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 7.4.0 is the latest version of PHP, which was released on 28 November. Some important points need to be noticed about PHP are as followed:

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

Why we learn PHP ?

It is one of the widely used open-source general-purpose scripting language that is used for backend Development. Apart from this, let's see why we should learn it.

Easy to Learn: It is easier to learn for anyone who has come across to any programming language for the first time.
Free of Cost: Since it is an open-source language, therefore developers are allowed to use its components and all methods for free.
Flexible: Since It is a dynamically typed language, therefore there are no hard rules on how to build features using it.

Supports nearly all databases: It supports all the widely used databases, including MySQL, ODBC, SQLite etc.
Secured: It has multiple security levels provides us a secure platform for developing websites as it has multiple security levels.
Huge Community Support: It is loved and used by a huge number of developers. The developers share their knowledge with other people of the community that want to know about it.

Advantages of PHP :

- **Scalable**
  If you're working on a large-scale project and need the help of a scripting language that can get the task done well, then this is the language for you. Facebook, WordPress, Tumblr, Yahoo and others were all created using PHP. Since one of the main advantages of PHP is that it loads fast, it also means the people coming to your website won't experience long wait times.

- **Free to use**
  Scripting languages can be of two types — open source and closed source. Open source language refers to languages that are free; closed source refers to the languages that are paid. PHP is one of the most popular languages on the server side scripting world that doesn't require any fee. You can use frameworks, syntax, libraries, databases, and all of its functionalities for free.

- **Community support**
  One of the biggest advantages of using PHP in web development is that the support of the PHP community is extremely strong. If you're stuck with a problem somewhere in development you can pose a question to the community and you'll have multiple software developers around the world to help you. PHP Community is one of the most active communities in the web development world.

- **Ease of use**
  When it comes to programming languages, PHP is one of the easier ones to learn. This is because even though its syntax is rigid, it is also forgiving. Aside from this obvious benefit, you can also access multiple libraries and resources to ensure that your code is optimized for speed and efficiency.

- **Lightning speed**
  its lightning speed. According to many experts, this programming language is almost three times as fast as Python, and the current versions (#7 and up) are even faster than the previous versions. As per Benchmarks Game's data, a PHP web page takes approximately 17.81 seconds to load.

Disadvantages of PHP :

- **Security:** Since it is open sourced, all people can see the source code. If there are bugs in the source code, it can be used by people to explore the weakness of it.
- **Not suitable of large applications:** It will be difficult to use it for programming huge applications. Since the programming language is not highly modular, huge applications created out of the programming language will be difficult to maintain.

- **Weak type:** Implicit conversion may surprise unwary programmers and lead to unexpected bugs. Confusion between arrays and hash tables. This is slow and could be faster. There are often a few ways to accomplish a task. It is not strongly typed. It is interpreted and uses curly braces.
- **Poor Error Handling Method:** The framework has a bad error handling method. It is not a proper solution for the developers. Therefore, as a qualified PHP developer, you will have to overcome it.
- **PHP is unable to handle large number of apps:** The technology is helpless to support a bunch of apps. It is highly tough to manage because, it is not competent modular. It already imitates the features of Java language.

Installation of 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, 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

PHP Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Basic PHP Syntax

A PHP script can be placed anywhere in the document.

A PHP script starts with <?php and ends with ?>:

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

The default file extension for PHP files is ".php".

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

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

```
<!DOCTYPE html>
```

```
<html>
<body>
<h1>My first PHP page</h1>


<?php
echo "Hello World!";
?>
</body>
</html>
```

Note: *PHP statements end with a semicolon (;).*

PHP Case Sensitivity

In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.

In the example below, all three echo statements below are equal and legal:

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

Note: *However; all variable names are case-sensitive!*

Look at the example below; only the first statement will display the value of the $color variable! This is because $color, $COLOR, and $coLOR are treated as three different variables:

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

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

Comments in PHP

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

Comments can be used to:

- Let others understand your code
- Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

PHP supports several ways of commenting:

1. **Syntax for single-line comments:**

```
<!DOCTYPE html>
<html>
<body>
<?php
// This is a single-line comment
# This is also a single-line comment
?>
</body>
</html>
```

2. Syntax for multiple-line comments:

```
<!DOCTYPE html>
<html>
<body>
<?php
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
?>
</body>
</html>
```

3. Using comments to leave out parts of the code:

```
<!DOCTYPE html>
<html>
<body>
<?php
// You can also use comments to leave out parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>
</body>
</html>
```

What is Variable in PHP

Variables are used to store data, like string of text, numbers, etc. Variable values can change over the course of a script. Here're some important things to know about variables:

- In PHP, a variable does not need to be declared before adding a value to it. PHP automatically converts the variable to the correct data type, depending on its value.
- After declaring a variable, it can be reused throughout the code.
- The assignment operator (=) used to assign value to a variable.

In PHP variable can be declared as: $var_name = value;

Example

```
<?php
// Declaring variables
$txt = "Hello World!";
$number = 10;
// Displaying variables value
echo $txt;  // Output: Hello World!
echo $number; // Output: 10
?>
```

In the above example we have created two variables where first one has assigned with a string value and the second has assigned with a number. Later we've displayed the variables values in the browser using the echo statement. The PHP echo statement is often used to output data to the browser. We will learn more about this in upcoming chapter.

Naming Conventions for PHP Variables

These are the following rules for naming a PHP variable:

- All variables in PHP start with a $ sign, followed by the name of the variable.
- A variable name must start with a letter or the underscore character _.
- A variable name cannot start with a number.
- A variable name in PHP can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- A variable name cannot contain spaces.

Note: Variable names in PHP are case sensitive, it means $x and $X are two different variables. So be careful while defining variable names.

PHP $ and $$ Variables

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

- The $$var (double dollar) is a reference variable that stores the value of the $variable inside it.
- To understand the difference better, let's see some examples.

Example 1

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

PHP Constants

PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants, which are not really constants. PHP constants can be defined by 2 ways:

i. Using define() function
ii. Using const keyword

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

Conventionally, PHP constants should be defined in uppercase letters.

Note: Unlike variables, constants are automatically global throughout the script.

PHP constant: define()

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

define(name, value, case-insensitive)

name: It specifies the constant name.

value: It specifies the constant value.

case-insensitive: Specifies whether a constant is case-insensitive. Default value is false. It means it is case sensitive by default.

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

File: constant1.php

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

PHP constant: const keyword

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

File: constant4.php

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

Output:

Hello const by Edunet PHP

Constant() function

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

Syntax

The syntax for the following constant function:

constant (name)

File: constant5.php

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

Output:

Edunet

Edunet

Constant vs Variables

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

PHP Data Types

Variables can store data of different types, and different data types can do different things.

The values assigned to a PHP variable may be of different data types including simple string and numeric types to more complex data types like arrays and objects.

PHP supports total eight primitive data types: Integer, Floating point number or Float, String, Booleans, Array, Object, resource and NULL. These data types are used to construct variables. Now let's discuss each one of them in detail.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

Integers – are whole numbers, without a decimal point, like 4195.

Doubles – are floating-point numbers, like 3.14159 or 49.1.

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

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

Strings – are sequences of characters, like 'PHP supports string operations.'

Arrays – are named and indexed collections of other values.

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

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

The first five are simple types, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

Integers

They are whole numbers, without a decimal point, like 4195. They are the simplest type they correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so −

```
$int_var = 12345;

$another_int = -12345 + 12345;
```

Doubles

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code

```
<?php
```

```
  $many = 2.2888800;
  $many_2 = 2.2111200;
  $few = $many + $many_2;
  print("$many + $many_2 = $few <br>");
?>
```

It produces the following browser output :

2.28888 + 2.21112 = 4.5

Boolean

They have only two possible values, either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so.

```
if (TRUE)
  print("This will always print<br>");
else
  print("This will never print<br>");
```

Interpreting other types as Booleans

Here are the rules for determining the "truth" of any value not already of the Boolean type

If the value is a number, it is false if exactly equal to zero and true otherwise. *If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.

Values of type NULL are always false.

If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.

Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).

Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

```
$true_num = 3 + 0.14159;
$true_str = "Tried and true"
$true_array[49] = "An array element";
```

```
$false_array = array();
$false_null = NULL;
$false_num = 999 - 999;
$false_str = "";
```

NULL

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this

```
$my_var = NULL;
```

The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed

```
$my_var = null;
```

A variable that has been assigned NULL has the following properties

It evaluates to FALSE in a Boolean context.

It returns FALSE when tested with IsSet() function.

Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = 'This is a somewhat longer, singly quoted string';
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

```
<?php
 $variable = "name";
 $literally = 'My $variable will not print!';
 print($literally);
 print "<br>";
 $literally = "My $variable will print!";
```

```
print($literally);
?>
```

This will produce following result

My $variable will not print!\n

My name will print

Operator

An operator is an element that takes one or more values, (called expressions in programming jargon), and yields another value. This makes the construction itself become an expression. For instance, when you add two numbers, say 2 and 5 (values) it yields 7 (value), such that the construction itself becomes an expression.

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 language supports following type of operators.

i. Arithmetic Operators
ii. Comparison Operators
iii. Logical (or Relational) Operators
iv. Assignment Operators
v. Conditional (or ternary) Operators

Lets have a look on all operators one by one.

Arithmetic Operators

There are following arithmetic operators supported by PHP language −

Assume variable A holds 10 and variable B holds 20 then −

Show Examples

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0

++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Example:

```
<?php
$x = 10;
$y = 4;
echo($x + $y); // 0utputs: 14
echo($x - $y); // 0utputs: 6
echo($x * $y); // 0utputs: 40
echo($x / $y); // 0utputs: 2.5
echo($x % $y); // 0utputs: 2
?>
```

Comparison Operators

There are following comparison operators supported by PHP language

Assume variable A holds 10 and variable B holds 20 then −

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Example:

```
<?php
$x = 25;
$y = 35;
$z = "25";
var_dump ($x == $z);  // Outputs: boolean true
var_dump ($x === $z); // Outputs: boolean false
var_dump ($x != $y);  // Outputs: boolean true
var_dump ($x !== $z); // Outputs: boolean true
var_dump ($x < $y);   // Outputs: boolean true
var_dump ($x > $y);   // Outputs: boolean false
var_dump ($x <= $y);  // Outputs: boolean true
var_dump ($x >= $y);  // Outputs: boolean false

?>
```

Logical Operators

There are following logical operators supported by PHP language

Assume variable A holds 10 and variable B holds 20 then −

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then condition becomes true.	(A and B) is true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Example:

```
<?php
$year = 2014;
// Leap years are divisible by 400 or by 4 but not 100
if(($year % 400 == 0) || (($year % 100 != 0) && ($year % 4 == 0))){
   echo "$year is a leap year.";
} else{
   echo "$year is not a leap year.";
}
?>
```

Assignment Operators

There are following assignment operators supported by PHP language −

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Example:

```
<?php
$x = 10;
```

```
echo $x; // Outputs: 10

$x = 20;
$x += 30;
echo $x; // Outputs: 50

$x = 50;
$x -= 20;
echo $x; // Outputs: 30

$x = 5;
$x *= 25;
echo $x; // Outputs: 125

$x = 50;
$x /= 10;
echo $x; // Outputs: 5

$x = 100;
$x %= 15;
echo $x; // Outputs: 10
?>
```

Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax −

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Operators Categories

All the operators we have discussed above can be categorised into following categories −

- Unary prefix operators, which precede a single operand.
- Binary operators, which take two operands and perform a variety of arithmetic and logical operations.
- The conditional operator (a ternary operator), which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression.
- Assignment operators, which assign a value to a variable.

Precedence of PHP Operators

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator −

For example x = 7 + 3 * 2; Here x is assigned 13, not 20 because operator * has higher precedence than + so it first get multiplied with 3*2 and then adds into 7.

Here operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity
Unary	! ++ --	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=	Right to left

Activity:

Write a program to add two numbers ’10’ and ‘5’

Write a program to subtract number ‘3’ from ’12’

Write a program to subtract number ’10’ from ‘5’

Write a program to multiply two numbers ‘7’ and ’11’

Write a program to divide ’10’ by ‘4’ (10/4) and check the output.

Write a program to check the remainder (using Modulus), when ’10’ divided by ‘4’.

What is a control structure?

Code execution can be grouped into categories as shown below

Sequential – this one involves executing all the codes in the order in which they have been written.

Decision – this one involves making a choice given a number of options. The code executed depends on the value of the condition.

