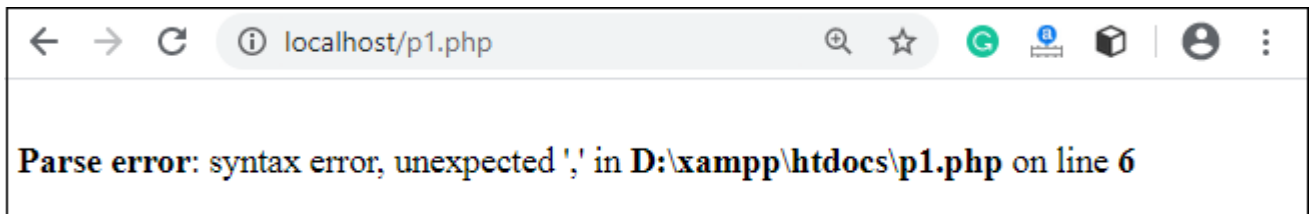
For Example (Check Return Value)

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

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

Output:

Parse error: syntax error, unexpected 'echo' (T_ECHO) in **D:\xampp\htdocs\p1.php** on line **4**
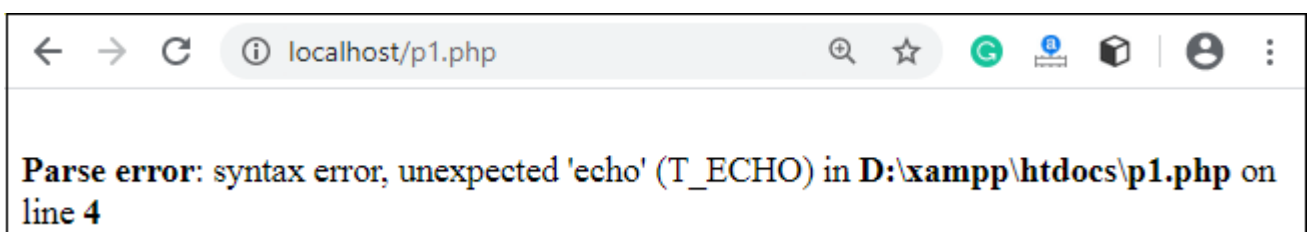
As we already discussed that print returns a value, which is always 1.

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

Output:
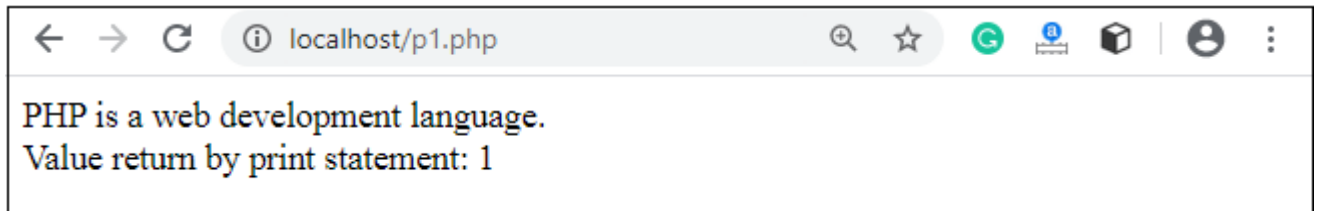
PHP Variables

<="" 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.
- 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:

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

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

Output:

```
hello
hello
```

File: variableinvalid.php

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

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

Output:

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

PHP: Loosely typed language

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

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

Output:

```
Local variable declared inside the function is: 45
```
File: local_variable2.php

```php
<?php
   function mytest()
   {
```

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


//  echo $lang;
?>
```

Output:

```
Web development language: PHP
Notice: Undefined variable: lang in D:\xampp\htdocs\program\p3.php on line 28
```

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
<?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: Sanaya Sharma

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

Example:

File: global_variable2.php

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

Output:

```
Notice: Undefined variable: name in D:\xampp\htdocs\program\p3.php on line 6
Variable inside the function:
```

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
<?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();
?>
```

Output:

```
Sum of global variables is: 18
```

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

Output:

```
Value of x: 7
```

Note: local variable has higher priority than the global variable.

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

```
?>
```

Output:


```
Static: 4
Non-static: 7
Static: 5
Non-static: 7
```

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

Output:


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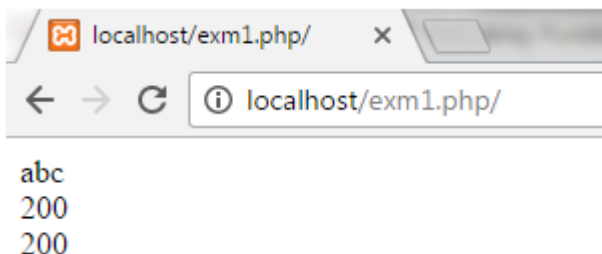
In the above example, we have assigned a value to the variable **x** as **abc**. Value of reference variable **$$x** is assigned as **200**.

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

Example2

```php
<?php
```

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

Output:

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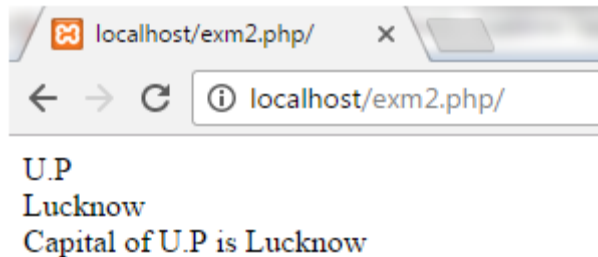
Now we have printed the values **$x, $$x** and a string.

Example3

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

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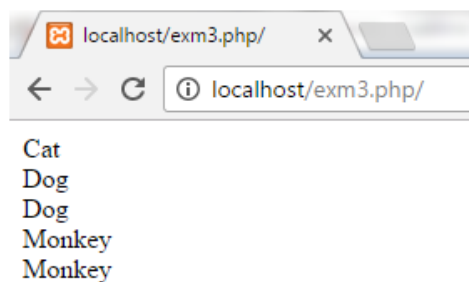
In the above example, we have assigned a value to the variable name **Cat**. Value of reference variable **${$name}** is assigned as **Dog** and **${${$name}}** as **Monkey**.

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

PHP 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
<?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:

- o An integer can be either positive or negative.

- o An integer must not contain decimal point.

- o Integer can be decimal (base 10), octal (base 8), or hexadecimal (base 16).

- o 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>"; //34
    echo "Octal number: " .$oct1. "<br>"; //163
    echo "HexaDecimal number: " .$hexa1. "<br>"; //69
?>
```

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

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:

```php
<?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>";
        ?>
```

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:

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

            $obj = new bike();
            $obj -> model();
        ?>
```

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 d

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

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:

$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:

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

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

- o **Unary Operators:** works on single operands such as ++, -- etc.
- o **Binary Operators:** works on two operands such as binary +, -, *, / etc.
- o **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 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 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 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 numb 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 Return 0 if $a is equal 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 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
1. <?php
2.     //class declaration
3.     class Developer
4.     {}
5.     class Programmer
6.     {}
7.     //creating an object of type Developer
8.     $charu = new Developer();
9.
10.    //testing the type of object
11.    if( $charu instanceof Developer)
12.    {
13.        echo "Charu is a developer.";
```

```
14.    }
15.    else
16.    {
17.        echo "Charu is a programmer.";
18.    }
19.    echo "</br>";
20.    var_dump($charu instanceof Developer);          //It will return true.
21.    var_dump($charu instanceof Programmer);       //It will return false.
22. ?>
```

Output:

```
Charu is a developer.
bool(true) bool(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 resu... Here, it will show the directories available in curre... folder.

> *Note: Note that backticks (``) are not single-quotes.*

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
||	logical OR	left

?:	ternary	left
= += -= *= **= /= .= %= &= |= ^= <<= >>= =>	assignment	right
and	logical	left
xor	logical	left
or	logical	left
,	many uses (comma)	left

PHP If Else

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

- o   if
- o   if-else
- o   if-else-if
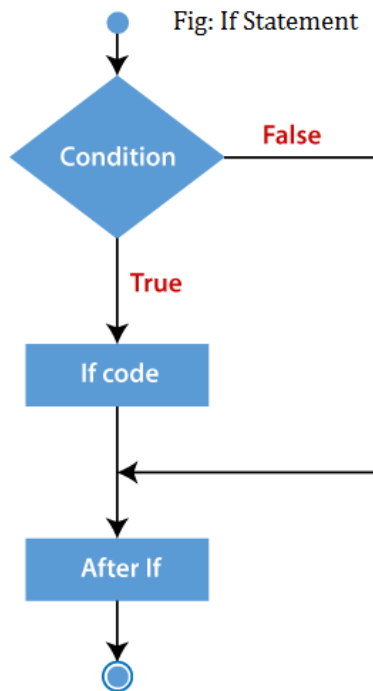- o   nested if

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

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

Flowchart

Fig: If Statement

Example

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

Output:

```
12 is less than 100
```

PHP If-else Statement

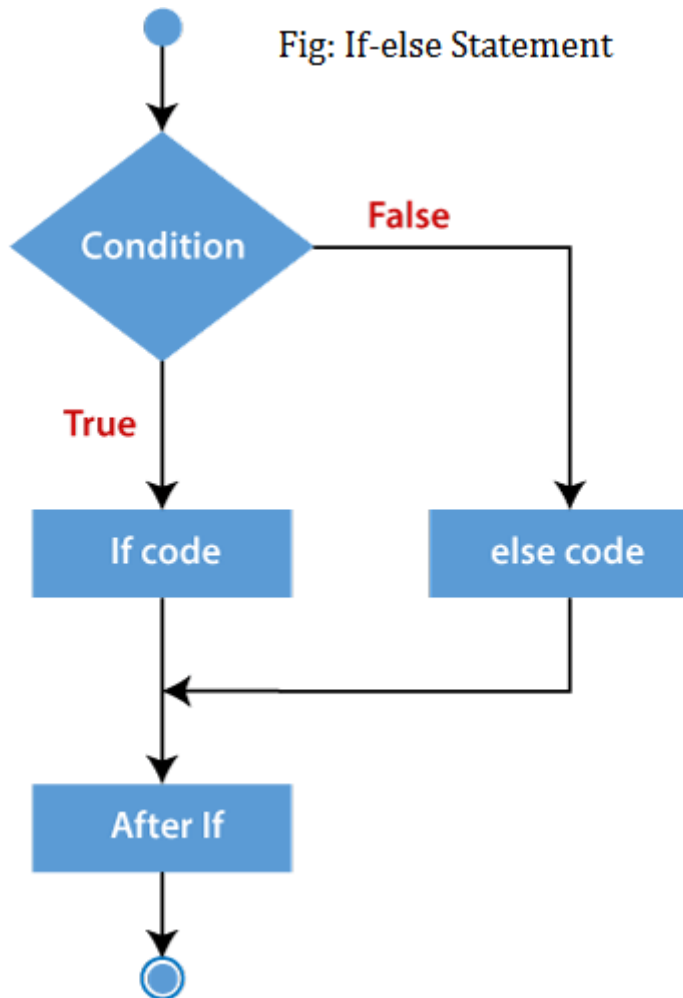PHP if-else statement is executed whether condition is true or false.