A control structure is a block of code that decides the execution path of a program depending on the value of the set condition.

Let’s now look at some of the control structures that PHP supports.

PHP If Else

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

i. If
ii. if-else
iii. if-else-if
iv. nested if

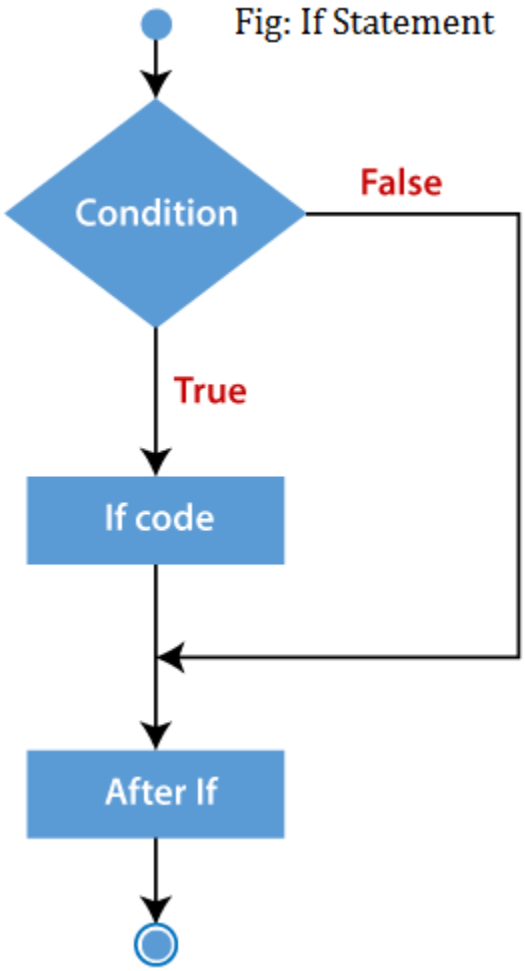
i. PHP If Statement

PHP if statement allows conditional execution of code. It is executed if condition is true.
If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.

Syntax

```
if(condition){

//code to be executed

}
```

Flowchart:



Example

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

Output:

12 is less than 100

ii. PHP If-else Statement

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

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

Syntax

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

Flowchart

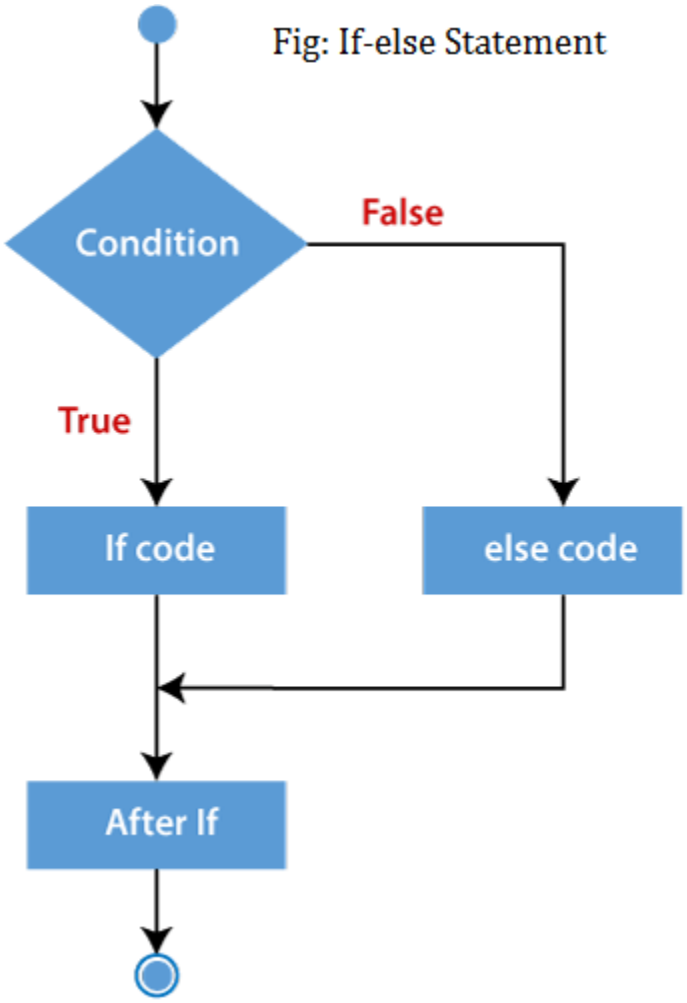


Fig: If-else Statement

Example

```
<?php
$num=12;
```

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

Output:

12 is even number

iii. PHP If-else-if Statement

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

Syntax

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

Flowchart

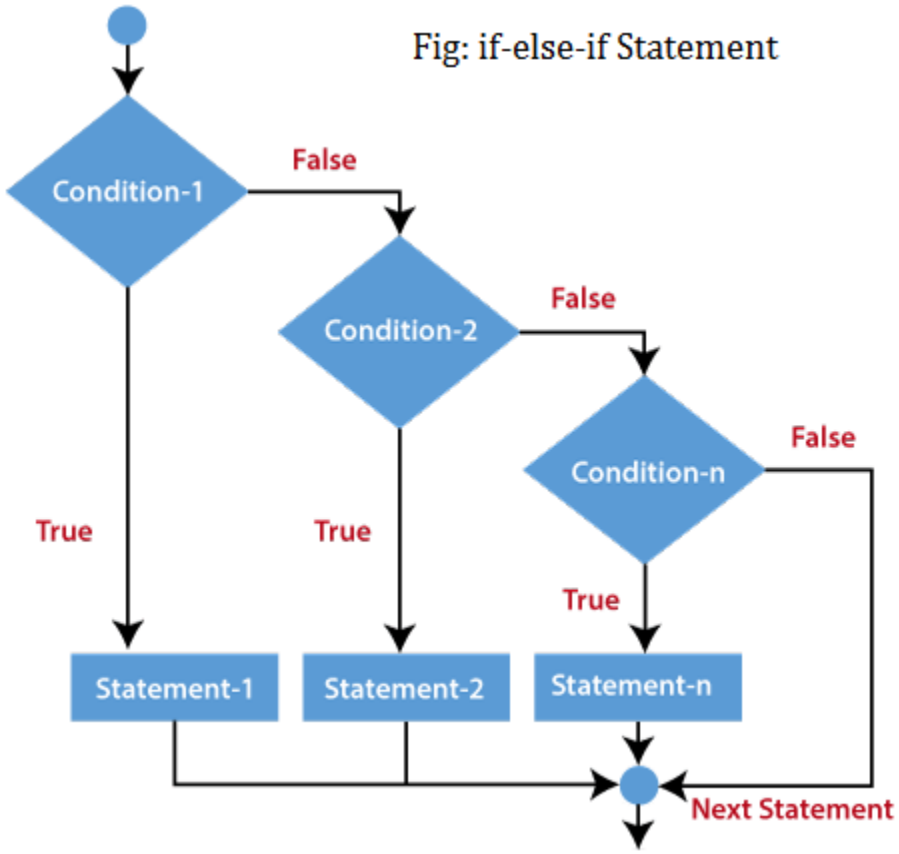


Fig: if-else-if Statement

Example:

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

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

Output:

B Grade

iv. 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

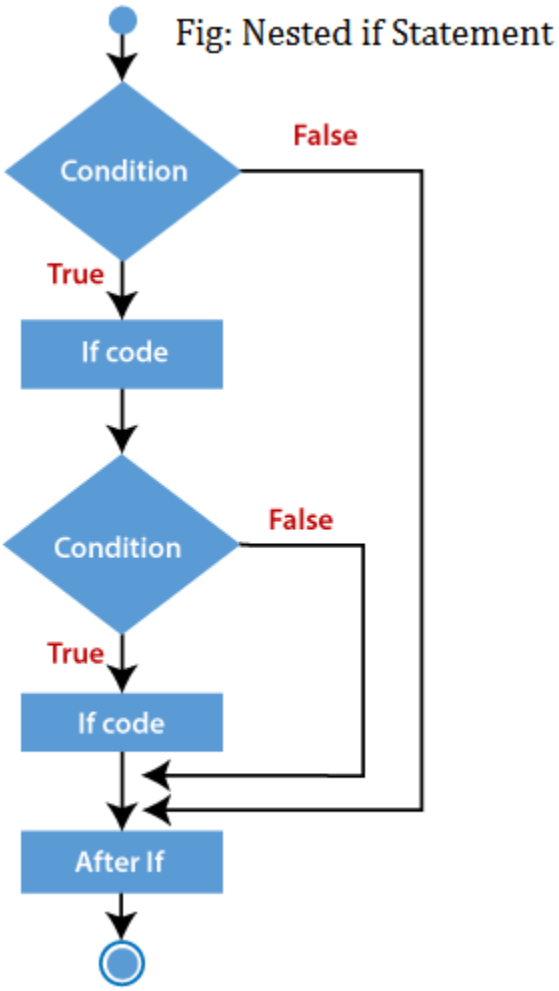


Fig: Nested if Statement

Example

```
<?php
        $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 Case

Switch… case is similar to the if then… else control structure

It only executes a single block of code depending on the value of the condition.

If no condition has been met then the default block of code is executed.

It has the following basic syntax.

```
<?php
switch(condition){
case value:
//block of code to be executed
break;
case value2:
//block of code to be executed
break;
default:
//default block code
break;
}
?>
```

HERE,

“switch(…){…}” is the control structure block code

“case value: case…” are the blocks of code to be executed depending on the value of the condition

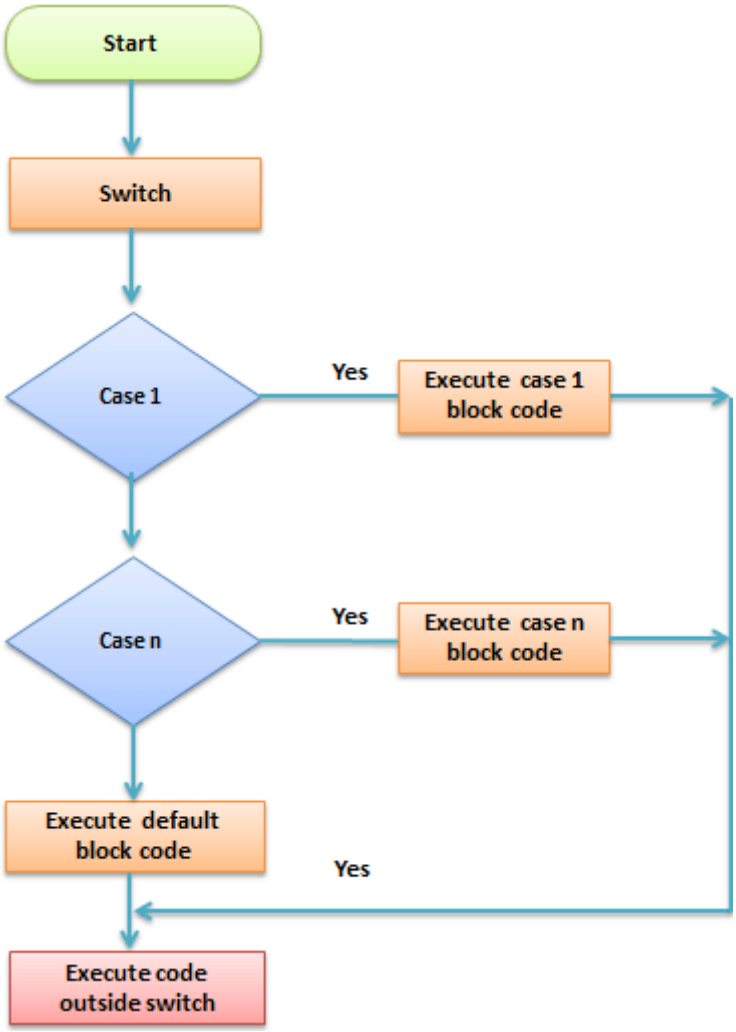
“default:” is the block of code to be executed when no value matches with the condition

Important points to be noticed about switch case:

- The default is an optional statement. Even it is not important, that default must always be the last statement.

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

Flowchart



Example:

```
<?php

$today = "wednesday";

switch($today){
```

```
case "sunday":
echo "pray for us sinners.";
break;
case "wednesday":
echo "ladies night, take her out for dinner";
break;
case "saturday":
echo "take care as you go out tonight.";
break;
default:
echo "have a nice day at work";
break;
}
?>
```

Output:

ladies night, take her out for dinner

PHP switch statement with character

Program to check Vowel and consonant

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

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

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

```
?>
```

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
  $ch = "B.Tech";
  switch ($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:
      echo "Wrong Choice";
      break;
  }
?>
```

Output:

B.Tech is 4 years course

PHP nested switch statement

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

```
<?php
   $car = "Hyundai";
      $model = "Tucson";
      switch( $car )
      {
         case "Honda":
            switch( $model )
            {
               case "Amaze":
                     echo "Honda Amaze price is 5.93 - 9.79 Lakh.";
                  break;
               case "City":
                     echo "Honda City price is 9.91 - 14.31 Lakh.";
                  break;
            }
            break;
         case "Renault":
            switch( $model )
            {
               case "Duster":
                  echo "Renault Duster price is 9.15 - 14.83 L.";
                  break;
               case "Kwid":
                     echo "Renault Kwid price is 3.15 - 5.44 L.";
                  break;
```

```
            }
            break;
        case "Hyundai":
            switch( $model )
            {
                case "Creta":
                    echo "Hyundai Creta price is 11.42 - 18.73 L.";
                    break;
    case "Tucson":
                    echo "Hyundai Tucson price is 22.39 - 32.07 L.";
                    break;
                case "Xcent":
                    echo "Hyundai Xcent price is 6.5 - 10.05 L.";
                    break;
            }
            break;
    }
?>
```

Output:

Hyundai Tucson price is 22.39 - 32.07 L.

Activities:

1. write a program to find even odd number in php.
2. write a php script to calculate the area of circle square and rectangle.
3. write a php script to calculate the area of square.
4. write a php program to find the sum of digits
5. write a php script to find the largest of three numbers

PHP Loops

Different Types of Loops in PHP

Loops are used to execute the same block of code again and again, as long as a certain condition is met. The basic idea behind a loop is to automate the repetitive tasks within a program to save the time and effort. PHP supports four different types of loops.

- **while —** loops through a block of code as long as the condition specified evaluates to true.
- **do…while —** the block of code executed once and then condition is evaluated. If the condition is true the statement is repeated as long as the specified condition is true.
- **for —** loops through a block of code until the counter reaches a specified number.
- **foreach —** loops through a block of code for each element in an array.
  You will also learn how to loop through the values of array using foreach() loop at the end of this chapter. The foreach() loop work specifically with arrays.

i. while Loop

The while statement will loops through a block of code as long as the condition specified in the while statement evaluate to true.

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

The example below define a loop that starts with $i=1. The loop will continue to run as long as $i is less than or equal to 3. The $i will increase by 1 each time the loop runs:

Example:

```
<?php
$i = 1;
while($i <= 3){
   $i++;
   echo "The number is " . $i . "<br>";
}
?>
```

ii. do…while Loop

The do-while loop is a variant of while loop, which evaluates the condition at the end of each loop iteration. With a do-while loop the block of code executed once, and then the condition is evaluated, if the condition is true, the statement is repeated as long as the specified condition evaluated to is true.

```
do{
```

```
   // Code to be executed
}
while(condition);
```

The following example define a loop that starts with $i=1. It will then increase $i with 1, and print the output. Then the condition is evaluated, and the loop will continue to run as long as $i is less than, or equal to 3.

Example:

```
<?php
$i = 1;
do{
   $i++;
   echo "The number is " . $i . "<br>";
}
while($i <= 3);
?>
```

Difference Between while and do…while Loop

The while loop differs from the do-while loop in one important way — with a while loop, the condition to be evaluated is tested at the beginning of each loop iteration, so if the conditional expression evaluates to false, the loop will never be executed.

With a do-while loop, on the other hand, the loop will always be executed once, even if the conditional expression is false, because the condition is evaluated at the end of the loop iteration rather than the beginning.

iii. for Loop

The for loop repeats a block of code as long as a certain condition is met. It is typically used to execute a block of code for certain number of times.

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

The parameters of for loop have following meanings:

initialization — it is used to initialize the counter variables, and evaluated once unconditionally before the first execution of the body of the loop.

condition — in the beginning of each iteration, condition is evaluated. If it evaluates to true, the loop continues and the nested statements are executed. If it evaluates to false, the execution of the loop ends.

increment — it updates the loop counter with a new value. It is evaluate at the end of each iteration.

The example below defines a loop that starts with $i=1. The loop will continued until $i is less than, or equal to 3. The variable $i will increase by 1 each time the loop runs:

Example:

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

iv. foreach Loop

The foreach loop is used to iterate over arrays.

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

The following example demonstrates a loop that will print the values of the given array:

Example:

```
<?php
$colors = array("Red", "Green", "Blue");

// Loop through colors array
foreach($colors as $value){
   echo $value . "<br>";
}
?>
```

Activities:

1. write a php script to print factorial of number.
2. write a php script to print prime number upto n.
3. write a php script to print fibonacci series.
4. write a php script to find the sum of digits of a given number.
5. write a php program to check whether the given number is prime or not.

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.

Why use Functions?

Better code organization – PHP functions allow us to group blocks of related code that perform a specific task together.

Reusability – once defined, a function can be called by a number of scripts in our PHP files. This saves us time of reinventing the wheel when we want to perform some routine tasks such as connecting to the database

Easy maintenance- updates to the system only need to be made in one place.

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 provides us with two major types of functions:

i. **Built-in functions :** PHP provides us with huge collection of built-in library functions. These functions are already coded and stored in form of functions. To use those we just need to call them as per our requirement like, var_dump, fopen(), print_r(), gettype() and so on.

ii. **User Defined Functions :** Apart from the built-in functions, PHP allows us to create our own customised functions called the user-defined functions.
Using this we can create our own packages of code and use it wherever necessary by simply calling it.

Creating a Function

While creating a user defined function we need to keep few things in mind:

- Any name ending with an open and closed parenthesis is a function.
- A function name always begins with the keyword function.
- To call a function we just need to write its name followed by the parenthesis
- A function name cannot start with a number. It can start with an alphabet or underscore.
- A function name is not case-sensitive.

Syntax:

```
function function_name(){
   executable code;
}
```

Example:

```
<?php
function funcWel()
{
   echo "Welcome to PHP Function";
}
// Calling the function
funcWel();
?>
```

Output:

Welcome to 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.

```
<?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.

```
<?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

Parameter passing to Functions

PHP allows us two ways in which an argument can be passed into a function:

i. **Pass by Value:** On passing arguments using pass by value, the value of the argument gets changed within a function, but the original value outside the function remains unchanged. That means a duplicate of the original value is passed as an argument.

ii. **Pass by Reference:** On passing arguments as pass by reference, the original value is passed. Therefore, the original value gets altered. In pass by reference we actually pass the address of the value, where it is stored using ampersand sign(&).

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.

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

Output:

Hello Call By Reference

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.

```
<?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.

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

Output:

Cube of 3 is: 27

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

Output:

10

Example 2

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

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

Built in functions are predefined functions in PHP that exist in the installation package.

These PHP inbuilt functions are what make PHP a very efficient and productive scripting language.

The built in functions of PHP can be classified into many categories. Below is the list of the categories.

- String Functions:- PHP provides various string functions to access and manipulate strings.
- Numeric Functions:- Numeric functions in PHP are the functions that return numeric results. Numeric php function can be used to format numbers, return constants, perform mathematical computations etc.
- Date Function:- The date function is used to format Unix date and time to human readable format.
- PHP Array Functions:- PHP provides various array functions to access and manipulate the elements of array. The important PHP array functions are given below.

PHP Arrays

PHP arrays are complex data structures. They may represent an ordered map with integer and string keys to any PHP values

An array is a special variable that we use to store or hold more than one value in a single variable without having to create more variables to store those values.

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.

Syntax:

An array can be created using the array() language construct. It takes any number of comma-separated key => value pairs as arguments.

```
array(
   key  => value,
   key2 => value2,
   key3 => value3,
   ...
)
```

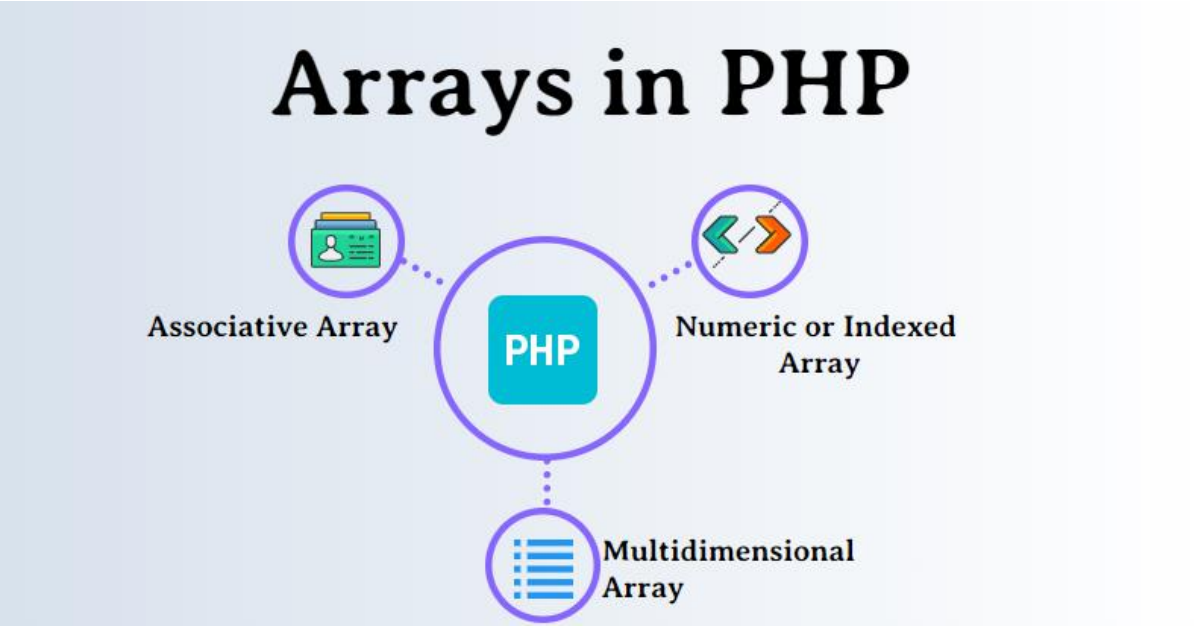
The comma after the last array element is optional and can be omitted. This is usually done for single-line arrays, i.e. array(1, 2) is preferred over array(1, 2,). For multi-line arrays on the other hand the trailing comma is commonly used, as it allows easier addition of new elements at the end.

Note:A short array syntax exists which replaces array() with [].

PHP Array Types

There are 3 types of array in PHP.

i. Indexed Array
ii. Associative Array
iii. Multidimensional Array



1. **Index Array/ Numeric Array:**
   These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.
   Example
   Following is the example showing how to create and access numeric arrays.
   Here we have used array() function to create array. This function is explained in function reference.

```
<html>
  <body>

    <?php
      /* First method to create array. */
      $numbers = array( 1, 2, 3, 4, 5);

      foreach( $numbers as $value ) {
        echo "Value is $value <br />";
      }

      /* Second method to create array. */
      $numbers[0] = "one";
      $numbers[1] = "two";
      $numbers[2] = "three";
      $numbers[3] = "four";
      $numbers[4] = "five";

      foreach( $numbers as $value ) {
        echo "Value is $value <br />";
      }
    ?>

  </body>
</html>
```

This will produce the following result −

Value is 1

Value is 2

Value is 3

Value is 4

Value is 5

Value is one

Value is two

Value is three

Value is four

Value is five

2. Associative Arrays

- The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.
- To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.
  NOTE − Don't keep associative array inside double quote while printing otherwise it would not return any value.

Example

```
<html>
  <body>
    <?php
      /* First method to associate create array. */
      $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" => 500);

      echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
      echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
      echo "Salary of zara is ".  $salaries['zara']. "<br />";

      /* Second method to create array. */
      $salaries['mohammad'] = "high";
      $salaries['qadir'] = "medium";
      $salaries['zara'] = "low";

      echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
      echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
      echo "Salary of zara is ".  $salaries['zara']. "<br />";
    ?>
  </body>
</html>
```

This will produce the following result −

Salary of mohammad is 2000

Salary of qadir is 1000

Salary of zara is 500

Salary of mohammad is high

Salary of qadir is medium

Salary of zara is low

3. Multidimensional Arrays

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Example

In this example we create a two dimensional array to store marks of three students in three subjects

This example is an associative array, you can create numeric array in the same fashion.