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

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

Flowchart

Fig: If-else Statement

Example

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

Output:

```
12 is even 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

1. `if (condition1){`

2. //code to be executed if condition1 is true
3. } **elseif** (condition2){
4. //code to be executed if condition2 is true
5. } **elseif** (condition3){
6. //code to be executed if condition3 is true
7. ....
8. } **else**{
9. //code to be executed if all given conditions are false
10.}

Flowchart

Fig: if-else-if Statement
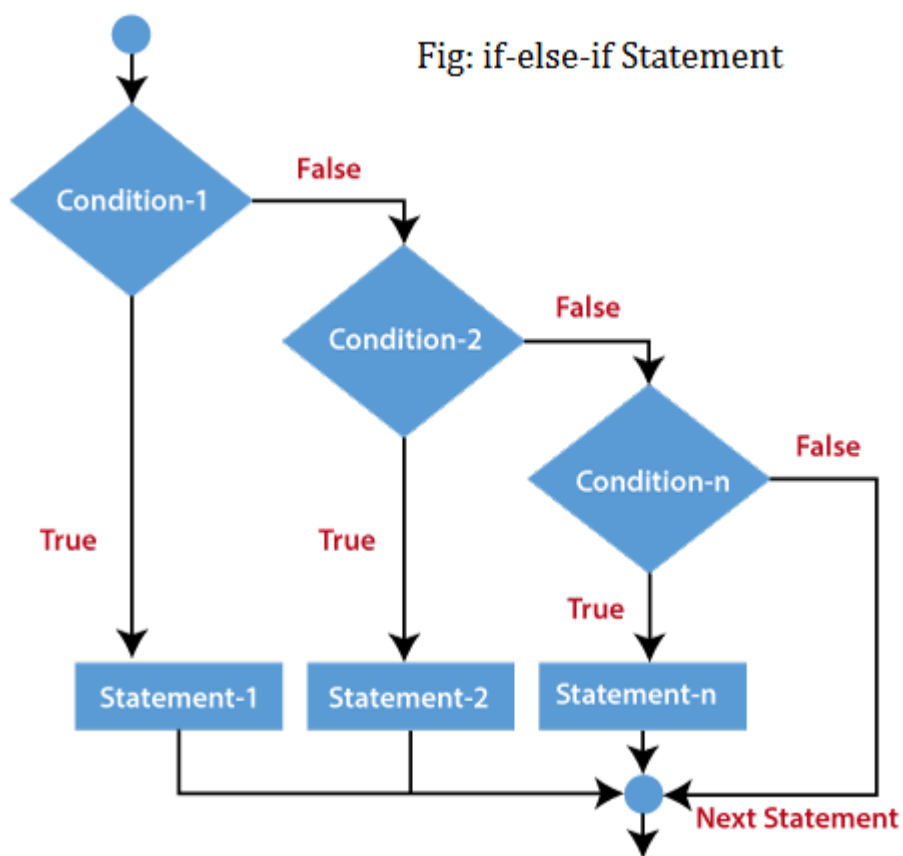
Example

```php
<?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";
        }
    ?>
```

Output:

```
B Grade
```

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

Fig: Nested if Statement
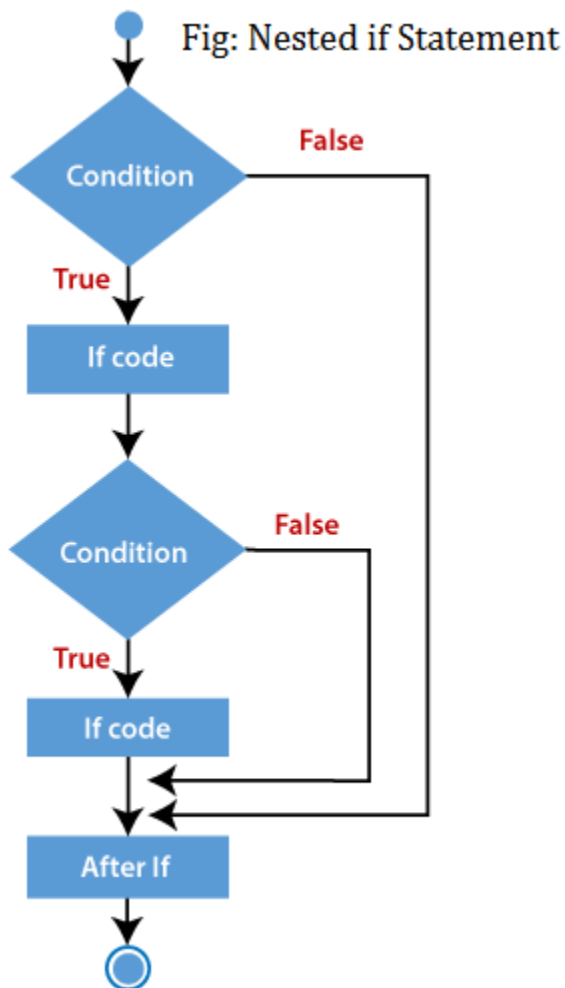
Example

```php
<?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";
      }
   }
?>
```

Output:

Eligible to give vote

PHP Switch Example

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

Output:

```
34 is smaller than 56 and 45
```

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

Fig: Switch Statement
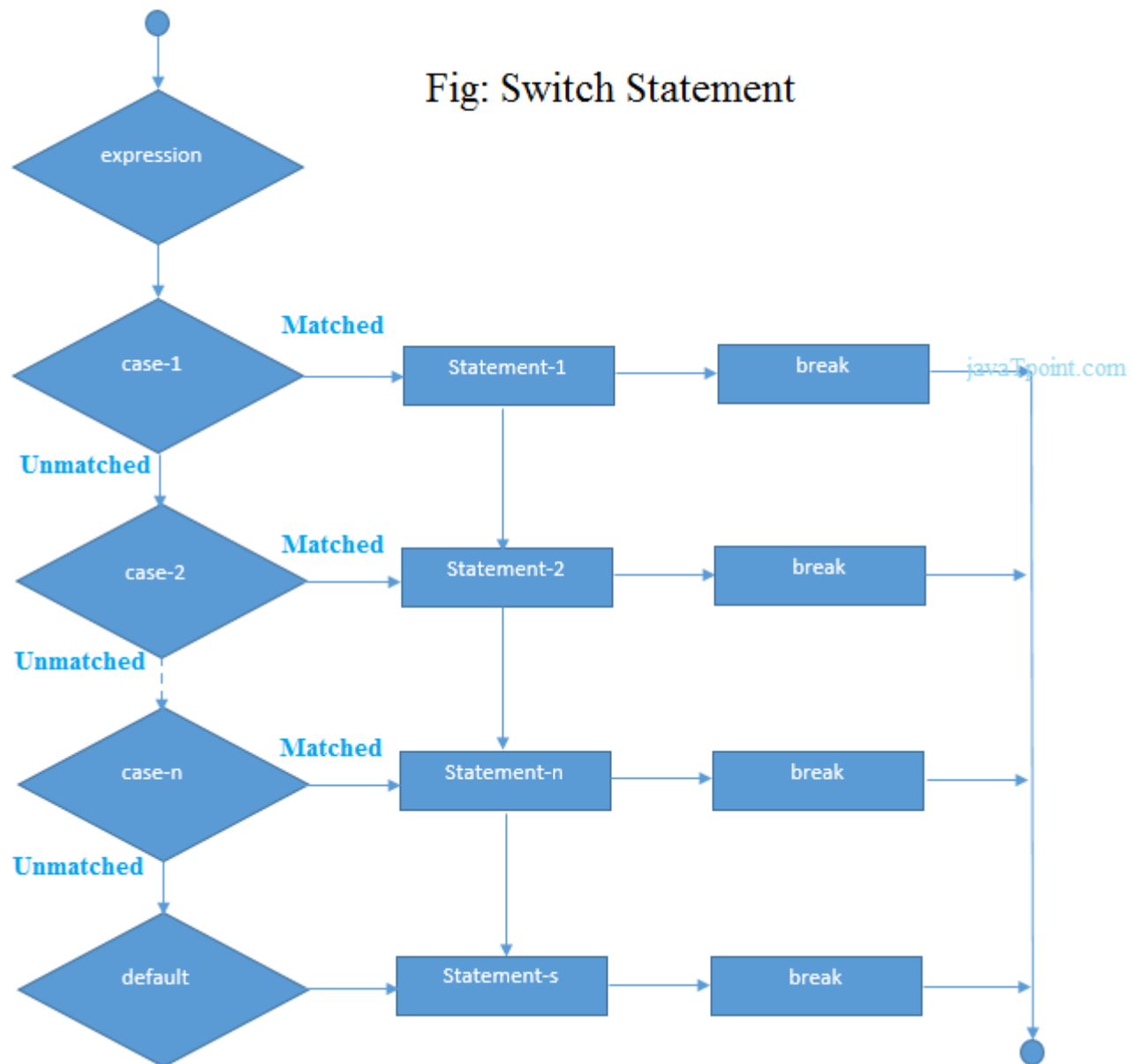
PHP Switch Example

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

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

Output:

```
number is equal to 20
```

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
<?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':
```

```php
        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;
    }
?>
```

Output:

```
Given character is vowel
```

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
<?php
$ch = "B.Tech";
switch ($ch)
{
    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:
```

```php
            echo "Wrong Choice";
                break;
        }
    ?>
```

Output:

B.Tech is 4 years course

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
    <?php
        $ch = 'c';
        switch ($ch)
        {
            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";
        }
    ?>
```

Output:

Choice c
Choice d
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
<?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;
```

```
        }
    ?>
```

Output:

Hyundai Tucson price is 22.39 - 32.07 L.

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

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

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
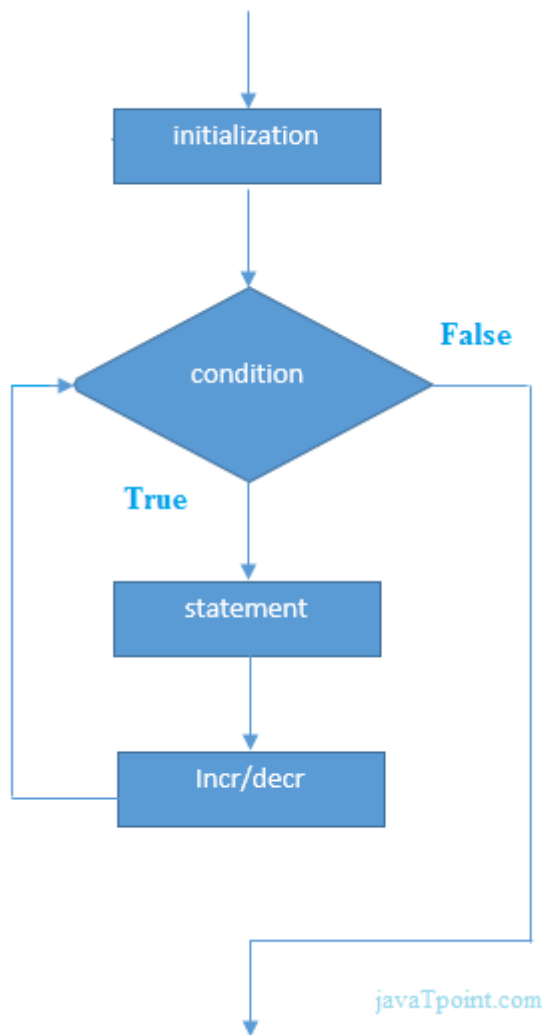
Example

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

Output:

```
1
2
3
4
5
6
7
8
9
10
```

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

Output:

```
1
2
3
4
.
.
.
```

Example

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

```php
<?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>";
```

```php
/* 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++);
?>
```

Output:

```
123456789
123456789
123456789
123456789
```

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

Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

PHP For Each Loop

PHP for each loop is used to traverse array elements.

Syntax

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

Example

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

Output:

```
Season is: summer
Season is: winter
Season is: spring
Season is: autumn
```

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

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

There is one more syntax of foreach loop.

Syntax

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

Flowchart
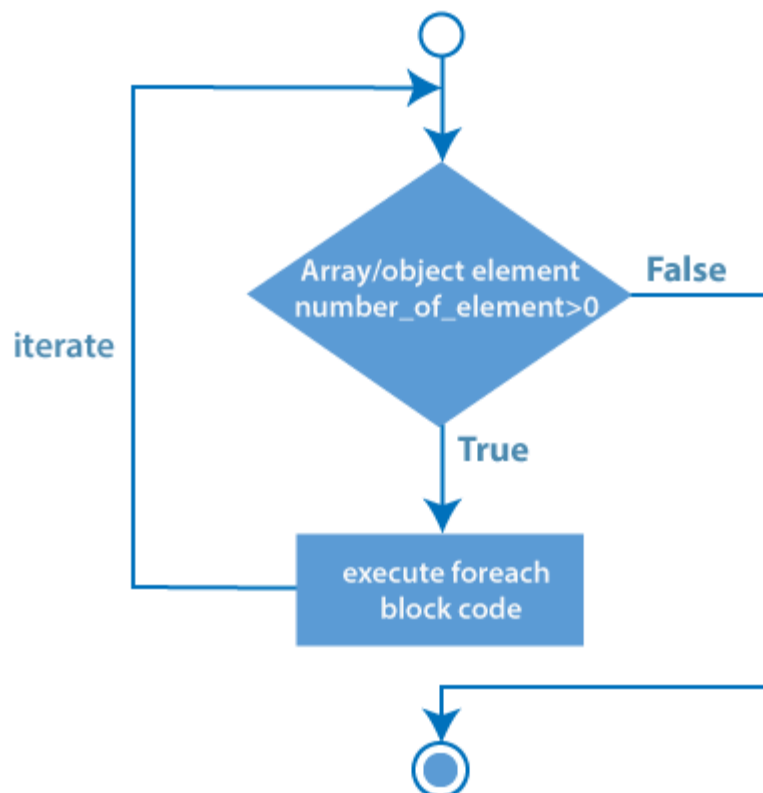
Example 1:

PHP program to print array elements using foreach loop.

```php
<?php
   //declare array
   $season = array ("Summer", "Winter", "Autumn", "Rainy");

   //access array elements using foreach loop
   foreach ($season as $element) {
      echo "$element";
      echo "</br>";
   }
```

```
?>
```

Output:

```
Summer
Winter
Autumn
Rainy
```

Example 2:

PHP program to print associative array elements using foreach loop.

```php
<?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>";
    }
?>
```

Output:

```
Name : Alex
Email : alex_jtp@gmail.com
Age : 21
Gender : Male
```

Example 3:

Multi-dimensional array

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

```php
        //display multi-dimensional array elements through foreach loop
        foreach ($a as $e1) {
           foreach ($e1 as $e2) {
              echo "$e2\n";
           }
        }
     ?>
```

Output:

Alex Bob Camila Denial

Example 4:

Dynamic array

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

Output:

j a v a t p o i n t

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

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

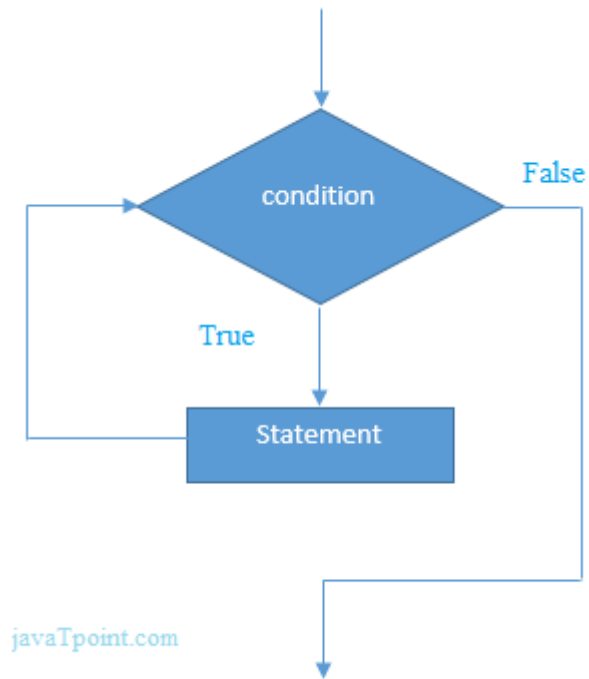## Alternative Syntax

while(condition):

//code to be executed

endwhile;

PHP While Loop Flowchart

PHP While Loop Example

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

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Alternative Example

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

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Example

Below is the example of printing alphabets using while loop.

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

Output:

```
A
B
C
D
E
F
G
```

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

Output:

```
1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3
```

PHP Infinite While Loop

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

Syntax

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

Example

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

```
?>
```

Output:

```
Hello Javatpoint!
Hello Javatpoint!
Hello Javatpoint!
Hello Javatpoint!
.
.
.
.
.
Hello Javatpoint!
Hello Javatpoint!
```

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
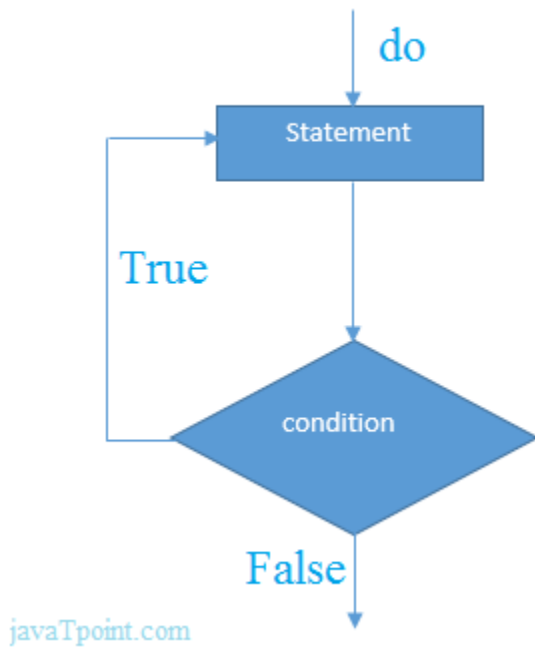
Example

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

Output:

```
1
2
3
4
5
6
7
8
9
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
<?php
  $x = 5;
```

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

Output:

```
Welcome to javatpoint!
Welcome to javatpoint!
Welcome to javatpoint!
Welcome to javatpoint!
Welcome to javatpoint!
```

Example

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

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

Output:

```
1 is not greater than 10.
2
```

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 first and

block of statements executes.	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.

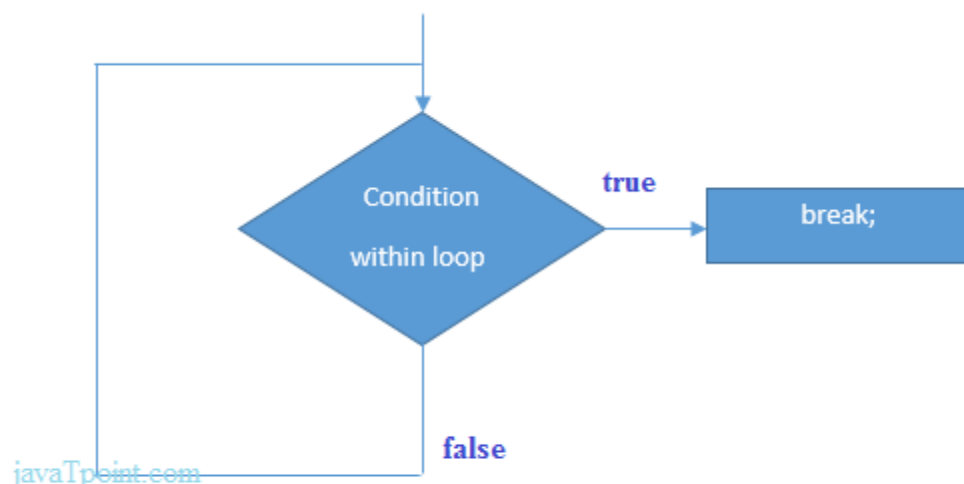The break statement can be used in all types of loops such as while, do-while, for, foreach loop, and also with switch case.

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

Output:

```
1
2
3
4
5
```

PHP Break: inside inner loop

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

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

Output:

```
1 1
1 2
1 3
2 1
2 2
3 1
3 2
3 3
```

PHP Break: inside switch statement

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

```php
<?php
$num=200;
```

```php
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");
}
?>
```

Output:

```
number is equal to 200
```

PHP Break: with array of string

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

Output:

```
One
Two
Three
```

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

Output:

```
$car is not Mercedes Benz
$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
<?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;
    }
}?>
```

Output:

```
At matched condition i = 5
At matched condition i = 10; quitting
```

PHP continue statement

The PHP continue statement is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.

The continue statement is used within looping and switch control structure when you immediately jump to the next iteration.
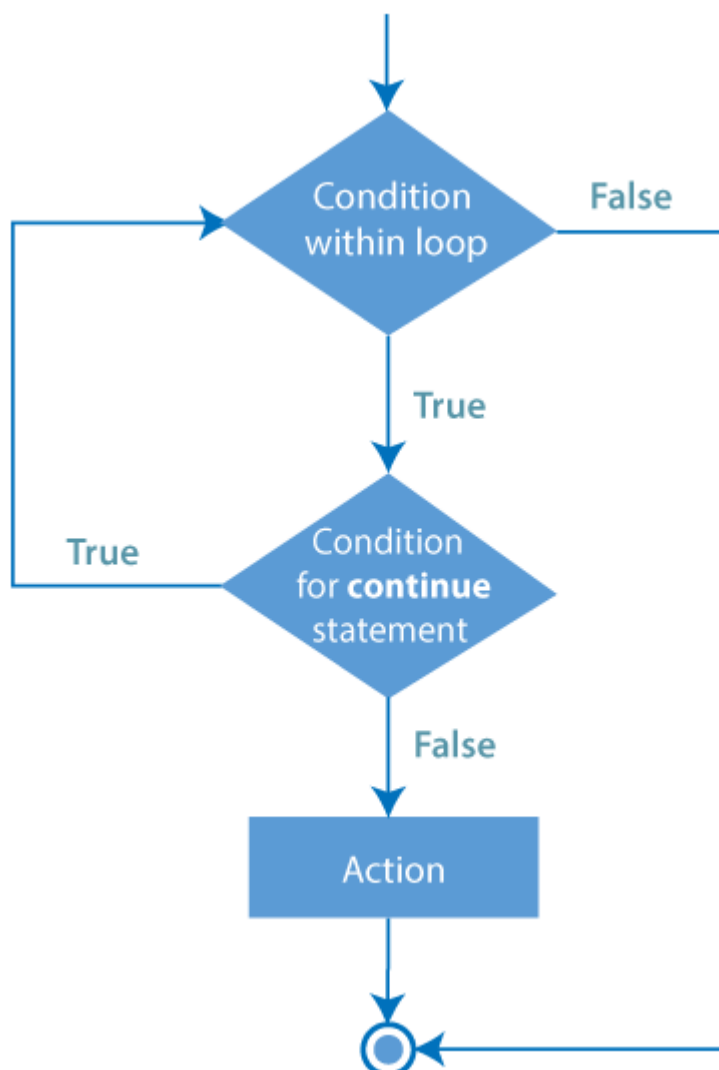
The continue statement can be used with all types of loops such as - for, while, do-while, and foreach loop. The continue statement allows the user to skip the execution of the code for the specified condition.

Syntax

The syntax for the continue statement is given below:

```
jump-statement;
continue;
```

Flowchart:

PHP Continue Example with for loop

Example

In the following example, we will print only those values of i and j that are same and skip others.

```php
<?php
    //outer loop
    for ($i =1; $i<=3; $i++) {
        //inner loop
        for ($j=1; $j<=3; $j++) {
            if (!($i == $j) ) {
                continue;       //skip when i and j does not have same values
            }
            echo $i.$j;
            echo "</br>";
        }
    }
?>
```

Output:

```
11
22
33
```

PHP continue Example in while loop

Example

In the following example, we will print the even numbers between 1 to 20.

```php
<?php
    //php program to demonstrate the use of continue statement

    echo "Even numbers between 1 to 20: </br>";
    $i = 1;
    while ($i<=20) {
        if ($i %2 == 1) {
            $i++;
            continue;   //here it will skip rest of statements
        }
        echo $i;
        echo "</br>";
```

```
        $i++;
    }
?>
```

Output:

```
Even numbers between 1 to 20:
2
4
6
8
10
12
14
16
18
20
```

PHP continue Example with array of string

Example

The following example prints the value of array elements except those for which the specified condition is true and continue statement is used.

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

Output:

```
One
Two
Three
Four
```

PHP continue Example with optional argument

The continue statement accepts an optional numeric value, which is used accordingly. The numeric value describes how many nested structures it will exit.

Example

Look at the below example to understand it better:

```php
<?php
    //outer loop
    for ($i =1; $i<=3; $i++) {
        //inner loop
        for ($j=1; $j<=3; $j++) {
            if (($i == $j) ) {      //skip when i and j have same values
                continue 1;     //exit only from inner for loop
            }
            echo $i.$j;
            echo "</br>";
        }
    }
?>
```

Output:

```
12
13
21
23
31
32
```

PHP Functions

PHP function is a piece of code that can be reused many times. It can take input as argument list and return value. There are thousands of built-in functions in PHP.

In PHP, we can define **Conditional function**, **Function within Function** and **Recursive function** also.

Advantage of PHP Functions

Code Reusability: PHP functions are defined only once and can be invoked many times, like in other programming languages.

Less Code: It saves a lot of code because you don't need to write the logic many times. By the use of function, you can write the logic only once and reuse it.

Easy to understand: PHP functions separate the programming logic. So it is easier to understand the flow of the application because every logic is divided in the form of functions.

PHP User-defined Functions

We can declare and call user-defined functions easily. Let's see the syntax to declare user-defined functions.

Syntax

```
function functionname(){
//code to be executed
}
```

Note: Function name must be start with letter and underscore only like other labels in PHP. It can't be start with numbers or special symbols.

PHP Functions Example

File: function1.php