```
<html>
  <body>

    <?php
      $marks = array(
        "mohammad" => array (
          "physics" => 35,
          "maths" => 30,
          "chemistry" => 39
        ),

        "qadir" => array (
          "physics" => 30,
          "maths" => 32,
          "chemistry" => 29
        ),
```

```
      "zara" => array (
         "physics" => 31,
         "maths" => 22,
         "chemistry" => 39
      )
   );

   /* Accessing multi-dimensional array values */
   echo "Marks for mohammad in physics : " ;
   echo $marks['mohammad']['physics'] . "<br />";

   echo "Marks for qadir in maths : ";
   echo $marks['qadir']['maths'] . "<br />";

   echo "Marks for zara in chemistry : " ;
   echo $marks['zara']['chemistry'] . "<br />";
?>
</body>
</html>
```

This will produce the following result −

```
Marks for mohammad in physics : 35
Marks for qadir in maths : 32
Marks for zara in chemistry : 39
```

PHP Array Functions

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

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
$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

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
$salary=array("Sonoo"=>"550000","Vimal"=>"250000","Ratan"=>"200000");
print_r(array_change_key_case($salary,CASE_UPPER));
?>
```

Output:

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

Example

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

Output:

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

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

Output:

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

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

Output:

4

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

Output:

```
autumn
spring
summer
winter
```

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

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

Output:

2

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
$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

unset() Function:

The unset() function is used to remove element from the array. The unset function is used to destroy any other variable and same way use to delete any element of an array. This unset command takes the array key as input and removed that element from the array. After removal the associated key and value does not change.

Syntax:

unset($variable)

Parameter: This function accepts single parameter variable. It is required parameter and used to unset the element.

Example:

```
<?php
// PHP program to delete an array
// element based on index

// Declare arr variable
// and initialize it
$arr = array('G', 'E', 'E', 'K', 'S');

// Display the array element
print_r($arr);

// Use unset() function delete
// elements
unset($arr[2]);

// Display the array element
print_r($arr);
?>
```

Output:

```
Array
(
```

```
    [0] => G
    [1] => E
    [2] => E
    [3] => K
    [4] => S
)
Array
(
    [0] => G
    [1] => E
    [3] => K
    [4] => S
)
```

How to check whether an array is empty using PHP?

An empty array can sometimes cause software crash or unexpected outputs. To avoid this, it is better to check whether an array is empty or not beforehand. There are various methods and functions available in PHP to check whether the defined or given array is an empty or not. Some of them are given below:

i. Using empty() Function:

This function determines whether a given variable is empty. This function does not return a warning if a variable does not exist.

Syntax:

bool empty($var)

```
<?php
// Declare an array and initialize it
$non_empty_array = array('URL' => 'https://www.geeksforgeeks.org/');
// Declare an empty array
$empty_array = array();
// Condition to check array is empty or not
if(!empty($non_empty_array))
        echo "Given Array is not empty <br>";
```

```
if(empty($empty_array))
        echo "Given Array is empty";
?>
```

Output:

Given Array is not empty

Given Array is empty

ii. Using count Function:

This function counts all the elements in an array. If number of elements in array is zero, then it will display empty array.

Syntax:

int count($array_or_countable)

Example:

```
<?php
// Declare an empty array
$empty_array = array();
// Function to count array
// element and use condition
if(count($empty_array) == 0)
   echo "Array is empty";
else
   echo "Array is non- empty";
?>
```

Output:

Array is empty

iii. Using sizeof() function:

This method check the size of array. If the size of array is zero then array is empty otherwise array is not empty.

Example:

```
<?php
// Declare an empty array
$empty_array = array();
// Use array index to check
// array is empty or not
if( sizeof($empty_array) == 0 )
   echo "Empty Array";
else
   echo "Non-Empty Array";
?>
```

Output:

Empty Array

PHP Math Functions

They are inbuilt functionalities of PHP as a programming language. The basic role of these functions is to provide a mechanism in which a developer can do some sort of math calculations or similar stuff. These provide a quick hands-on for development without writing a long piece of code. Now with that let us know the range of these PHP math functions

Range of PHP math functions

The range of these php math functions is within integer and float types. Range of integer data type in PHP for a 32-bit computer is -2,147,483,647 to 2,147,483,647. Any number smaller than -2,147,483,647 or any number greater than 2,147,483,647 or any number smaller than -2,147,483,647 is considered as float.

Now we will try to understand different PHP math functions, along with its use an Example:

abs() function

It was introduced in PHP 4+ version. It returns the absolute value of the number. The return type of the function is float or integer number depending upon what type of argument passed in the function.

Example:

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

```
<body>
<?php
echo(abs(3.5) . "<br>");
echo(abs(-3.5) . "<br>");
echo(abs(5) . "<br>");
echo(abs(-5));
?>
</body>
</html>
```

Output:

PHP Math Functions-1

acos() function

It was introduced in PHP 4+ version. It expects argument in the range of -1 to +1. If number out of specified range is passed in an argument then it returns NaN otherwise it returns arc cosine value of the number. The return type of the function is arc cosine of a number

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
echo(acos(0.35) . "<br>");
echo(acos(-0.35) . "<br>");
echo(acos(5) . "<br>");
echo(acos(0.7253));
?>
</body>
</html>
```

Output:

PHP Math Functions-2

asin() function

It was introduced in PHP 4+ version. It expects argument in the range of -1 to +1. If number out of specified range is passed in an argument then it returns NaN otherwise it returns arc sine value of the number. The return type of the function is arc sine of a number

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
echo(asin(0.35) . "<br>");
echo(asin(-0.35) . "<br>");
echo(asin(5) . "<br>");
echo(asin(0.7253));
?>
</body>
</html>
```

Output:

PHP Math Functions-3

ceil() function

It was introduced in PHP 4+ version. It rounds a number up to the nearest integer. For example, the ceil of 3.2 will give 4. It returns the whole number in the form of nearest integer which is greater than the passed argument

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
echo(ceil(3.35) . "<br>");
echo(ceil(-4.35) . "<br>");
echo(ceil(5) . "<br>");
```

```
echo(ceil(14.8114700666069059));
?>
</body>
</html>
```

Output:

PHP Math Functions-4

floor() function

It was introduced in PHP 4+ version. It rounds a number down to the nearest integer. For example, the floor of 3.2 will give 3. It returns the whole number in the form of nearest integer which is smaller than the passed argument

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
echo(floor(3.35) . "<br>");
echo(floor(-2.35) . "<br>");
echo(floor(5) . "<br>");
echo(floor(14.811470062));
?>
</body>
</html>
```

Output:

PHP Math Functions-5

pi() function

It was introduced in PHP 4+ version. It returns value of a PI and its return type is a float

Example:

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

```
<?php
echo(pi() . "<br>");
?>
</body>
</html>
```

Output:

PHP Math Functions-6

pow() function

It was introduced in PHP 4+ version. It accepts two arguments say x and y. It calculates x raised to the power of y. its return type is integer or float which depends upon the nature of the argument

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
echo(pow(2,3) . "<br>");
echo(pow(2,4) . "<br>");
echo(pow(5,6) . "<br>");
echo(pow(3,5));
?>
</body>
</html>
```

Output:

pow() function-7

log() function

It was introduced in PHP 4+ version. It accepts two arguments say x and y where x is a number and y is the logarithm of a number to base. If y is not passed then the default value ‘e’ is assumed. Its return type is float

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
echo(log(2.718) . "<br>");
echo(log(2) . "<br>");
echo(log(1) . "<br>");
echo(log(0));
?>
</body>
</html>
```

Output:

log() function -8

log10() function

It was introduced in PHP 4+ version. It accepts one argument says x where x is a number whose base 10 logarithm needs to be calculated. Its return type is float

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
echo(log10(656) . "<br>");
echo(log10(455) . "<br>");
echo(log10(145) . "<br>");
?>
</body>
</html>
```

Output:

log10() function -9

round() function

It was introduced in PHP 4+ version. It rounds a number. It expects three parameters where the first parameter is number, the second parameter is for precision and the third argument is for mode. The only first argument is mandatory

Example:

```
<!DOCTYPE html>
<html>
<body>
<?php
echo(round(3.35) . "<br>");
echo(round(-2.35) . "<br>");
echo(round(5) . "<br>");
?>
</body>
</html>
```

Output

round() function-10

PHP Date and Time

Date and time are some of the most frequently used operations in PHP while executing SQL queries or designing a website etc. PHP serves us with predefined functions for these tasks. Some of the predefined functions in PHP for date and time are discussed below.

PHP date() Function: The PHP date() function converts timestamp to a more readable date and time format.

Syntax:

date(format, timestamp)

Explanation:

- The format parameter in the date() function specifies the format of returned date and time.
- The timestamp is an optional parameter, if it is not included then the current date and time will be used.

Example: The below program explains the usage of the date() function in PHP.

```
<?php
echo "Today's date is :";
$today = date("d/m/Y");
echo $today;
?>
```

Output:

Today's date is :05/12/2017

Formatting options available in date() function:

The format parameter of the date() function is a string that can contain multiple characters allowing to generate the dates in various formats. Date-related formatting characters that are commonly used in the format string:

d: Represents day of the month; two digits with leading zeros (01 or 31).

D: Represents day of the week in the text as an abbreviation (Mon to Sun).

m: Represents month in numbers with leading zeros (01 or 12).

M: Represents month in text, abbreviated (Jan to Dec).

y: Represents year in two digits (08 or 14).

Y: Represents year in four digits (2008 or 2014).

The parts of the date can be separated by inserting other characters, like hyphens (-), dots (.), slashes (/), or spaces to add additional visual formatting.

Example: The below example explains the usage of the date() function in PHP.

```
<?php
echo "Today's date in various formats:" . "\n";
echo date("d/m/Y") . "\n";
echo date("d-m-Y") . "\n";
echo date("d.m.Y") . "\n";
echo date("d.M.Y/D");
?>
```

Output:

Today's date in various formats:

05/12/2017

05-12-2017

05.12.2017

05.Dec.2017/Tue

The following characters can be used along with the date() function to format the time string:

h: Represents hour in 12-hour format with leading zeros (01 to 12).

H: Represents hour in 24-hour format with leading zeros (00 to 23).

i: Represents minutes with leading zeros (00 to 59).

s: Represents seconds with leading zeros (00 to 59).

a: Represents lowercase antemeridian and post meridian (am or pm).

A: Represents uppercase antemeridian and post meridian (AM or PM).

Example: The below example explains the usage of the date() function in PHP.

```
<?php
echo date("h:i:s") . "\n";
echo date("M,d,Y h:i:s A") . "\n";
echo date("h:i a");
?>
```

Output:

03:04:17

Dec,05,2017 03:04:17 PM

03:04 pm

PHP time() Function:

The time() function is used to get the current time as a Unix timestamp (the number of seconds since the beginning of the Unix epoch: January 1, 1970, 00:00:00 GMT).

The following characters can be used to format the time string:

h: Represents hour in 12-hour format with leading zeros (01 to 12).

H: Represents hour in 24-hour format with leading zeros (00 to 23).

i: Represents minutes with leading zeros (00 to 59).

s: Represents seconds with leading zeros (00 to 59).

a: Represents lowercase antemeridian and post meridian (am or pm).

A: Represents uppercase antemeridian and post meridian (AM or PM).

Example: The below example explains the usage of the time() function in PHP.

```
<?php
$timestamp = time();
```

```
echo($timestamp);
echo "\n";
echo(date("F d, Y h:i:s A", $timestamp));
?>
```

Output:

1512486297

December 05, 2017 03:04:57 PM

PHP mktime() Function:

The mktime() function is used to create the timestamp for a specific date and time. If no date and time are provided, the timestamp for the current date and time is returned.

Syntax:

mktime(hour, minute, second, month, day, year)

Example:

The below example explains the usage of the mktime() function in PHP.

```
<?php
echo mktime(23, 21, 50, 11, 25, 2017);
?>
```

Output:

1511652110

The above code creates a time stamp for 25th Nov 2017,23 hrs 21mins 50secs.

PHP checkdate() Function

The checkdate() function is a built-in function in PHP which checks the validity of the date passed in the arguments. It accepts the date in the format mm/dd/yyyy. The function returns a boolean value. It returns true if the date is a valid one, else it returns false.

Syntax:

checkdate ($month, $day, $year)

Parameters: The function accepts three mandatory parameters as shown above and described below:

$month – This parameter specifies the month. The month has to be in between 1 to 12 for a valid date.

$day – This parameter specifies the day. The day can be in range 1-31 depending on the month entered for it to be a valid day. In case of a leap year, the day is in range 1-29 and for a non-leap year the day is in range 1-28.

$year – This parameter specifies the year. The year has to be in range 1-32767 inclusive depending on the $month and $day for it to be a valid date.

Return Value: The function returns a boolean value. It returns true if the passed date is a valid date. It returns false if the passed date is not a valid one.

Examples:

The program below check if the date is a valid one or not.