```
<?php
function sayHello(){
echo "Hello PHP Function";
}
sayHello();//calling function
?>
```

Output:

```
Hello PHP Function
```

PHP Function Arguments

We can pass the information in PHP function through arguments which is separated by comma.

PHP supports **Call by Value** (default), **Call by Reference**, **Default argument values** and **Variable-length argument list**.

Let's see the example to pass single argument in PHP function.

File: functionarg.php

```
<?php
function sayHello($name){
echo "Hello $name<br/>";
}
sayHello("Sonoo");
sayHello("Vimal");
sayHello("John");
?>
```

Output:

```
Hello Sonoo
Hello Vimal
Hello John
```

Let's see the example to pass two argument in PHP function.

File: functionarg2.php

```php
<?php
function sayHello($name,$age){
echo "Hello $name, you are $age years old<br/>";
}
sayHello("Sonoo",27);
sayHello("Vimal",29);
sayHello("John",23);
?>
```

Output:

```
Hello Sonoo, you are 27 years old
Hello Vimal, you are 29 years old
Hello John, you are 23 years old
```

PHP Call By Reference

Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.

By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

Let's see a simple example of call by reference in PHP.

File: functionref.php

```php
<?php
function adder(&$str2)
{
    $str2 .= 'Call By Reference';
}
$str = 'Hello ';
adder($str);
echo $str;
?>
```

Output:

PHP Function: Default Argument Value

We can specify a default argument value in function. While calling PHP function if you don't specify any argument, it will take the default argument. Let's see a simple example of using default argument value in PHP function.

File: functiondefaultarg.php

```php
<?php
function sayHello($name="Sonoo"){
echo "Hello $name<br/>";
}
sayHello("Rajesh");
sayHello();//passing no value
sayHello("John");
?>
```

Output:

```
Hello Rajesh
Hello Sonoo
Hello John
```

PHP Function: Returning Value

Let's see an example of PHP function that returns value.

File: functiondefaultarg.php

```php
<?php
function cube($n){
return $n*$n*$n;
}
echo "Cube of 3 is: ".cube(3);
?>
```

Output:

```
Cube of 3 is: 27
```

PHP Parameterized Function

PHP Parameterized functions are the functions with parameters. You can pass any number of parameters inside a function. These passed parameters act as variables inside your function.

They are specified inside the parentheses, after the function name.

The output depends upon the dynamic values passed as the parameters into the function.
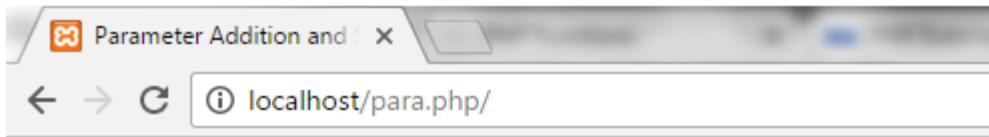
PHP Parameterized Example 1

Addition and Subtraction

In this example, we have passed two parameters **$x** and **$y** inside two functions **add()** and **sub()**.

```php
<!DOCTYPE html>
<html>
<head>
   <title>Parameter Addition and Subtraction Example</title>
</head>
<body>
<?php
    //Adding two numbers
    function add($x, $y) {
      $sum = $x + $y;
      echo "Sum of two numbers is = $sum <br><br>";
    }
    add(467, 943);

    //Subtracting two numbers
    function sub($x, $y) {
      $diff = $x - $y;
      echo "Difference between two numbers is = $diff";
    }
    sub(943, 467);
  ?>
</body>
</html>
```

Output:

Sum of two numbers is = 1410

Difference between two numbers is = 476
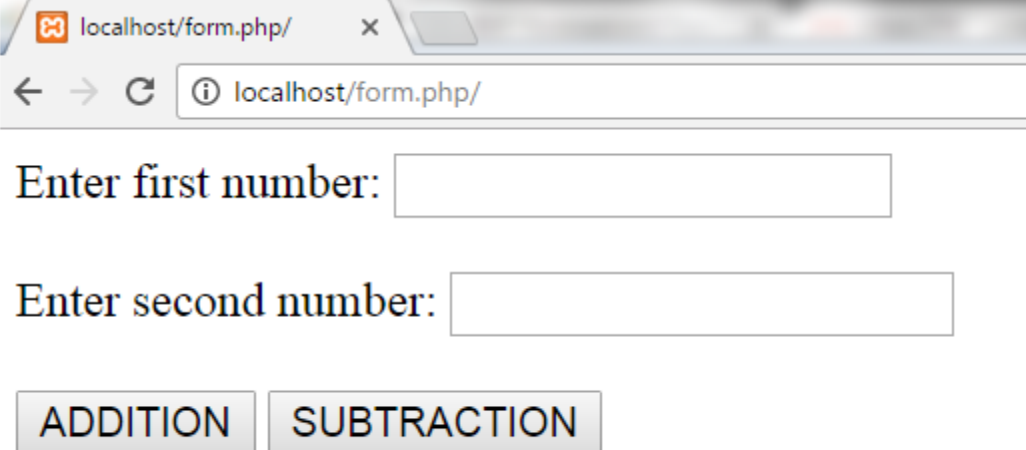
PHP Parameterized Example 2

Addition and Subtraction with Dynamic number

In this example, we have passed two parameters **$x** and **$y** inside two functions **add()** and **sub()**.

```php
<?php
//add() function with two parameter
function add($x,$y)
{
$sum=$x+$y;
echo "Sum = $sum <br><br>";
}
//sub() function with two parameter
function sub($x,$y)
{
$sub=$x-$y;
echo "Diff = $sub <br><br>";
}
//call function, get  two argument through input box and click on add or sub button
if(isset($_POST['add']))
{
//call add() function
 add($_POST['first'],$_POST['second']);
}
if(isset($_POST['sub']))
{
//call add() function
sub($_POST['first'],$_POST['second']);
}
```

```
?>
<form method="post">
Enter first number: <input type="number" name="first"/><br><br>
Enter second number: <input type="number" name="second"/><br><br
>
<input type="submit" name="add" value="ADDITION"/>
<input type="submit" name="sub" value="SUBTRACTION"/>
</form>
```
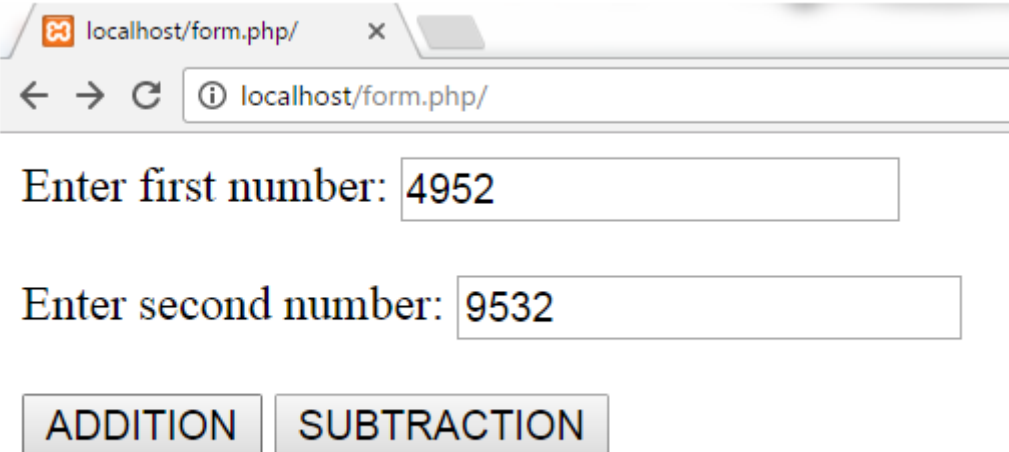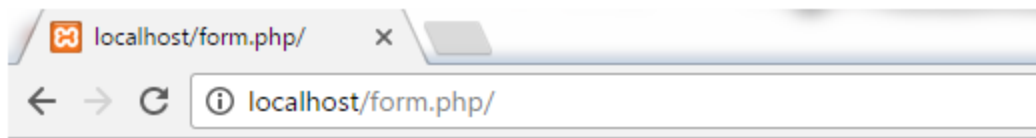
Output:

We passed the following number,

Now clicking on ADDITION button, we get the following output.

Sum = 14484

Enter first number: [   ]

Enter second number: [   ]

[ADDITION] [SUBTRACTION]

Now clicking on SUBTRACTION button, we get the following output.
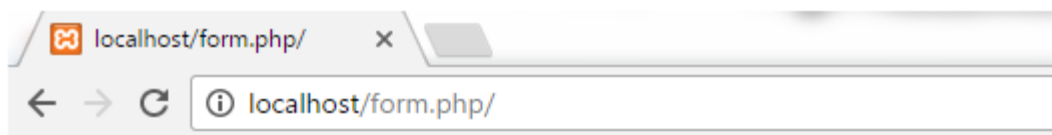
Diff = -4580

Enter first number: [   ]

Enter second number: [   ]

[ADDITION] [SUBTRACTION]

PHP Call By Value

PHP allows you to call function by value and reference both. In case of PHP call by value, actual value is not modified if it is modified inside the function.

Let's understand the concept of call by value by the help of examples.

Example 1

In this example, variable $str is passed to the adder function where it is concatenated with 'Call By Value' string. But, printing $str variable results 'Hello' only. It is because changes are done in the local variable $str2 only. It doesn't reflect to $str variable.

```php
<?php
```

```php
    function adder($str2)
    {
        $str2 .= 'Call By Value';
    }
    $str = 'Hello ';
    adder($str);
    echo $str;
    ?>
```

Output:

```
Hello
```

Example 2

Let's understand PHP call by value concept through another example.

```php
    <?php
    function increment($i)
    {
        $i++;
    }
    $i = 10;
    increment($i);
    echo $i;
    ?>
```

Output:

```
10
```

PHP Call By Reference

In case of PHP call by reference, actual value is modified if it is modified inside the function. In such case, you need to use & (ampersand) symbol with formal arguments. The & represents reference of the variable.

Let's understand the concept of call by reference by the help of examples.

Example 1

In this example, variable $str is passed to the adder function where it is concatenated with 'Call By Reference' string. Here, printing $str variable results 'This is Call By Reference'. It is because changes are done in the actual variable $str.

```php
<?php
function adder(&$str2)
{
    $str2 .= 'Call By Reference';
}
$str = 'This is ';
adder($str);
echo $str;
?>
```

Output:

```
This is Call By Reference
```

Example 2

Let's understand PHP call by reference concept through another example.

```php
<?php
function increment(&$i)
{
    $i++;
}
$i = 10;
increment($i);
echo $i;
?>
```

Output:

```
11
```

PHP Default Argument Values Function

PHP allows you to define C++ style default argument values. In such case, if you don't pass any value to the function, it will use default argument value.

Let' see the simple example of using PHP default arguments in function.

Example 1

```php
<?php
function sayHello($name="Ram"){
echo "Hello $name<br/>";
}
```

```php
sayHello("Sonoo");
sayHello();//passing no value
sayHello("Vimal");
?>
```

Output:

```
Hello Sonoo
Hello Ram
Hello Vimal
```

> *Since PHP 5, you can use the concept of default argument value with call by reference also.*

Example 2

```php
<?php
function greeting($first="Sonoo",$last="Jaiswal"){
echo "Greeting: $first $last<br/>";
}
greeting();
greeting("Rahul");
greeting("Michael","Clark");
?>
```

Output:

```
Greeting: Sonoo Jaiswal
Greeting: Rahul Jaiswal
Greeting: Michael Clark
```

Example 3

```php
<?php
function add($n1=10,$n2=10){
$n3=$n1+$n2;
echo "Addition is: $n3<br/>";
}
add();
add(20);
add(40,40);
?>
```

Output:

```
Addition is: 20
Addition is: 30
Addition is: 80
```

PHP Variable Length Argument Function

PHP supports variable length argument function. It means you can pass 0, 1 or n number of arguments in function. To do so, you need to use 3 ellipses (dots) before the argument name.

The 3 dot concept is implemented for variable length argument since PHP 5.6.

Let's see a simple example of PHP variable length argument function.

```php
<?php
function add(...$numbers) {
    $sum = 0;
    foreach ($numbers as $n) {
        $sum += $n;
    }
    return $sum;
}

echo add(1, 2, 3, 4);
?>
```

Output:

```
10
```

PHP Recursive Function

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

It is recommended to avoid recursive function call over 200 recursion level because it may smash the stack and may cause the termination of script.

Example 1: Printing number

```php
<?php
function display($number) {
    if($number<=5){
     echo "$number <br/>";
     display($number+1);
    }
}

display(1);
```

```
?>
```

Output:

```
1
2
3
4
5
```

Example 2 : Factorial Number

```php
<?php
function factorial($n)
{
   if ($n < 0)
      return -1; /*Wrong value*/
   if ($n == 0)
      return 1; /*Terminating condition*/
   return ($n * factorial ($n -1));
}

echo factorial(5);
?>
```

Output:

```
120
```

PHP Arrays

PHP array is an ordered map (contains value on the basis of key). It is used to hold multiple values of similar type in a single variable.

Advantage of PHP Array

Less Code: We don't need to define multiple variables.

Easy to traverse: By the help of single loop, we can traverse all the elements of an array.

Sorting: We can sort the elements of array.

PHP Array Types

There are 3 types of array in PHP.

1. Indexed Array

2. Associative Array
3. Multidimensional Array

PHP Indexed Array

PHP index is represented by number which starts from 0. We can store number, string and object in the PHP array. All PHP array elements are assigned to an index number by default.

There are two ways to define indexed array:

1st way: $season=**array**("summer","winter","spring","autumn");

2nd way:

$season[0]="summer";
$season[1]="winter";
$season[2]="spring";
$season[3]="autumn";

Example

File: array1.php

```php
<?php
$season=array("summer","winter","spring","autumn");
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";

?>
```

Output:

Season are: summer, winter, spring and autumn

File: array2.php

```php
<?php
$season[0]="summer";
$season[1]="winter";
$season[2]="spring";
$season[3]="autumn";
echo "Season are: $season[0], $season[1], $season[2] and $season[3]";
?>
```

Output:

Season are: summer, winter, spring and autumn

PHP Associative Array

We can associate name with each array elements in PHP using => symbol.

There are two ways to define associative array:

1st way:

$salary=**array**("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");

2nd way:

```
$salary["Sonoo"]="350000";
$salary["John"]="450000";
$salary["Kartik"]="200000";
```

Example

File: arrayassociative1.php

```php
<?php
$salary=array("Sonoo"=>"350000","John"=>"450000","Kartik"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "John salary: ".$salary["John"]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>";
?>
```

Output:

```
Sonoo salary: 350000
John salary: 450000
Kartik salary: 200000
```

File: arrayassociative2.php

```php
<?php
$salary["Sonoo"]="350000";
$salary["John"]="450000";
$salary["Kartik"]="200000";
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "John salary: ".$salary["John"]."<br/>";
echo "Kartik salary: ".$salary["Kartik"]."<br/>";
?>
```

Output:

```
Sonoo salary: 350000
John salary: 450000
Kartik salary: 200000
```

PHP Indexed Array

PHP indexed array is an array which is represented by an index number by default. All elements of array are represented by an index number which starts from 0.

PHP indexed array can store numbers, strings or any object. PHP indexed array is also known as numeric array.

Definition

There are two ways to define indexed array:

1st way:

$size=**array**("Big","Medium","Short");

2nd way:

```
$size[0]="Big";
$size[1]="Medium";
$size[2]="Short";
```

PHP Indexed Array Example

File: array1.php

```
<?php
$size=array("Big","Medium","Short");
echo "Size: $size[0], $size[1] and $size[2]";
?>
```

Output:

```
Size: Big, Medium and Short
```

File: array2.php

```
<?php
$size[0]="Big";
$size[1]="Medium";
$size[2]="Short";
```

```php
echo "Size: $size[0], $size[1] and $size[2]";
?>
```

Output:

```
Size: Big, Medium and Short
```

Traversing PHP Indexed Array

We can easily traverse array in PHP using foreach loop. Let's see a simple example to traverse all the elements of PHP array.

File: array3.php

```php
<?php
$size=array("Big","Medium","Short");
foreach( $size as $s )
{
  echo "Size is: $s<br />";
}
?>
```

Output:

```
Size is: Big
Size is: Medium
Size is: Short
```

Count Length of PHP Indexed Array

PHP provides count() function which returns length of an array.

```php
<?php
$size=array("Big","Medium","Short");
echo count($size);
?>
```

Output:

```
3
```

PHP Associative Array

PHP allows you to associate name/label with each array elements in PHP using => symbol. Such way, you can easily remember the element because each element is represented by label than an incremented number.

Definition

There are two ways to define associative array:

1st way:

$salary=**array**("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");

2nd way:

```
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000";
```

Example

File: arrayassociative1.php

```php
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```

Output:

```
Sonoo salary: 550000
Vimal salary: 250000
Ratan salary: 200000
```

File: arrayassociative2.php

```php
<?php
$salary["Sonoo"]="550000";
$salary["Vimal"]="250000";
$salary["Ratan"]="200000";
echo "Sonoo salary: ".$salary["Sonoo"]."<br/>";
echo "Vimal salary: ".$salary["Vimal"]."<br/>";
echo "Ratan salary: ".$salary["Ratan"]."<br/>";
?>
```

Output:

```
Sonoo salary: 550000
Vimal salary: 250000
Ratan salary: 200000
```

Traversing PHP Associative Array

By the help of PHP for each loop, we can easily traverse the elements of PHP associative array.

```php
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
foreach($salary as $k => $v) {
echo "Key: ".$k." Value: ".$v."<br/>";
}
?>
```

Output:

```
Key: Sonoo Value: 550000
Key: Vimal Value: 250000
Key: Ratan Value: 200000
```

PHP Multidimensional Array

PHP multidimensional array is also known as array of arrays. It allows you to store tabular data in an array. PHP multidimensional array can be represented in the form of matrix which is represented by row * column.

Definition

```php
$emp = array
(
array(1,"sonoo",400000),
array(2,"john",500000),
array(3,"rahul",300000)
);
```

PHP Multidimensional Array Example

Let's see a simple example of PHP multidimensional array to display following tabular data. In this example, we are displaying 3 rows and 3 columns.

Id	Name	Salary
1	sonoo	400000

| 2 | john | 500000 |
| 3 | rahul | 300000 |

File: multiarray.php

```php
<?php
$emp = array
 (
 array(1,"sonoo",400000),
 array(2,"john",500000),
 array(3,"rahul",300000)
 );

for ($row = 0; $row < 3; $row++) {
 for ($col = 0; $col < 3; $col++) {
   echo $emp[$row][$col]."  ";
 }
 echo "<br/>";
}
?>
```

Output:

```
1 sonoo 400000
2 john 500000
3 rahul 300000
```

PHP Array Functions

PHP provides various array functions to access and manipulate the elements of array. The important PHP array functions are given below.

1) PHP array() function

PHP array() function creates and returns an array. It allows you to create indexed, associative and multidimensional arrays.

Syntax: **array array** ([mixed $...])

Example

```php
<?php
$season=array("summer","winter","spring","autumn");
```

```
    echo "Season are: $season[0], $season[1], $season[2] and $season[3]";

    ?>
```

Output:

```
Season are: summer, winter, spring and autumn
```

2) PHP array_change_key_case() function

PHP array_change_key_case() function changes the case of all key of an array.

Note: It changes case of key only.

Syntax:
array array_change_key_case (**array** $array [, int $case = CASE_LOWER])

Example

```php
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");

print_r(array_change_key_case($salary,CASE_UPPER));
?>
```

Output:

```
Array ( [SONOO] => 550000 [VIMAL] => 250000 [RATAN] => 200000 )
```

Example

```php
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");

print_r(array_change_key_case($salary,CASE_LOWER));
?>
```

Output:

```
Array ( [sonoo] => 550000 [vimal] => 250000 [ratan] => 200000 )
```

3) PHP array_chunk() function

PHP array_chunk() function splits array into chunks. By using array_chunk() method, you can divide array into many parts.

Syntax:
array array_chunk (array $array , int $size [, bool $preserve_keys = false])

Example

```php
<?php
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_chunk($salary,2));
?>
```

Output:

```
Array (
[0] => Array ( [0] => 550000 [1] => 250000 )
[1] => Array ( [0] => 200000 )
)
```

4) PHP count() function

PHP count() function counts all elements in an array.

Syntax:
int count (mixed $array_or_countable [, int $mode = COUNT_NORMAL])

Example

```php
<?php
$season=array("summer","winter","spring","autumn");
echo count($season);
?>
```

Output:

```
4
```

5) PHP sort() function

PHP sort() function sorts all the elements in an array.

Syntax:
bool sort (array &$array [, int $sort_flags = SORT_REGULAR])

Example

```php
<?php
$season=array("summer","winter","spring","autumn");
```

```
    sort($season);
    foreach( $season as $s )
    {
      echo "$s<br />";
    }
    ?>
```

Output:

```
autumn
spring
summer
winter
```

6) PHP array_reverse() function

PHP array_reverse() function returns an array containing elements in reversed order.

Syntax:
array array_reverse (array $array [, bool $preserve_keys = false])

Example

```
    <?php
    $season=array("summer","winter","spring","autumn");
    $reverseseason=array_reverse($season);
    foreach( $reverseseason as $s )
    {
      echo "$s<br />";
    }
    ?>
```

Output:

```
autumn
spring
winter
summer
```

7) PHP array_search() function

PHP array_search() function searches the specified value in an array. It returns key if search is successful.

Syntax:
mixed array_search (mixed $needle , array $haystack [, bool $strict = false])

Example

```php
<?php
$season=array("summer","winter","spring","autumn");
$key=array_search("spring",$season);
echo $key;
?>
```

Output:

```
2
```

8) PHP array_intersect() function

PHP array_intersect() function returns the intersection of two array. In other words, it returns the matching elements of two array.

Syntax:
array array_intersect (array $array1 , array $array2 [, array $...])

Example

```php
<?php
$name1=array("sonoo","john","vivek","smith");
$name2=array("umesh","sonoo","kartik","smith");
$name3=array_intersect($name1,$name2);
foreach( $name3 as $n )
{
  echo "$n<br />";
}
?>
```

Output:

```
sonoo
smith
```

PHP String

PHP string is a sequence of characters i.e., used to store and manipulate text. PHP supports only 256-character set and so that it does not offer native Unicode support. There are 4 ways to specify a string literal in PHP.

1. single quoted
2. double quoted

Single Quoted

We can create a string in PHP by enclosing the text in a single-quote. It is the easiest way to specify string in PHP.

For specifying a literal single quote, escape it with a backslash (\) and to specify a literal backslash (\) use double backslash (\\). All the other instances with backslash such as \r or \n, will be output same as they specified instead of having any special meaning.

For Example

Following some examples are given to understand the single quoted PHP String in a better way:

Example 1

```php
<?php
    $str='Hello text within single quote';
    echo $str;
?>
```

Output:

```
Hello text within single quote
```

We can store multiple line text, special characters, and escape sequences in a single-quoted PHP string.

Example 2

```php
<?php
$str1='Hello text
multiple line
text within single quoted string';
$str2='Using double "quote" directly inside single quoted string';
$str3='Using escape sequences \n in single quoted string';
echo "$str1 <br/> $str2 <br/> $str3";
?>
```

Output:

```
Hello text multiple line text within single quoted string
Using double "quote" directly inside single quoted string
Using escape sequences \n in single quoted string
```

Example 3

```php
<?php
```

```php
$num1=10;
$str1='trying variable $num1';
$str2='trying backslash n and backslash t inside single quoted string \n \t'
;
$str3='Using single quote \'my quote\' and \\backslash';
echo "$str1 <br/> $str2 <br/> $str3";
?>
```

Output:

```
trying variable $num1
trying backslash n and backslash t inside single quoted string \n \t
Using single quote 'my quote' and \backslash
```

Note: *In single quoted PHP strings, most escape sequences and variables will not be interpreted. But, we can use single quote through \' and backslash through \\ inside single quoted PHP strings.*

Double Quoted

In PHP, we can specify string through enclosing text within double quote also. But escape sequences and variables will be interpreted using double quote PHP strings.

Example 1

```php
<?php
$str="Hello text within double quote";
echo $str;
?>
```

Output:

```
Hello text within double quote
```

Now, you **can't use double quote directly** inside double quoted string.

Example 2

```php
<?php
$str1="Using double "quote" directly inside double quoted string";
echo $str1;
?>
```

Output:

```
Parse error: syntax error, unexpected 'quote' (T_STRING) in C:\wamp\www\string1.php on line 2
```

We **can store multiple line text, special characters and escape sequences** in a double quoted PHP string.