```
<?php
// PHP program to demonstrate the checkdate() function
$month = 12;
$day = 31;
$year = 2017;
// returns a boolean value after validation of date
var_dump(checkdate($month, $day, $year));
?>
```

Output:

```
bool(true)
```

PHP date_diff() Function

The date_diff() is an inbuilt function in PHP which is used to calculate the difference between two dates. This function returns a DateInterval object on the success and returns FALSE on failure.

Syntax:

```
date_diff($datetime1, $datetime2);
```

Parameters: The date_diff() function accepts two parameters as mentioned above and described below:

$datetime1: It is a mandatory parameter which specifies the first DateTime object.

$datetime2: It is a mandatory parameter which specifies the second DateTime object.

Return Value: It returns the difference between two DateTime objects otherwise, FALSE on failure.

Example:

```
<?php
// PHP program to illustrate
// date_diff() function


// creates DateTime objects
$datetime1 = date_create('2017-06-28');
```

```
$datetime2 = date_create('2018-06-28');


// calculates the difference between DateTime objects
$interval = date_diff($datetime1, $datetime2);


// printing result in days format
echo $interval->format('%R%a days');
?>
```

Output:

```
+365 days
```

PHP date_modify() Function

The date_modify() function is an inbuilt function in PHP. Through the help of this function, we can modify or can alter the timestamp of DateTime object. The DateTime object and string modify are parameters in the calling function.

Syntax:

```
date_modify(DateTime $object, string $modify);
```

Parameters:

The function accepts two parameters as described below:

$object : This is a mandatory parameter. It specifies a DateTime object returned by date_create() .This object is modified by the above mentioned function.

$modify : This is also a mandatory parameter. This specifies a date/time string. It is incremented or decremented to modify the DateTime object.

Return Values :

This function returns a DateTime object on success. And returns FALSE on failure.

Below programs illustrate the date_modify() function :

```
<?php
// PHP program to illustrate date_modify()
// function


// creating DateTime object
$date=date_create("2018-04-08");
```

```
// calling of date modify function
// with two required parameters
date_modify($date, "+15 days");

// printing the modified date in "y-m-d" format
echo date_format($date, "Y-m-d");
?>
```

Output:

2018-04-13

This program will be increase months by one

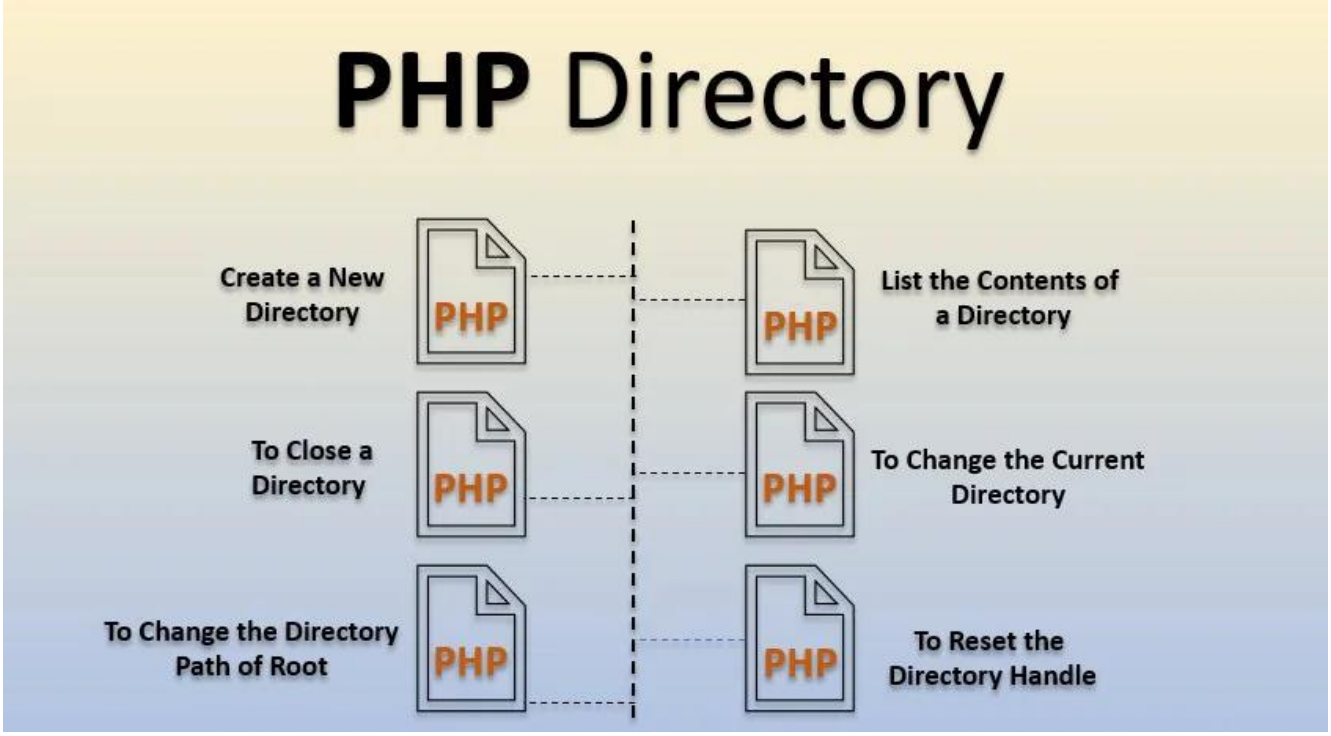
```
<?php
// PHP program to illustrate date_modify()
// function

// creating a DateTime object
$date = date_create('2000-10-14');

// calling date_modify function with
// two required parameters
date_modify($date, '+1 month');

// printing the modified date
echo date_format($date, 'Y-m-d');
?>
```

Output:

2000-11-14

Understanding File & Directory

PHP Directory



Introduction to PHP Directory

PHP directory functions as their name suggests are a set of functions used in retrieving details, modifying them and fetching information on various file system directories and their specific contents. A lot of operations can be performed on the directories like creating, deleting, changing the present working directory, listing files present in the directory and so on. There is no separate installation required for these functions as they come as part of the PHP core. But to enable chroot() function we need to configure –enable-chroot-func option.

Functions of PHP Directory

Let us go through a few of the basic PHP directory functions as below:

1. **Create a New Directory**
   We use the mkdir() function to create a new directory in the PHP programming script.

Syntax:

```
mkdir($dir_path,$mode,$recursive_flag,$context);
```

where,

$dir_path is either the relative or the absolute path where the new directory specified will be created.

$mode is the parameter that will take in octal values which determines the level in which the newly created directory will be accessible.

$recursive is a flag type field that has 2 values either true or false which can either allow us to create nested directories or not.

$context is similar to what we have with PHP unlink() like having a stream to specify certain protocols etc. This will also return only a boolean value which will be true if the execution is completed successfully and false otherwise.

Example:

```
<?php
mkdir("/articles/");
echo("Directory created");
?>
```

Output:

Directory Output

This is a basic example to show the creation of a directory in the path we require. Make sure the path has sufficient permissions else “permission denied” error will be thrown.

Example2:

```
<?php
$file="data.txt";


if(!file_exists("textfile")){
    mkdir("textfile");
  }
  else{
    echo "Folder already exist";
  }


?>
```

2. List the Contents of a Directory

We use opendir() and readdir() for opening the directory link and to read it respectively. Step 1 will be to open the directory and Step 2 will be to read it.
Step 1: To open the directory link, opendir() is the function we use to do this step. It requires two input arguments as specified below.

Syntax:

opendir($dir_path,$context);

$dir_path is the path of the directory which needs to be opened.

$context is an optional parameter where we can specify if a context stream is present.

This returns resource data value as its output. This resource ID which it provides is used in our further processing steps else we get an error as resource ID is invalid.

Step 2: To read the contents of the directory, readdir() is the function which is used for this purpose and it needs to be called recursively till the end of the directory is reached by the directory handle.

Example:

```
<?php
$dir=".";
if(is_dir($dir)){
   if($d=opendir($dir)){
      while($file=readdir($d)){
         echo "filename:" . $file. "<br>";
      }
      closedir($d);
   }
}
?>
```

Output:

filename:.
filename:..
filename:constant.php
filename:control.php
filename:Demo.php

filename:FirstProgram.php
filename:function.php
filename:images
filename:loop.php
filename:operator.php
filename:operator1.php
filename:variable.php
filename:“ratan”

In this example first, we are declaring the directory path which needs to be read. We are checking in the if statement if the directory is present and then proceeding to open the contents of the directory and read. The output displays the filenames present inside the directory.

3. To Close a Directory

We use closedir() function in order to close a directory after reading its contents.

Syntax:

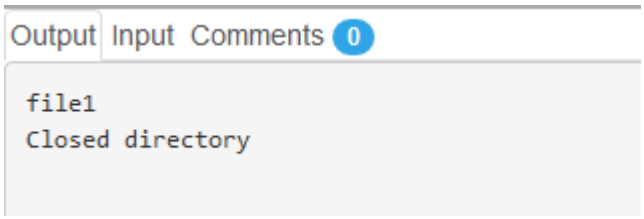
```
$dir_handle = opendir($dir_path);
...
...
closedir($dir_handle);
```

Example:

```
<?php
$dir = "/file1";
if (is_dir($dir)) {
if ($dh = opendir($dir)) {
$direc = readdir($dh);
echo("File present inside directory are:" .direc);
closedir($dh);
echo("Closed directory");
}
}
```

```
?>
```

Output:



In this example, we are first declaring the path of our directory. Then using the if conditional statement we are checking if the path is valid and if yes, then we are opening the directory, reading its variables and then closing it. Thus, any operation can be done between the opening and closing of the directory.

4. **To Change the Current Directory**
   We use the function chdir() to change the current working directory in which it is pointing to.

Syntax:

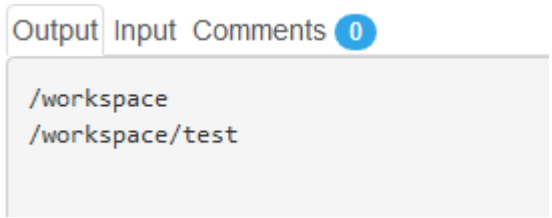
```
chdir(directory)
```

It requires only one parameter that is the directory to which the current working directory should be pointed to. It returns true on success and false if failed to change the directory.

Example:

```
<?php
// Get current directory
echo getcwd()."\n";
// Change directory
chdir("/workspace/test");
// Get current directory
echo getcwd();
?>
```

Output:



In this example, we are first printing the present working directory. Then we are changing the same using chdir function to “test” directory and printing the same on the output. Hence make sure the entire path we are giving here exists.

```
<?php
mkdir("images");
chdir("images");
echo getcwd();
?>
```

5. **Scandir()**

It returns an array of files and directories from the passed directory.
The scandir() function in PHP is an inbuilt function that is used to return an array of files and directories of the specified directory. The scandir() function lists the files and directories which are present inside a specified path. The directory, stream behavior, and sorting_order of the files and directories are passed as a parameter to the scandir() function and it returns an array of filenames on success, or false on failure.

Syntax:

scandir(directory, sorting_order, context);

Parameters: The scandir() function in PHP accepts 3 parameters that are listed below:

directory: It is a mandatory parameter that specifies the path.
sorting_order: It is an optional parameter that specifies the sorting order. Alphabetically ascending order (0) is the default sort order. It can be set to SCANDIR_SORT_DESCENDING or 1 to sort in alphabetically descending order, or SCANDIR_SORT_NONE to return the result unsorted.
context: It is an optional parameter that specifies the behavior of the stream.
Return Value: It returns an array of filenames on success, or false on failure.

Example:

```
<php
$dir=getcwd();
echo "<pre>";
print_r(scandir($dir));
print_r(scandir($dir,0));
echo "</pre>";
```

```
?>
```

PHP Directory Functions

Function	Description
chdir()	Changes the current directory
chroot()	Changes the root directory
closedir()	Closes a directory handle
dir()	Returns an instance of the Directory class
getcwd()	Returns the current working directory
opendir()	Opens a directory handle
readdir()	Returns an entry from a directory handle
scandir()	Returns an array of files and directories of a specified directory

PHP File Handling

- Using PHP file handling mechanism, we can get external file resources to store as a reference. 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 provides set of in-built functions to handle files. Some of the functions are, fopen(), file_exists(), file_get_contents() and etc.
- PHP File Functions help in-store/delete/manipulate/copy the data in the file or deleting the file etc. Here is the list of some of the file functions. They are:
  i. Readfile Function
  ii. file_exists Function

iii. fopen Function
iv. fwrite Function
v. fclose Function
vi. fgets Function
vii. copy Function
viii. file_get_contents Function
ix. deleting a File
x. file size
xi. filetype
xii. realpath

Different types of File MODE

Modes	Description
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist
a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

The file may be opened in one of the following modes:

I. readfile()

The readfile() function reads a file and writes it to the output buffer.

Tip: You can use a URL as a filename with this function if the fopen wrappers have been enabled in the php.ini file.

Syntax

readfile(file, include_path, context)

Parameter Values

Parameter	Description
file	Required. Specifies the file to read
include_path	Optional. Set this parameter to TRUE if you want to search for the file in the include_path (in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

Example:

```
<?php
   echo readfile("data.txt");
?>
```

To read some content of the file.