Example 3

```php
<?php
$str1="Hello text
multiple line
text within double quoted string";
$str2="Using double \"quote\" with backslash inside double quoted string";
$str3="Using escape sequences \n in double quoted string";
echo "$str1 <br/> $str2 <br/> $str3";
?>
```

Output:

```
Hello text multiple line text within double quoted string
Using double "quote" with backslash inside double quoted string
Using escape sequences in double quoted string
```

In double quoted strings, **variable will be interpreted**.

Example 4

```php
<?php
$num1=10;
echo "Number is: $num1";
?>
```

Output:

```
Number is: 10
```

PHP String Functions

PHP provides various string functions to access and manipulate strings.

A list of PHP string functions are given below.

addcslashes()	It is used to return a string with backslashes.
addslashes()	It is used to return a string with backslashes.
bin2hex()	It is used to converts a string of ASCII characters to hexadeci values.
chop()	It removes whitespace or other characters from the right end

	string
chr()	It is used to return a character from a specified ASCII value.
chunk_split()	It is used to split a string into a series of smaller parts.
convert_cyr_string()	It is used to convert a string from one Cyrillic character-set another.
convert_uudecode()	It is used to decode a uuencoded string.
convert_uuencode()	It is used to encode a string using the uuencode algorithm.
count_chars()	It is used to return information about characters used in a strin
crc32()	It is used to calculate a 32-bit CRC for a string.
crypt()	It is used to create hashing string One-way.
echo()	It is used for output one or more strings.
explode()	It is used to break a string into an array.
fprint()	It is used to write a formatted string to a stream.
get_html_translation_table()	Returns translation table which is used by htmlspecialchars() htmlentities().
hebrev()	It is used to convert Hebrew text to visual text.
hebrevc()	It is used to convert Hebrew text to visual text and new lines (into .
hex2bin()	It is used to convert string of hexadecimal values to AS characters.
htmlentities()	It is used to convert character to HTML entities.
html_entity_decode()	It is used to convert HTML entities to characters.

htmlspecialchars()	Converts the special characters to html entities.
htmlspecialchars_decode()	Converts the html entities back to special characters.
Implode()	It is used to return a string from the elements of an array.
Join()	It is the Alias of implode() function.
Levenshtein()	It is used to return the Levenshtein distance between two string
Lcfirst()	It is used to convert the first character of a string to lowercase.
localeconv()	Get numeric formatting information
ltrim()	It is used to remove whitespace from the left side of a string.
md5()	It is used to calculate the MD5 hash of a string.
md5_files()	It is used to calculate MD5 hash of a file.
metaphone()	It is used to calculate the metaphone key of a string.
money_format()	It is used to return a string formatted as a currency string.
nl2br()	It is used to insert HTML line breaks in front of each newline i string.
nl_langinfo()	Query language and locale information
number_format()	It is used to format a number with grouped thousands.
ord()	It is used to return ASCII value of the first character of a string
parse_str()	It is used to parse a query string into variables.
print()	It is used for output one or more strings.
printf()	It is used to show output as a formatted string.

quoted_printable_decode()	Converts quoted-printable string to an 8-bit string
quoted_printable_encode()	Converts the 8-bit string back to quoted-printable string
quotemeta()	Quote meta characters
rtrim()	It is used to remove whitespace from the right side of a string.
setlocale()	It is used to set locale information.
sha1()	It is used to return the SHA-1 hash of a string.
sha1_file()	It is used to return the SHA-1 hash of a file.
similar_text()	It is used to compare the similarity between two strings.
Soundex()	It is is used to calculate the soundex key of a string.
sprintf()	Return a formatted string
sscanf()	It is used to parse input from a string according to a format.
strcasecmp()	It is used to compare two strings.
strchr()	It is used to find the first occurrence of a string inside another string.
strcmp()	Binary safe string comparison (case-sensitive)
strcoll()	Locale based binary comparison(case-sensitive)
strcspn()	It is used to reverses a string.
stripcslashes()	It is used to unquote a string quoted with addcslashes().
stripos()	It is used to return the position of the first occurrence of a string inside another string.
stristr()	Case-insensitive strstr

strlen()	It is used to return the length of a string.
strncasecmp()	Binary safe case-insensitive string comparison
strnatcasecmp()	It is used for case-insensitive comparison of two strings using "natural order" algorithm
strnatcmp()	It is used for case-sensitive comparison of two strings using "natural order" algorithm
strncmp()	It is used to compare of the first n characters.
strpbrk()	It is used to search a string for any of a set of characters.
strripos()	It finds the position of the last occurrence of a case-insensitive substring in a string.
strrpos()	It finds the length of the last occurrence of a substring in a string.
strpos()	It is used to return the position of the first occurrence of a string inside another string.
strrchr()	It is used to find the last occurrence of a string inside another string.
strrev()	It is used to reverse a string.
strspn()	Find the initial length of the initial segment of the string
strstr()	Find the occurrence of a string.
strtok()	Splits the string into smaller strings
strtolower()	Convert the string in lowercase
strtoupper()	Convert the strings in uppercase
strtr()	Translate certain characters in a string or replace the substring
str_getcsv()	It is used to parse a CSV string into an array.

str_ireplace()	It is used to replace some characters in a string (case insensitive).
str_pad()	It is used to pad a string to a new length.
str_repeat()	It is used to repeat a string a specified number of times.
str_replace()	It replaces all occurrences of the search string with replacement string.
str_rot13()	It is used to perform the ROT13 encoding on a string.
str_shuffle()	It is used to randomly shuffle all characters in a string.
str_split()	It is used to split a string into an array.
strcoll()	It is locale based string comparison.
strip_tags()	It is used to strip HTML and PHP tags from a string.
str_word_count()	It is used to count the number of words in a string.
substr()	Return the part of a string
substr_compare()	Compares two strings from an offset up to the length characters. (Binary safe comparison)
substr_count()	Count the number of times occurrence of a substring
substr_replace()	Replace some part of a string with another substring
trim()	Remove whitespace or other characters from the beginning and end of the string.
ucfirst()	Make the first character of the string to uppercase
ucwords()	Make the first character of each word in a string to uppercase
vfprintf()	Write a formatted string to a stream

vprintf()	Display the output as a formatted string according to format
vsprintf()	It returns a formatted string
wordwrap()	Wraps a string to a given number of characters

PHP String Function Examples

1) PHP strtolower() function

The strtolower() function returns string in lowercase letter.

Syntax: string strtolower (string $string)

Example

```php
<?php
$str="My name is KHAN";
$str=strtolower($str);
echo $str;
?>
```

Output:

```
my name is khan
```

2) PHP strtoupper() function

The strtoupper() function returns string in uppercase letter.

Syntax: string strtoupper (string $string)

Example

```php
<?php
$str="My name is KHAN";
$str=strtoupper($str);
echo $str;
?>
```

Output:

```
MY NAME IS KHAN
```

3) PHP ucfirst() function

The ucfirst() function returns string converting first character into uppercase. It doesn't change the case of other characters.

Syntax: string ucfirst (string $str)

Example

```php
<?php
$str="my name is KHAN";
$str=ucfirst($str);
echo $str;
?>
```

Output:

```
My name is KHAN
```

4) PHP lcfirst() function

The lcfirst() function returns string converting first character into lowercase. It doesn't change the case of other characters.

Syntax: string lcfirst (string $str)

Example

```php
<?php
$str="MY name IS KHAN";
$str=lcfirst($str);
echo $str;
?>
```

Output:

```
mY name IS KHAN
```

5) PHP ucwords() function

The ucwords() function returns string converting first character of each word into uppercase.

Syntax: string ucwords (string $str)

Example

```php
<?php
$str="my name is Sonoo jaiswal";
```

```php
$str=ucwords($str);
echo $str;
?>
```

Output:

My Name Is Sonoo Jaiswal

6) PHP strrev() function

The strrev() function returns reversed string.

Syntax: string strrev (string $string)

Example

```php
<?php
$str="my name is Sonoo jaiswal";
$str=strrev($str);
echo $str;
?>
```

Output:

lawsiajoonoSsiemanym

7) PHP strlen() function

The strlen() function returns length of the string.

Syntax: int strlen (string $string)

Example

```php
<?php
$str="my name is Sonoo jaiswal";
$str=strlen($str);
echo $str;
?>
```

Output:

24

PHP Form Handling

We can create and use forms in PHP. To get form data, we need to use PHP superglobals $_GET and $_POST.

The form request may be get or post. To retrieve data from get request, we need to use $_GET, for post request $_POST.

PHP Get Form

Get request is the default form request. The data passed through get request is visible on the URL browser so it is not secured. You can send limited amount of data through get request.

Let's see a simple example to receive data from get request in PHP.

File: form1.html
```
<form action="welcome.php" method="get">
Name: <input type="text" name="name"/>
<input type="submit" value="visit"/>
</form>
```
File: welcome.php
```
<?php
$name=$_GET["name"];//receiving name field value in $name variable
echo "Welcome, $name";
?>
```

PHP Post Form

Post request is widely used to submit form that have large amount of data such as file upload, image upload, login form, registration form etc.

The data passed through post request is not visible on the URL browser so it is secured. You can send large amount of data through post request.

Let's see a simple example to receive data from post request in PHP.

File: form1.html
```
<form action="login.php" method="post">
<table>
<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
</table>
</form>
```
File: login.php
```
<?php
$name=$_POST["name"];//receiving name field value in $name variable
$password=$_POST["password"];//receiving password field value in $password variable

echo "Welcome: $name, your password is: $password";
?>
```

PHP - A Simple HTML Form

The example below displays a simple HTML form with two input fields and a submit button:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="username"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

User Name:

E-mail:

Submit

When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.

To display the submitted data you could simply echo all the variables. The "welcome.php" looks like this:

```
<html>
<body>

Welcome <?php echo $_POST["username"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

The output could be something like this:

Welcome John
Your email address is john.doe@example.com

The same result could also be achieved using the HTTP GET method:

Example

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="username"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
```

```html
</body>
</html>
```

Name: [ ]
E-mail: [ ]

[Submit]

and "welcome_get.php" looks like this:

```html
<html>
<body>

Welcome <?php echo $_GET["username"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

GET vs. POST

Both GET and POST create an array (e.g. array(key1 => value1, key2 => value2, key3 => value3, ...)). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as $_GET and $_POST. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

$_GET is an array of variables passed to the current script via the URL parameters.

$_POST is an array of variables passed to the current script via the HTTP POST method.

When to use GET?

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

Developers prefer POST for sending form data.

Example 1: GET METHOD

```
Welcome
 <?php echo $_GET["username"]; ?>
<br>
Your email address is:
<?php echo $_GET["email"]; ?>

<html>
<body>

<form action="<?php $_PHP_SELF ?>" method="get">
Name: <input type="text" name="username"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Example 2: POST METHOD

```
Welcome
 <?php echo $_POST["username"]; ?>
<br>

Your email address is:
<?php echo $_POST["email"]; ?>

<html>
<body>

<form action="<?php $_PHP_SELF ?>" method="post">
Name: <input type="text" name="username"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

Welcome hhh
Your email address is: dfdfs
Name: [   ]
E-mail: [   ]
Submit

PHP Form Validation

The HTML form we will be working at in these chapters, contains various input fields: required and optional text fields, radio buttons, and a submit button:

PHP Form Validation Example

* required field

Name: [   ] *

E-mail: [   ] *

Website: [   ]

Comment: [   ]

Gender: ○ Female ○ Male ○ Other *

Submit

Your Input:

The validation rules for the form above are as follows:

he validation rules for the form above are as follows:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)

Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

First we will look at the plain HTML code for the form:

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
```

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

```
Gender:
<input type="radio" name="gender" value="female">Female
<input type="radio" name="gender" value="male">Male
<input type="radio" name="gender" value="other">Other
```

The Form Element

The HTML code of the form looks like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

When the form is submitted, the form data is sent with method="post".

What is the $_SERVER["PHP_SELF"] variable?

The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.

So, the $_SERVER["PHP_SELF"] sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the htmlspecialchars() function?

The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with < and >. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

Validate Form Data With PHP

What is Validation ?

Validation means check the input submitted by the user. There are two types of validation are available in PHP. They are as follows –

- **Client-Side Validation** – Validation is performed on the client machine web browsers.

- **Server Side Validation** – After submitted by data, The data has sent to a server and perform validation checks in server machine.