```
<?php
$file=fopen("data.txt","r");
echo fread($file,5);

?>
```

To read entire file.

```
<?php
$file=fopen("data.txt","r");
echo fread($file,filesize("data.txt"));

?>
```

II. file_exists Function

In order to write something in the file or manipulate the data in the delete however you want then at first, you have to check whether the file exists in the directory or not in order to process it. This PHP function also helps you in creating a new if the file you search is not present in the server and you want to create a new file at the server.

Syntax:

```
<?php
file_exists($file_name);
```

```
?>
```

Explanation:

"file_exists()" function is a PHP Function which returns the result as TRUE only if the file exists in the server or the result will FALSE if the file doesn't exist/found in the server/server directory.
$file_name variable is the file path and the name of the file at the end of the path which is to check.

Example:

```
<?php
$file="data.txt";
   if(file_exists($file)){
      echo readfile("data.txt");
   }
   else{
      echo "File does not found";
   }
?>
```

Output:

III. PHP fopen Function

PHP fopen function will help you to open the file/files which are in the server.

Syntax:

```
<?php
fopen($file_name, $mode, $use_include_path,$context);
?>
```

Explanation:

- "fopen" is the PHP file function which is used to open the file which is in the server/server directory.
- "$file_name" is the actual file name which is to open
- "mode" is like what you want to do with the file like reading, writing, appending, etc.
    - ✓ Mode "r" will read the file from the beginning and returns false if the file don't even exist. It helps to read-only rather than read and write mode. For read and write mode, one must use "r+" mode.
    - ✓ Mode "w" will helps to write some data to the file. It will truncate the file to the zero-length. If the file doesn't even exist then the file will be created to write only rather than read and write. In order to read and write "w+" mode will be used.

- ✓ Mode “a” will appends the file at the end. If the file doesn’t even exist then the file will be created with write only mode. For read and write mode of appending then “a+” mode will be used.

- “$use_include_path” is the optional term and by default the result is false, if it is set to the TRUE result then the functions help the include path which is present too. Likewise “$context” is also optional which can be used in order to specify the context support.

Example:

```
<?php
// using r+
/*
$file=fopen("data.txt","r+");
fwrite($file,"Here we have new line.");
*/
//Using w+
/*
$file=fopen("data.txt","w+");
fwrite($file,"Here we have new line.");
*/
// Using a+
$file=fopen("data.txt","w+");
fwrite($file,"\n Here we have new line.");
?>
```

IV. PHP write Function

PHP write function will help you to write files.

Syntax:

```
<?php
fwrite($handle,$string,$length);
?>
```

Explanation:

“fwrite” PHP function will help to write some data to the files.

“$handle” term is the file pointer’s resource.

“$string” term is the data/information which is to be written inside the file.

“$length” term is optional which helps to specify the maximum file length.

V. PHP Fclose Function

Fclose Function will help to close the file which is opened already in the server.

Syntax:

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

Explanation:

"fclose" will helps you to close the function which is opened already in the server/server directory.

"$handle" is the pointer's resource of the file.

VI. PHP fgets Function

PHP Fgets Functions will help to read the file/files are red line by line using the syntax:

fgets($handle);

- "$fgets" is to read the lines of the file.
- "$handle" is the resource of the file pointer.

Example:

```
<?php
$file=fopen("data.txt","r");
echo fgets($file);


?>
```

VII. PHP Copy Function

PHP copy function will be used in order to copy the files.

Syntax:

copy($file, $file_copied);

Explanation:

- "$file" is the path of the file which is to be copied.
- "$file_copied" term is the name of the copied file.

```
<?php
$file="data.txt";
   if(file_exists($file)){
      echo readfile("data.txt");
```

```
    copy($file,"newfile.txt");
  }
  else{
    echo "File does not found";
  }
?>
```

VIII. PHP file_get_contents Function

This function helps in reading the entire contents of the file. Difference between the fgets and file_get_contents will return the whole data as a string but the fgets will be red the whole file line by line.

Syntax:

file_get_contents();

Example:

```
<?php
echo "<pre>"; // Enables the display of the line feeds
echo file_get_contents("file_name.txt");
echo "</pre>"; // Now it Terminates the pre tag
?>
```

IX. Deleting a File (Unlink Function)

Unlink Function will help to delete a file.

The unlink() function is an inbuilt function in PHP which is used to delete files. It is similar to UNIX unlink() function. The $filename is sent as a parameter that needs to be deleted and the function returns True on success and false on failure.

Syntax:

unlink($filename, $context)

Parameters: This function accepts two parameters as mentioned above and described below:

- $filename: It is a mandatory parameter which specifies the filename of the file which has to be deleted.
- $context: It is an optional parameter which specifies the context of the file handle which can be used to modify the nature of the stream.
- Return Value: It returns True on success and False on failure.

Example:

```
<?php
$file="newfile.txt";
```

```
   if(file_exists($file)){
      unlink("newfile.txt");
   }
   else{
      echo "File does not found";
   }
?>
```

X. Rename()

The rename() function in PHP is an inbuilt function which is used to rename a file or directory. It makes an attempt to change an old name of a file or directory with a new name specified by the user and it may move between directories if necessary.
If the new name specified by the user already exists, the rename() function overwrites it. The old name of the file and the new name specified by the user are sent as parameters to the rename() function and it returns True on success and a False on failure.

Syntax:

```
rename(oldname, newname, context)
```

Parameters Used:
The rename() function in PHP accepts three parameter.

- oldname : It is a mandatory parameter which specifies the old name of the file or directory.
- newname : It is a mandatory parameter which specifies the new name of the file or directory.
- context : It is an optional parameter which specifies the behavior of the stream .
- Return Value: It returns True on success and a False on failure.

Example:

```
<?php
$file="data.txt";
   if(file_exists($file)){
      rename("oldfile.txt","newfile.txt");
   }
   else{
      echo "File does not found";
   }
?>
```

XI. Filesize

The filesize() function in PHP is an inbuilt function which is used to return the size of a specified file. The filesize() function accepts the filename as a parameter and returns the size of a file in bytes on success and False on failure.

The result of the filesize() function is cached and a function called clearstatcache() is used to clear the cache.

Syntax:

filesize($filename)

Parameters: The filesize() function in PHP accepts only one parameter $filename. It specifies the filename of the file whose size you want to check.
Return Value: It returns the size of a file in bytes on success and False on failure.

Example:

```
<?php
$file="data.txt";
echo filesize($file);

?>
```

XII. Filetype()

The filetype() function in PHP is an inbuilt function which is used to return the file type of a specified file or a directory.
The filetype() function accepts the filename as a parameter and returns one of the seven file types on success and False on failure.
Syntax:

filetype($filename)

Parameters: The filetype() function in PHP accepts only one parameter $filename. It specifies the filename of the file whose type you want to know.
Return Value: It returns the type of a file on success and False on failure.
Example:

```
<?php
$file="data.txt";
echo filetype($file);

?>
```

XIII. Realpath

The realpath() function in PHP is an inbuilt function which is used to return the canonicalized absolute pathname.
The realpath() function removes all symbolic links such as '/./' '/../' and extra '/' and returns the absolute pathname.
The path is sent as a parameter to the realpath() function and it returns the absolute pathname on success and a False on failure.

Syntax:

realpath(path)

Parameters Used:

The realpath() function in PHP accepts only one parameter.

- path : It is a mandatory parameter which specifies the symbolic path whose absolute path the user wants to know.
- Return Value:It returns the absolute pathname on success and a False on failure.

Example:

```
<?php
$file="data.txt";
echo realpath($file);
?>
```

Ex:2

```
<?php
$file="data.txt";
$path=realpath($file);
echo "<pre>";
print_r(pathinfo($path));
echo "</pre>";

?>
```

Building a text editor file: uploading & downloading

Building a text editor File

The first thing that must be done is define the file(s). An array is used so that both files can be identified and called through $files and later through the $file. A value is being defined which will help pull the text from the respective submit buttons in the form fields.

PHP File Upload

- A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script.
- Information in the phpinfo.php page describes the temporary directory that is used for file uploads as upload_tmp_dir and the maximum permitted size of files that can be uploaded is stated as upload_max_filesize. These parameters are set into PHP configuration file php.ini

The process of uploading a file follows these steps −

- The user opens the page containing a HTML form featuring a text files, a browse button and a submit button.
- The user clicks the browse button and selects a file to upload from the local PC.
- The full path to the selected file appears in the text filed then the user clicks the submit button.
- The selected file is sent to the temporary directory on the server.
- The PHP script that was specified as the form handler in the form's action attribute checks that the file has arrived and then copies the file into an intended directory.
- The PHP script confirms the success to the user.

As usual when writing files it is necessary for both temporary and final locations to have permissions set that enable file writing. If either is set to be read-only then process will fail.

An uploaded file could be a text file or image file or any document.

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

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

Note:

- **the upload form must use the HTTP post method and must contain an enctype="multipart/form-data" attribute. This attribute ensures that the form data is encoded as mulitpart MIME data — which is required for uploading the large quantities of binary data such as image, audio, video, etc.**
- **multipart/form-data: When you use the multipart/form-data value for the enctype attribute, it allows you to upload files using the POST method. Also, it makes sure that the characters are not encoded when the form is submitted.**

File: uploader.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!";
}
?>
```

$ target_path:- specifies the directory where the file is going to be placed

PHP Download File

PHP enables you to download file easily using built-in readfile() function. The readfile() function reads a file and writes it to the output buffer.

PHP readfile() function

Syntax

int readfile (string $filename [, bool $use_include_path = false [, resource $context]])

- $filename: represents the file name

- $use_include_path: it is the optional parameter. It is by default false. You can set it to true to the search the file in the included_path.
- $context: represents the context stream resource.
- int: it returns the number of bytes read from the file.

PHP Download File Example: Text File

File: download1.php

```
<?php
$file_url = 'https://edunetfoundation.org/wp-content/uploads/2022/06/edunet-logo-white.png';
header('Content-Type: application/octet-stream');
header("Content-Transfer-Encoding: utf-8");
header("Content-disposition: attachment; filename=\"" . basename($file_url) . "\"");
readfile($file_url);
?>
```

PHP Cookie

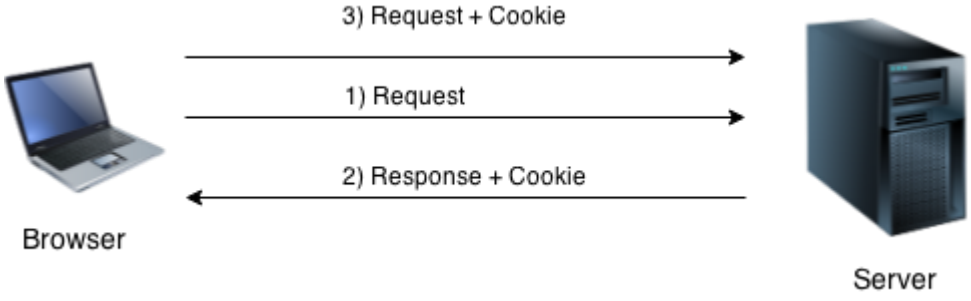
What is Cookie

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

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 transparently supports HTTP cookies.

There are three steps involved in identifying returning users −

- Server script sends a set of cookies to the browser. For example name, age, or identification number etc.
- Browser stores this information on local machine for future use.
- When next time browser sends any request to web server then it sends those cookies information to the server and server uses that information to identify the user.