The first thing we will do is to pass all variables through PHP's htmlspecialchars() function.

When we use the htmlspecialchars() function; then if a user tries to submit the following in a text field:

<script>location.href('http://www.hacked.com')</script>

- this would not be executed, because it would be saved as HTML escaped code, like this:

<script>location.href('http://www.hacked.com')</script>

The code is now safe to be displayed on a page or inside an e-mail.

We will also do two more things when the user submits the form:

1. Strip unnecessary characters (extra space, tab, newline) from the user input data (with the PHP trim() function)
2. Remove backslashes (\) from the user input data (with the PHP stripslashes() function)

The next step is to create a function that will do all the checking for us (which is much more convenient than writing the same code over and over again).

We will name the function test_input().

Now, we can check each $_POST variable with the test_input() function, and the script looks like this:

Example 3: FORM VALIDATION

<form.php>

```
<html>
<head>
<title>PHP Form Validation</title>
</head>
<body>
<?php
```

```php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
$name = test_input($_POST["name"]);
$email = test_input($_POST["email"]);
$website = test_input($_POST["website"]);
$comment = test_input($_POST["comment"]);
$gender = test_input($_POST["gender"]);
}
function test_input($data) {
$data = trim($data);
$data = stripslashes($data);
$data = htmlspecialchars($data);
return $data;
}
?>
<h2>Student Registration</h2>
<form method = "post" action = "<?php $_PHP_SELF ?>">
<table>
<tr>
<td>Name:</td>
<td><input type = "text" name = "name"></td>
</tr>
<tr>
<td>E-mail:</td>
<td><input type = "text" name = "email"></td>
</tr>
<tr>
<td>Specific Time:</td>
<td><input type = "text" name = "website"></td>
</tr>
<tr>
<td>Class details:</td>
<td><textarea name = "comment" rows = "5" cols =
"40"></textarea></td>
</tr>
<tr>
<td>Gender:</td>
<td>
<input type = "radio" name = "gender" value = "female">Female
<input type = "radio" name = "gender" value = "male">Male
</td>
</tr>
<tr>
<td>
<input type = "submit" name = "submit" value = "Submit">
```

```
</td>
</tr>
</table>
</form>
<?php
echo "<h2>Your Given details are as :</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>
```

PHP – COMPLETE FORM

This page explains about time real-time form
with actions. Below example will take input
fields as text, radio button, drop down menu,
and checked box.

Example 4: FORM VALIDATION WITH ACTIONS

```
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";
```

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
if (empty($_POST["name"])) {
$nameErr = "Name is required";
}else {
$name = test_input($_POST["name"]);
}
if (empty($_POST["email"])) {
$emailErr = "Email is required";
}else {
$email = test_input($_POST["email"]);
// check if e-mail address is well-formed
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
$emailErr = "Invalid email format";
}
}
if (empty($_POST["website"])) {
$website = "";
}else {
$website = test_input($_POST["website"]);
}
if (empty($_POST["comment"])) {
$comment = "";
}else {
$comment = test_input($_POST["comment"]);
}
if (empty($_POST["gender"])) {
$genderErr = "Gender is required";
}else {
$gender = test_input($_POST["gender"]);
}
}
function test_input($data) {
$data = trim($data);
$data = stripslashes($data);
$data = htmlspecialchars($data);
return $data;
}
?>
<h2>Student registration</h2>
<p><span class = "error">* required field.</span></p>
<form method = "post" action = "<?php
echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
<table>
<tr>
<td>Name:</td>
<td><input type = "text" name = "name">
```

```
<span class = "error">* <?php echo $nameErr;?></span>
</td>
</tr>
<tr>
<td>E-mail: </td>
<td><input type = "text" name = "email">
<span class = "error">* <?php echo $emailErr;?></span>
</td>
</tr>
<tr>
<td>Time:</td>
<td> <input type = "text" name = "website">
<span class = "error"><?php echo $websiteErr;?></span>
</td>
</tr>
<tr>
<td>Classes:</td>
<td> <textarea name = "comment" rows = "5" cols =
"40"></textarea></td>
</tr>
<tr>
<td>Gender:</td>
<td>
<input type = "radio" name = "gender" value = "female">Female
<input type = "radio" name = "gender" value = "male">Male
<span class = "error">* <?php echo $genderErr;?></span>
</td>
</tr>
<td>
<input type = "submit" name = "submit" value = "Submit">
</td>
</table>
</form>
<?php
echo "<h2>Your given values are as:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
```

</html>

Student registration

* required field.

Name: [_____] *
E-mail: [_____] *
Time: [_____]

Classes: [_____]

Gender: ○ Female ○ Male *
[Submit]

Your given values are as:

raj kumar k
raju@gmail.com
4:30
II CSE
male

PHP Cookie

PHP cookie is a small piece of information which is stored at client browser. It is used to recognize the user.

Cookie is created at server side and saved to client browser. Each time when client sends request to the server, cookie is embedded with request. Such way, cookie can be received at the server side.
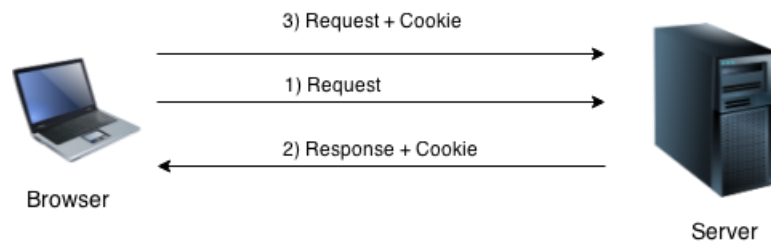
In short, cookie can be created, sent and received at server end.

Note: PHP Cookie must be used before <html> tag

PHP setcookie() function

PHP setcookie() function is used to set cookie with HTTP response. Once cookie is set, you can access it by $_COOKIE superglobal variable.

Syntax :
bool setcookie (string $name [, string $value [, int $expire = 0 [, string $path [, string $domain [, bool $secure = false [, bool $httponly = false]]]]]])

Syntax :   setcookie(*name, value, expire, path, domain, secure, httponly*);

Only the *name* parameter is required. All other parameters are optional.

Example:
setcookie("CookieName", "CookieValue");/* defining name and value only*/
setcookie("CookieName", "CookieValue", 84600); // one day i.e. 84600 seconds
//using expiry in 1 hour(1*60*60 seconds or 3600 seconds) 24*60*60
setcookie("CookieName", "CookieValue", time()+24*60*60, "/mypath/", "mydomain.com", 1);

PHP $_COOKIE

PHP $_COOKIE superglobal variable is used to get cookie.

PHP Cookie Example

File: cookie1.php

        <?php

```php
setcookie("user", "Raju");  //name is user and value is raju
?>

<html>
<body>

<?php
if(!isset($_COOKIE["user"])) {     //consider value is ramu
   echo "Sorry, cookie is not found!";
} else {
   echo "<br/>Cookie Value: " . $_COOKIE["user"];
}
?>
</body>
</html>
```

Output:

Sorry, cookie is not found!

Firstly cookie is not set. But, if you *refresh* the page, you will see cookie is set now.

Output:

Cookie Value: Raju

Example 2:

File: cookie2.php

```php
<?php
$cookie_name = "user";
$cookie_value = "Raju";
setcookie($cookie_name, $cookie_value, time() +
(86400 * 30), "/"); // 86400 = 1 day  now total expire time is 30
days
?>


<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_value];
}
?>
```

```
</body>
</html>
```

PHP Delete Cookie

If you set the expiration date in past, cookie will be deleted.
File: cookie3.php

```
<?php
setcookie ("CookieName", "", time() - 3600);
// set the expiration date to one hour ago
?>
<html>

<head>
<title>Deleting Cookies with PHP</title>
</head>

<body>
<?php echo "Deleted Cookies" ?>
</body>

</html>
```

PHP Session

PHP session is used to store and pass information from one page to another temporarily (until user close the website).

PHP session technique is widely used in shopping websites where we need to store and pass cart information e.g. username, product code, product name, product price etc from one page to another.

PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.
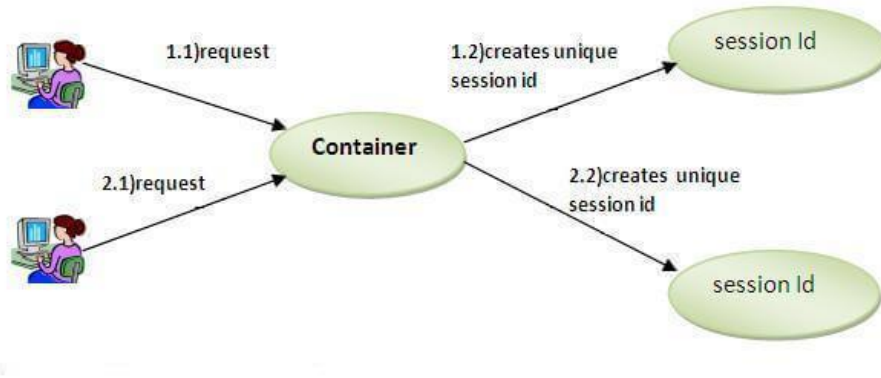
PHP session_start() function

PHP session_start() function is used to start the session. It starts a new or resumes existing session. It returns existing session if session is created already. If session is not available, it creates and returns new session.

Syntax : bool session_start (void)

Example: session_start();

PHP $_SESSION

PHP $_SESSION is an associative array that contains all session variables. It is used to set and get session variable values.

Example: Store information : $_SESSION["user"] = "Sachin";

Example: Get information : echo $_SESSION["user"]; //Sachin

PHP Session Example

File: session1.php

```php
<?php
session_start();
?>

<html>
<body>

<?php
$_SESSION["user"] = "Ramu";  //only one session
// $_SESSION["user2"] = "Raju";  2nd session

echo "Session information are set successfully.<br/>";
?>
```

```
<a href="session2.php">Visit next page</a>

</body>
</html>
```

File: session2.php

```php
<?php
session_start();
?>
<html>
<body>
<?php
echo "User is: ".$_SESSION["user"];  //get information   i.e. Sachin
?>
</body>
</html>
```

PHP Session Counter Example

File: sessioncounter.php

```php
<?php
session_start();

if (!isset($_SESSION['counter'])) {
    $_SESSION['counter'] = 1;
} else {
    $_SESSION['counter']++;
}
echo ("Page Views: ".$_SESSION['counter']);
?>
```

PHP Destroying Session

PHP session_destroy() function is used to destroy all session variables completely.

File: session3.php

```php
<?php
session_start();
session_destroy();
?>
```

PHP File Handling

PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

PHP Open File - fopen()

The PHP fopen() function is used to open a file.

Syntax:
fopen (string $filename , string $mode [, bool $use_include_path = false [, resource $context]])

Example:

```php
<?php
$fp = fopen("c:\\folder\\file.txt", "r");
?>
```

PHP Close File - fclose()

The PHP fclose() function is used to close an open file pointer.

Syntax: bool fclose (resource $handle)

Example

```php
<?php
fclose($fp);
?>
```

PHP Read File - fread()

The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.

Syntax: string fread (resource $handle , int $length)

Example

```php
<?php
$filename = "readfile.txt";  //this file is in the current directory

$fp = fopen($filename, "r");//open file in read mode

$contents = fread($fp, filesize($filename));//open     and read file

echo $contents;//printing data of file
fclose($fp);//close file
?>
```

Output

PHP Write File - fwrite()

The PHP fwrite() function is used to write content of the string into file.

Syntax: int fwrite (resource $handle , string $string [, int $length])

Example

```php
<?php
$fp = fopen('writefile.txt', 'w');//open file in write mode
fwrite($fp, 'hello ');
fwrite($fp, 'php file');
fwrite($fp, 'welcome');

fclose($fp);

echo "File written successfully";
?>
```

Output

PHP Delete File - unlink()

The PHP unlink() function is used to delete file.

Syntax: bool unlink (string $filename [, resource $context])

Example