Create Cookies With PHP

A cookie is created with the 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

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

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

Here is the detail of all the arguments −

- **Name –** This sets the name of the cookie and is stored in an environment variable called HTTP_COOKIE_VARS. This variable is used while accessing cookies.
- **Value –** This sets the value of the named variable and is the content that you actually want to store.
- **Expiry –** This specify a future time in seconds since 00:00:00 GMT on 1st Jan 1970. After this time cookie will become inaccessible. If this parameter is not set then cookie will automatically expire when the Web Browser is closed.
- **Path –** This specifies the directories for which the cookie is valid. A single forward slash character permits the cookie to be valid for all directories.
- **Domain –** This can be used to specify the domain name in very large domains and must contain at least two periods to be valid. All cookies are only valid for the host and domain which created them.
- **Security –** This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Following example will create two cookies name and age these cookies will be expired after one hour.

```
<?php
  setcookie("name", "John Watkin", time()+3600, "/","", 0);
  setcookie("age", "36", time()+3600, "/", "",  0);
?>
<html>
<head>
              <title>Setting Cookies with PHP</title>
            </head>
            <body>
              <?php echo "Set Cookies"?>
            </body>
```

```
</html>
```

Accessing Cookies with PHP

PHP provides many ways to access cookies. Simplest way is to use either $_COOKIE or $HTTP_COOKIE_VARS variables. Following example will access all the cookies set in above example.

```
<html>

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

  <body>

    <?php
      echo $_COOKIE["name"]. "<br />";

      /* is equivalent to */
      echo $HTTP_COOKIE_VARS["name"]. "<br />";

      echo $_COOKIE["age"] . "<br />";

      /* is equivalent to */
      echo $HTTP_COOKIE_VARS["age"] . "<br />";
    ?>

  </body>
</html>
```

John Watkin
36

You can use isset() function to check if a cookie is set or not.

```
<html>

  <head>

    <title>Accessing Cookies with PHP</title>

  </head>

  <body>

    <?php

      if( isset($_COOKIE["John Watkin "]))
```

```
      echo "Welcome " . $_COOKIE["John Watkin "] . "<br />";
    else
      echo "Sorry... Not recognized" . "<br />";
  ?>
 </body>
</html>
```

Deleting Cookie with PHP

Officially, to delete a cookie you should call setcookie() with the name argument only but this does not always work well, however, and should not be relied on.

It is safest to set the cookie with a date that has already expired −

```
<?php
  setcookie( "name", "", time()- 60, "/","", 0);
  setcookie( "age", "", time()- 60, "/","", 0);
?>
<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.

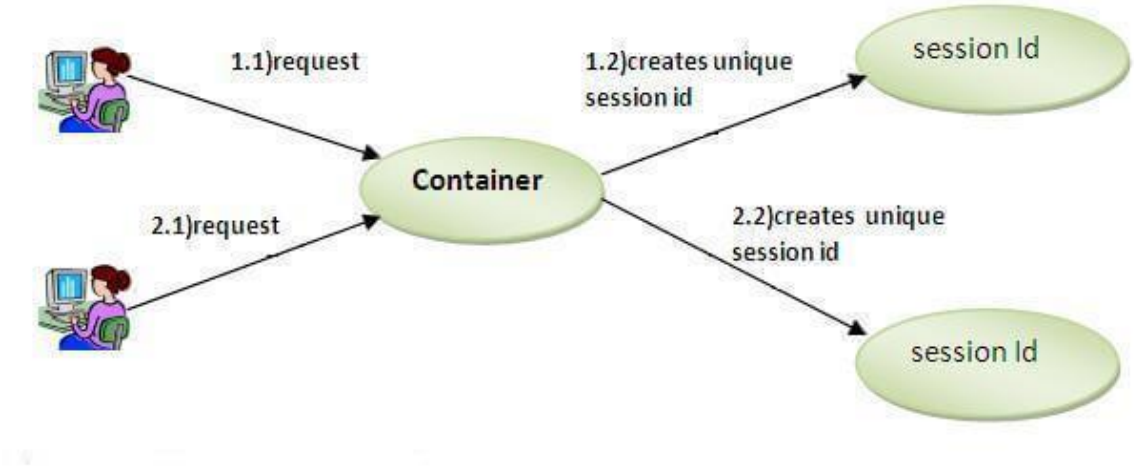
PHP session creates unique user id for each browser to recognize the user and avoid conflict between multiple browsers.

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.

So; Session variables hold information about one single user, and are available to all pages in one application.



How are sessions better than cookies?

Although cookies are also used for storing user related data, they have serious security issues because cookies are stored on the user's computer and thus they are open to attackers to easily modify the content of the cookie. Addition of harmful data by the attackers in the cookie may result in the breakdown of the application.

Apart from that cookies affect the performance of a site since cookies send the user data each time the user views a page. Every time the browser requests a URL to the server, all the cookie data for that website is automatically sent to the server within the request.

What Happens When You Start a Session in PHP?

The following things occur when a session is started:

- It creates a random 32 digit hexadecimal value as an identifier for that particular session. The identifier value will look something like 4af5ac6val45rf2d5vre58sd648ce5f7.
- It sends a cookie named PHPSESSID to the user's system. As the name gives out, the PHPSESSID cookie will store the unique session id of the session.

- A temporary file gets created on the server and is stored in the specified directory. It names the file on the hexadecimal id value prefixed with sess_. Thus, the above id example will be held in a file called sess_4af5ac6val45rf2d5vre58sd648ce5f7.

PHP will access the PHPSESSID cookie and get the unique id string to get session variables' values. It will then look into its directory for the file named with that string.

When you close the browser or the website, it terminates the session after a certain period of a predetermined time.

Start a PHP Session

A session is started with the session_start() function.

Session variables are set with the PHP global variable: $_SESSION.

Now, let's create a new page called "demo_session1.php". In this page, we start a new PHP session and set some session variables:

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Note: The session_start() function must be the very first thing in your document. Before any HTML tags.

Get PHP Session Variable Values

Next, we create another page called "demo_session2.php". From this page, we will access the session information we set on the first page ("demo_session1.php").

Notice that session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session_start()).

Also notice that all session variable values are stored in the global $_SESSION variable:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

Another way to show all the session variable values for a user session is to run the following code:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
```

```
print_r($_SESSION);
?>


</body>
</html>
```

Modify a PHP Session Variable

To change a session variable, just overwrite it:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>


<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>


</body>
</html>
```

Destroy a PHP Session

To remove all global session variables and destroy the session, use session_unset() and session_destroy():

Example

```
<?php
session_start();
?>
<!DOCTYPE html>
```

```
<html>
<body>


<?php
// remove all session variables
session_unset();


// destroy the session
session_destroy();
?>


</body>
</html>
```

PHP - Regular Expressions

Regular Expression

Regular expressions are commonly known as regex. These are nothing more than a pattern or a sequence of characters, which describe a special search pattern as text string.

Regular expression allows you to search a specific string inside another string. Even we can replace one string by another string and also split a string into multiple chunks. They use arithmetic operators (+, -, ^) to create complex expressions.

By default, regular expressions are case sensitive.

Operators in Regular Expression

Operator	Description
^	It indicates the start of string.
$	It indicates the end of the string.
.	It donates any single character.
()	It shows a group of expressions.
[]	It finds a range of characters, **e.g.**, [abc] means a, b, or c.
[^]	It finds the characters which are not in range, **e.g.**, [^xyz] means NOT x, y, or z.
-	It finds the range between the elements, **e.g.**, [a-z] means a through z.
|	It is a logical OR operator, which is used between the elements. **E.g.**, a|b, which means either a OR b.
?	It indicates zero or one of preceding character or element range.
*	It indicates zero or more of preceding character or element range.
+	It indicates zero or more of preceding character or element range.
{n}	It denotes at least n times of preceding character range. For example - n{3}
{n, }	It denotes at least n, but it should not be more than m times, e.g., n{2,5} means 2 to 5 of n.
{n, m}	It indicates at least n, but it should not be more than m times. For example - n{3,6} means 3 to 6 of n.
\	It denotes the escape character.

Special character class in Regular Expression

Special Character	Description
\n	It indicates a new line.
\r	It indicates a carriage return.
\t	It represents a tab.
\v	It represents a vertical tab.
\f	It represents a form feed.
\xxx	It represents an octal character.
\xxh	It denotes hexadecimal character hh.

PHP offers two sets of regular expression functions:

i. POSIX Regular Expression
ii. PERL Style Regular Expression

POSIX Regular Expression

The structure of POSIX regular expression is similar to the typical arithmetic expression: several operators/elements are combined together to form more complex expressions.

The simplest regular expression is one that matches a single character inside the string. For example - "g" inside the toggle or cage string.

PERL Style Regular Expression

Perl-style regular expressions are much similar to POSIX. The POSIX syntax can be used with Perl-style regular expression function interchangeably. The quantifiers introduced in POSIX section can also be used in PERL style regular expression.

Metacharacters

A metacharacter is an alphabetical character followed by a backslash that gives a special meaning to the combination.

For example - '\d' metacharacter can be used search large money sums: /([\d]+)000/. Here /d will search the string of numerical character.

Below is the list of metacharacters that can be used in PERL Style Regular Expressions –

Character	**Description**
.	Matches a single character
\s	It matches a whitespace character like space, newline, tab.
\S	Non-whitespace character
\d	It matches any digit from 0 to 9.
\D	Matches a non-digit character.
\w	Matches for a word character such as - a-z, A-Z, 0-9, _
\W	Matches a non-word character.
[aeiou]	It matches any single character in the given set.
[^aeiou]	It matches any single character except the given set.
(foo|baz|bar)	Matches any of the alternatives specified.

Modifiers

There are several modifiers available, which makes the work much easier with a regular expression. For example - case-sensitivity or searching in multiple lines, etc.

Below is the list of modifiers used in PERL Style Regular Expressions –

Character	Description
i	Makes case insensitive search
m	It specifies that if a string has a carriage return or newline characters, the $ and ^ operator will match against a newline boundary rather than a string boundary.
o	Evaluates the expression only once
s	It allows the use of .(dot) to match a newline character
x	This modifier allows us to use whitespace in expression for clarity.
g	It globally searches all matches.
cg	It allows the search to continue even after the global match fails.

PHP Regexp POSIX Function

PHP currently provides seven functions to search strings using POSIX-style regular expression

Function	Description
preg_match()	This function searches the pattern inside the string and returns **true** if the pattern exists otherwise returns **false**.
preg_match_all()	This function matches all the occurrences of pattern in the string.
preg_replace()	The preg_replace() function is similar to the ereg_replace() function, except that the regular expressions can be used in search and replace.
preg_split()	This function exactly works like split() function except the condition is that it accepts regular expression as an input parameter for pattern. Mainly it divides the string by a regular expression.
preg_grep()	The preg_grep() function finds all the elements of input_array and returns the array elements matched with regexp (relational expression) pattern.
preg_quote()	Quote the regular expression characters.

PHP preg_match() function

The preg_match() function is a built-in function of PHP that performs a regular expression match. This function searches the string for pattern, and returns true if the pattern exists otherwise returns false.

Generally, the searching starts from the beginning of $subject string parameter. The optional parameter $offset is used to start the search from the specified position.

Syntax

int preg_match (string $pattern, string $subject, array $matches, int $flags, int $offset)

> ***Note: $offset is an optional parameter that specifies the position from where to begin the search.***

Parameters

This function accepts five parameters, which are described below:

Pattern

It is a string type parameter. This parameter holds the pattern to search as a string.

subject

This parameter holds the input string in which we search for pattern.

matches

If matches parameter is provided, it will contain the search results.

matches[0] - It will hold the text, which matched with the complete pattern.

matches[1] - It will contain the text, which matched with the first captured parenthesized subpattern, and so on.

Flags

The flags can have the following flags given below:

- **PREG_OFFSET_CAPTURE:** If this flag is passed in preg_match(), for every occurring match the appendant string offset will also return.
- **PREG_UNMATCHED_AS_NULL:** If this flag is passed in preg_match(), unmatched subpattern will be reported as NULL, otherwise they will be reported as empty string.

Offset

By default, the search starts from the beginning of the $subject parameter. The offset parameter is used to specify the place where the searching will start. It is an optional parameter.

Return Type

The preg_match() function returns true if pattern matches otherwise, it returns false.

Examples

```
<?php
   //initialize a variable of string type
   $site = "edunetfoundation";
```