```php
<?php
unlink('data.txt');

echo "File deleted successfully";
?>
```

PHP Open File

PHP fopen() function is used to open file or URL and returns resource. The fopen() function accepts two arguments: $filename and $mode. The $filename represents the file to be opened and $mode represents the file mode for example read-only, read-write, write-only etc.

Syntax:
resource fopen (string $filename , string $mode [, bool $use_include_path = false [, resource $context]])

PHP Open File Mode

Mode	Description
r	Opens file in **read-only** mode. It places the file pointer at the beginning of the file.
r+	Opens file in **read-write** mode. It places the file pointer at the beginning of the file.
w	Opens file in **write-only** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
w+	Opens file in **read-write** mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
a	Opens file in **write-only** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
a+	Opens file in **read-write** mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
x	Creates and opens file in **write-only** mode. It places the file pointer at the beginning of the file. If file is found, fopen() function returns FALSE.
x+	It is same as x but it creates and opens file in **read-write** mode.
c	Opens file in **write-only** mode. If the file does not exist, it is created. If it exists, it is neither truncated (as opposed to 'w'), nor the call to this function fails (as is the case with 'x'). The file pointer is positioned on the beginning of the file
c+	It is same as c but it opens file in **read-write** mode.

PHP Open File Example

```php
<?php
$handle = fopen("c:\\folder\\file.txt", "r");
?>
```

PHP Read File

PHP provides various functions to read data from file. There are different functions that allow you to read all file data, read data line by line and read data character by character.

The available PHP file read functions are given below.

- o   fread()
- o   fgets()
- o   fgetc()

PHP Read File - fread()

The PHP fread() function is used to read data of the file. It requires two arguments: file resource and file size.

Syntax: string fread (resource $handle , int $length)

$handle represents file pointer that is created by fopen() function.

$length represents length of byte to be read.

Example

```php
<?php
$filename = "c:\\file1.txt";
$fp = fopen($filename, "r");//open file in read mode

$contents = fread($fp, filesize($filename));//read file

echo "<pre>$contents</pre>";//printing data of file
fclose($fp);//close file
?>
```

Output:

```
this is first line
this is another line
this is third line
```

PHP Read File - fgets()

The PHP fgets() function is used to read single line from the file.

Syntax: string fgets (resource $handle [, int $length])

Example

```php
<?php
$fp = fopen("c:\\file1.txt", "r");//open file in read mode
echo fgets($fp);
fclose($fp);
?>
```

Output

```
this is first line
```

PHP Read File - fgetc()

The PHP fgetc() function is used to read single character from the file. To get all data using fgetc() function, use !feof() function inside the while loop.

Syntax: string fgetc (resource $handle)

Example

```php
<?php
$fp = fopen("c:\\file1.txt", "r");//open file in read mode
while(!feof($fp)) {
  echo fgetc($fp);
}
fclose($fp);
?>
```

Output

```
this is first line this is another line this is third line
```

PHP Write File

PHP fwrite() and fputs() functions are used to write data into file. To write data into file, you need to use w, r+, w+, x, x+, c or c+ mode.

PHP Write File - fwrite()

The PHP fwrite() function is used to write content of the string into file.

Syntax: int fwrite (resource $handle , string $string [, int $length])

Example:

```php
<?php
$fp = fopen('data.txt', 'w');//opens file in write-only mode
fwrite($fp, 'welcome ');
fwrite($fp, 'to php file write');
fclose($fp);

echo "File written successfully";
?>
```

Output: data.txt

```
welcome to php file write
```

PHP Overwriting File

If you run the above code again, it will erase the previous data of the file and writes the new data. Let's see the code that writes only new data into data.txt file.

```php
<?php
$fp = fopen('data.txt', 'w');//opens file in write-only mode
fwrite($fp, 'hello');
fclose($fp);

echo "File written successfully";
?>
```

Output: data.txt

```
hello
```

PHP Append to File

If you use **a** mode, it will not erase the data of the file. It will write the data at the end of the file. Visit the next page to see the example of appending data into file

PHP Append to File

You can append data into file by using a or a+ mode in fopen() function. Let's see a simple example that appends data into data.txt file.

Let's see the data of file first.

data.txt

PHP Append to File - fwrite()

The PHP fwrite() function is used to write and append data into file.

Example

```php
<?php
$fp = fopen('data.txt', 'a');//opens file in append mode
fwrite($fp, ' this is additional text ');
fwrite($fp, 'appending data');
fclose($fp);

echo "File appended successfully";
?>
```

Output: data.txt

welcome to php file write this is additional text appending data

PHP Delete File

In PHP, we can delete any file using unlink() function. The unlink() function accepts one argument only: file name. It is similar to UNIX C unlink() function.

PHP unlink() generates E_WARNING level error if file is not deleted. It returns TRUE if file is deleted successfully otherwise FALSE.

Syntax: bool unlink (string $filename [, resource $context])

$filename represents the name of the file to be deleted.

PHP Delete File Example

```php
<?php
$status=unlink('data.txt');    //true
if($status)  //if the status is true then file is deleted
{
echo "File deleted successfully";
}else{
echo "Sorry!";   // if you want to delete the file that file is not exist in the current directory
}
?>
```

Output

File deleted successfully

PHP: LIST ALL FILES IN A DIRECTORY

It deals with the process of how to list all files in a directory using PHP. We will do this   using PHP's glob function, which allows us to retrieve a list of file pathnames that match a certain pattern.

For this example, I have created a folder called "test". Inside the folder, I have created three files:

- test.txt
- names.txt
- file.php

Here is a screenshot of the directory:
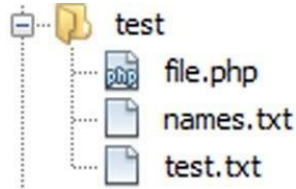

```php
<?php
//Get a list of file paths using the glob function.
$fileList = glob('test/*'); // test is the folder or directory … wrt this directory how many files are available
//Loop through the array that glob returned.    Here * indicates all the types of files
foreach($fileList as $filename)
{
//Simply print them out onto the screen.
echo $filename, '<br>';
}
```

The result will look something like this:

1   test/file.php
2   test/names.txt
3   test/test.txt

However, what if we wanted to list all files with a particular file extension? i.e. What if we only want to list the .txt files and not the **.php** file that is currently present?

// $fileList = glob('test/*');

Well, the solution is pretty simple:

```
//Get a list of all files
ending in .txt
$fileList = glob('test/*.txt');
$fileList = glob('test/*.php');  //display only php files
```

In the code snippet above, we told the glob function to return a list of file pathnames that ended .txt

Warning: In some cases, the folder may have subdirectories. In cases where you are listing everything that is inside a specified folder, these subdirectories will be returned by the glob function. To avoid printing out or interacting with subdirectories, you can simply use the is_file function to confirm that the file pathname in question leads to an actual file:

```php
<?php
$fileList = glob('test/*');
foreach($fileList as $filename){
//Use the is_file function to make sure that it is not a directory.
if(is_file($filename))
{
echo $filename, '<br>';
}
}
```

PHP File Upload

PHP allows you to upload single and multiple files through few lines of code only.

PHP file upload features allows you to upload binary and text files both. Moreover, you can have the full control over the file to be uploaded through PHP authentication and file operation functions.

PHP $_FILES

The PHP global $_FILES contains all the information of file. By the help of $_FILES global, we can get file name, file type, file size, temp file name and errors associated with file.

Here, we are assuming that file name is *filename*.

$_FILES['filename']['name']   **returns file name.**

$_FILES['filename']['type']   **returns MIME type of the file.**

$_FILES['filename']['size']   **returns size of the file (in bytes).**

$_FILES['filename']['tmp_name']   **returns temporary file name of the file which was stored on the server.**

$_FILES['filename']['error']   **returns error code associated with this file.**

move_uploaded_file() function

The move_uploaded_file() function moves the uploaded file to a new location. The move_uploaded_file() function checks internally if the file is uploaded thorough the POST request. It moves the file if it is uploaded through the POST request.

Syntax: bool move_uploaded_file (string $filename , string $destination)

PHP File Upload Example

File: uploadform.html

```html
<form action="uploader.php" method="post">
   Select File:
   <input type="file" name="fileToUpload"/>
   <input type="submit" value="Upload Image" name="submit"/>
</form>
```
File: uploader.php

```php
<?php
$target_path = "e:/";
$target_path = $target_path.basename( $_FILES['fileToUpload']['name']);

if(move_uploaded_file($_FILES['fileToUpload']['tmp_name'], $target_path)) {
    echo "File uploaded successfully!";
} else{
    echo "Sorry, file not uploaded, please try again!";
}
?>
```

PHP MySQL Connect

PHP mysqli_connect()

PHP **mysqli_connect() function** is used to connect with MySQL database. It returns *resource* if connection is established or *null*.

Syntax: resource mysqli_connect (server, username, password)

PHP mysqli_close()

PHP **mysqli_close() function** is used to disconnect with MySQL database. It returns *true* if connection is closed or *false*.

Syntax: bool mysqli_close(resource $resource_link)

PHP MySQL Connect Example

```php
<?php

$servername = "localhost";

$username = "root";

$password = "";

// Create connection

$conn = new mysqli($servername, $username, $password);

// Check connection

if ($conn->connect_error) {

  die("Connection failed: " . $conn->connect_error);

}

echo "Connected successfully";
```

```php
?>
```

Structured connection

```php
<?php
$servername = "localhost";

$username = "root";

$password = "";

$con = mysqli_connect("$servername","$username","$password");

// Check connection

if (!$con)

 {

 die("sorry we failed to connect:".mysqli_connect_error());

}

else

{

 echo "Connection was successful ";

}
?>
```

pdo connection

```php
<?php
$servername = "localhost";

$username = "root";

$password = "";


try {

  $conn = new PDO("mysql:host=$servername;dbname=test", $username, $password);

  // set the PDO error mode to exception

  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  echo "Connected successfully";

} catch(PDOException $e) {
```

```php
    echo "Connection failed: " . $e->getMessage();
}
?>
```

PHP MySQLi Create Table Example

Example

```php
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'testdb';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = "create table emp5(id INT AUTO_INCREMENT,name VARCHAR(20)
NOT NULL, emp_salary INT NOT NULL,primary key (id))";
if(mysqli_query($conn, $sql)){
 echo "Table emp5 created successfully";
}else{
echo "Could not create table: ". mysqli_error($conn);
}

mysqli_close($conn);
?>
```

PHP MySQLi Insert Record Example

Example

```php
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = 'INSERT INTO emp4(name,salary) VALUES ("sonoo", 9000)';
if(mysqli_query($conn, $sql)){
 echo "Record inserted successfully";
}else{
echo "Could not insert record: ". mysqli_error($conn);
}
```

```php
        mysqli_close($conn);
        ?>
```

PHP MySQLi Update Record Example

Example

```php
        <?php
        $host = 'localhost:3306';
        $user = '';
        $pass = '';
        $dbname = 'test';

        $conn = mysqli_connect($host, $user, $pass,$dbname);
        if(!$conn){
          die('Could not connect: '.mysqli_connect_error());
        }
        echo 'Connected successfully<br/>';

        $id=2;
        $name="Rahul";
        $salary=80000;
        $sql = "update emp4 set name=\"$name\", salary=$salary where id=$id";
        if(mysqli_query($conn, $sql)){
         echo "Record updated successfully";
        }else{
        echo "Could not update record: ". mysqli_error($conn);
        }

        mysqli_close($conn);
        ?>
```

PHP MySQLi Delete Record Example

Example

```php
<?php
$host = 'localhost:3306';
$user = '';
$pass = '';
$dbname = 'test';

$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$id=2;
$sql = "delete from emp4 where id=$id";
if(mysqli_query($conn, $sql)){
 echo "Record deleted successfully";
}else{
echo "Could not deleted record: ". mysqli_error($conn);
}

mysqli_close($conn);
?>
```

PHP MySQL Select Query

There are two other MySQLi functions used in select query.

- o **mysqli_num_rows(mysqli_result $result)**: returns number of rows.
- o **mysqli_fetch_assoc(mysqli_result $result)**: returns row as an associative array. Each key of the array represents the column name of the table. It return NULL if there are no more rows.

PHP MySQLi Select Query Example

Example

```php
<?php
$host = 'localhost';
$user = '';
$pass = '';
$dbname = 'testdb';
$conn = mysqli_connect($host, $user, $pass,$dbname);
if(!$conn){
  die('Could not connect: '.mysqli_connect_error());
}
echo 'Connected successfully<br/>';

$sql = 'SELECT * FROM emp4';
$retval=mysqli_query($conn, $sql);

if(mysqli_num_rows($retval) > 0){
 while($row = mysqli_fetch_assoc($retval)){
   echo "EMP ID :{$row['id']}  <br> ".
       "EMP NAME : {$row['name']} <br> ".
       "EMP SALARY : {$row['salary']} <br> ".
       "------------------------------<br>";
 } //end of while
}else{
echo "0 results";
}
mysqli_close($conn);
?>
```