```
    preg_match('/(edu)(net)(foundation)/', $site, $matches, PREG_OFFSET_CAPTURE);
    //display the matches result

    echo "<pre>";

    print_r($matches);
    echo "</pre>";
?>
```

PHP preg_replace() function

The preg_replace() function is a built-in function of PHP. It is used to perform a regular expression search and replace.

This function searches for pattern in subject parameter and replaces them with the replacement.

Syntax

preg_replace (mixed $pattern, mixed $replacement, mixed $subject, int $limit, int $count)

Parameters

This function accepts five parameters, which are described below:

Pattern

This parameter can be either a string or an array with strings. It holds the pattern to search in subject parameter.

replacement

It is a string or an array with strings parameter. This parameter replaces the pattern matched in subject parameter. It is a mandatory parameter.

If the replacement parameter is a string and the pattern parameter is an array, all patterns will be replaced by that string.

If both replacement and pattern parameters are arrays, each pattern will be replaced by the replacement counterpart.

If the replacement array consists of fewer elements than the pattern array, any extra pattern will be replaced by an empty string.

subject

The subject parameter can also be either a string or an array of string to search and replace.

If the subject is an array, the search and replacement are performed on every entry of subject, and the returned value will also be an array.

limit

The limit is an optional parameter that specifies the maximum possible replacement for each pattern. The default value of limit is -1, which means no limit.

count

It is an optional parameter. If this parameter is passed, this variable will contain the number of replacements done. This parameter added in PHP 5.1.0.

Return Type

The preg_replace() function returns an array if the subject parameter is an array otherwise it returns a string.

After the replacement has done, the modified string will be returned.

If any matches do not find, the string will remain unchanged.

Examples

Simple Replacing

```
$res = preg_replace('/abc/', 'efg', $string);      #Replace all 'abc' with 'efg'

$res = preg_replace('/abc/i', 'efg', $string);     #Replace with case-insensitive matching

$res = preg_replace('/\s+/', '', $string);     #Strip all whitespace
```

See the detailed examples to understand the preg_replace() function practically:

Example using backreference followed by numeric literals

```
<?php
   $date = 'May 29, 2020';
   $pattern = '/(\w+) (\d+), (\d+)/i';
   $replacement = '${1} 5,$3';

   //display the result returned by preg_replace
   echo preg_replace($pattern, $replacement, $date);
?>
```

preg_match_all()

Definition and Usage

The preg_match_all() function matches all occurrences of pattern in string.

It will place these matches in the array pattern_array in the order you specify using the optional input parameter order. There are two possible types of order −

- PREG_PATTERN_ORDER − is the default if the optional order parameter is not included. PREG_PATTERN_ORDER specifies the order in the way that you might think most logical; $pattern_array[0] is an array of all complete pattern matches, $pattern_array[1] is an array of all strings matching the first parenthesized regexp, and so on.
- PREG_SET_ORDER − will order the array a bit differently than the default setting. $pattern_array[0] will contain elements matched by the first parenthesized regexp, $pattern_array[1] will contain elements matched by the second parenthesized regexp, and so on.

Return Value

Returns the number of matchings.

Syntax:

int preg_match_all (string pattern, string string, array pattern_array [, int order]);

Example

Following is the piece of code, copy and paste this code into a file and verify the result.

```
<?php
  $userinfo = "Name: <b>John Poul</b> <br> Title: <b>PHP Guru</b>";
  preg_match_all ("/<b>(.*)<\/b>/U", $userinfo, $pat_array);

  print $pat_array[0][0]." <br> ".$pat_array[0][1]."\n";
?>
```

preg_split()

Definition and Usage

The preg_split() function operates exactly like split(), except that regular expressions are accepted as input parameters for pattern.

If the optional input parameter limit is specified, then only limit number of substrings are returned.

flags can be any combination of the following flags −

- PREG_SPLIT_NO_EMPTY − If this flag is set, only non-empty pieces will be returned by preg_split().
- PREG_SPLIT_DELIM_CAPTURE − If this flag is set, parenthesized expression in the delimiter pattern will be captured and returned as well.
- PREG_SPLIT_OFFSET_CAPTURE − If this flag is set, for every occurring match the appendant string offset will also be returned.

Return Value

Returns an array of strings after splitting up a string.

Syntax

array preg_split (string pattern, string string [, int limit [, int flags]]);

Example

Following is the piece of code, copy and paste this code into a file and verify the result.

```
<?php
  $ip = "123.456.789.000"; // some IP address
  $iparr = preg_split ("/\./", $ip);
  print "$iparr[0] <br />";
  print "$iparr[1] <br />" ;
```

```
  print "$iparr[2] <br />" ;
  print "$iparr[3] <br />" ;
?>
```

preg_quote()

Definition and Usage

preg_quote() takes str and puts a backslash in front of every character that is part of the regular expression syntax.

Return Value

Returns the quoted string.

Syntax:

string preg_quote (string $str [, string $delimiter]);

Example

Following is the piece of code, copy and paste this code into a file and verify the result.

```
<?php
  $keywords = '$40 for a g3/400';
  $keywords = preg_quote($keywords, '/');

  echo $keywords;
?>
```

PHP Form Handling

Get and Post Methods in PHP

PHP provides two methods through which a client (browser) can send information to the server. These methods are given below, and discussed in detail:

i. GET method
ii. POST method

Get and Post methods are the HTTP request methods used inside the <form> tag to send form data to the server.

HTTP protocol enables the communication between the client and the server where a browser can be the client, and an application running on a computer system that hosts your website can be the server.

GET method

The GET method is used to submit the HTML form data. This data is collected by the predefined $_GET variable for processing.

The information sent from an HTML form using the GET method is visible to everyone in the browser's address bar, which means that all the variable names and their values will be displayed in the URL. Therefore, the get method is not secured to send sensitive information.

The below code will display an HTML form containing two input fields and a submit button. In this HTML form, we used the method = "get" to submit the form data.

file: test1.html

```
<html>
  <body>
    <form action = "gettest.php" method = "GET">
      Username: <input type = "text" name = "username" /> <br>
      Blood Group: <input type = "text" name = "bloodgroup" /> <br>
      <input type = "submit" />
    </form>
  </body>
</html>
```

Create gettest.php file, which will accept the data sent by HTML form.

file: gettest.php

```
<html>
  <body>

    Welcome <?php echo $_GET["username"]; ?> </br>
    Your blood group is: <?php echo $_GET["bloodgroup"]; ?>

  </body>
</html>
```

When the user will click on Submit button after filling the form, the URL sent to the server could look for result.

Advantages of GET method (method = "get")

- You can bookmark the page with the specific query string because the data sent by the GET method is displayed in URL.
- GET requests can be cached.
- GET requests are always remained in the browser history.

Disadvantages of GET Method

- The GET method should not be used while sending any sensitive information.
- A limited amount of data can be sent using method = "get". This limit should not exceed 2048 characters.
- For security reasons, never use the GET method to send highly sensitive information like username and password, because it shows them in the URL.
- The GET method cannot be used to send binary data (such as images or word documents) to the server.

POST method

Similar to the GET method, the POST method is also used to submit the HTML form data. But the data submitted by this method is collected by the predefined superglobal variable $_POST instead of $_GET.

Unlike the GET method, it does not have a limit on the amount of information to be sent. The information sent from an HTML form using the POST method is not visible to anyone.

Example

The below code will display an HTML form containing two input fields and a submit button. In this HTML form, we used the method = "post" to submit the form data.

file: test2.html

```
<html>
  <body>
    <form action = "posttest.php" method = "post">
      Username: <input type = "text" name = "username" /> <br>
      Blood Group: <input type = "text" name = "bloodgroup" /> <br>
      <input type = "submit" />
    </form>
  </body>
</html>
```

Now create posttest.php file to accept the data sent by HTML form.

file: posttest.php

```
<html>
  <body>

    Welcome <?php echo $_POST["username"]; ?> </br>
    Your blood group is: <?php echo $_POST["bloodgroup"]; ?>

  </body>
</html>
```

When the user will click on Submit button after filling the form, the URL sent to the server could give the output.

Advantages of POST method (method = "post")

- The POST method is useful for sending any sensitive information because the information sent using the POST method is not visible to anyone.
- There is no limitation on size of data to be sent using the POST Method. You can send a large amount of information using this method.
- Binary and ASCII data can also be sent using the POST method.

- Data security depends on the HTTP protocol because the information sent using the POST method goes through the HTTP header. By using secure HTTP, you can ensure that your data is safe.

Disadvantages of POST Method

- POST requests do not cache.
- POST requests never remain in the browser history.
- It is not possible to bookmark the page because the variables are not displayed in URL.

$_REQUEST variable

The $_REQUEST variable is a superglobal variable, which can hold the content of both $_GET and $_POST variable. In other words, the PHP $_REQUEST variable is used to collect the form data sent by either GET or POST methods. It can also collect the data for $_COOKIE variable because it is not a method-specific variable.

Form Validation in PHP

An HTML form contains various input fields such as text box, checkbox, radio buttons, submit button, and checklist, etc. These input fields need to be validated, which ensures that the user has entered information in all the required fields and also validates that the information provided by the user is valid and correct.

There is no guarantee that the information provided by the user is always correct. PHP validates the data at the server-side, which is submitted by HTML form. You need to validate a few things:

- Empty String
- Validate String
- Validate Numbers
- Validate Email
- Validate URL
- Input length

i. **Empty String**

The code below checks that the field is not empty. If the user leaves the required field empty, it will show an error message. Put these lines of code to validate the required field.

```
if (empty ($_POST["name"])) {
   $errMsg = "Error! You didn't enter the Name.";
        echo $errMsg;
} else {
   $name = $_POST["name"];
}
```

ii. **Validate String**

The code below checks that the field will contain only alphabets and whitespace, for example - name. If the name field does not receive valid input from the user, then it will show an error message:

```
$name = $_POST ["Name"];
if (!preg_match ("/^[a-zA-z]*$/", $name) ) {
   $ErrMsg = "Only alphabets and whitespace are allowed.";
        echo $ErrMsg;
} else {
   echo $name;
}
```

Validate Number

The below code validates that the field will only contain a numeric value. For example - Mobile no. If the Mobile no field does not receive numeric data from the user, the code will display an error message:

```
$mobileno = $_POST ["Mobile_no"];

if (!preg_match ("/^[0-9]*$/", $mobileno) ){

   $ErrMsg = "Only numeric value is allowed.";

   echo $ErrMsg;

} else {

   echo $mobileno;

}
```

Validate Email

A valid email must contain @ and . symbols. PHP provides various methods to validate the email address. Here, we will use regular expressions to validate the email address.

The below code validates the email address provided by the user through HTML form. If the field does not contain a valid email address, then the code will display an error message:

```
$email = $_POST ["Email"];

$pattern = "^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$^";

if (!preg_match ($pattern, $email) ){

   $ErrMsg = "Email is not valid.";

        echo $ErrMsg;

} else {

   echo "Your valid email address is: " .$email;

}
```

Input Length Validation

The input length validation restricts the user to provide the value between the specified range, for Example - Mobile Number. A valid mobile number must have 10 digits.

The given code will help you to apply the length validation on user input:

```
$mobileno = strlen ($_POST ["Mobile"]);

$length = strlen ($mobileno);


if ( $length < 10 && $length > 10) {
   $ErrMsg = "Mobile must have 10 digits.";
        echo $ErrMsg;
} else {
   echo "Your Mobile number is: " .$mobileno;
}
```

Validate URL

The below code validates the URL of website provided by the user via HTML form. If the field does not contain a valid URL, the code will display an error message, i.e., "URL is not valid".

```
$websiteURL = $_POST["website"];

if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "URL is not valid";
  echo $websiteErr;
} else {
   echo "Website URL is: " .$websiteURL;
}
```

Button Click Validate

The below code validates that the user click on submit button and send the form data to the server one of the following method - get or post.

```
if (isset ($_POST['submit']) {
   echo "Submit button is clicked.";
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      echo "Data is sent using POST method ";
```

```
    }
} else {
    echo "Data is not submitted";
}
```

Note: Remember that validation and verification both are different from each other.

Now we will apply all these validations to an HTML form to validate the fields. Thereby you can learn in detail how these codes will be used to validation form.

Create a registration form using HTML and perform server-side validation. Follow the below instructions as given:

Create and validate a Registration form

```
<!DOCTYPE html>
<html>
<head>
<style>
.error {color: #FF0001;}
</style>
</head>
<body>


<?php
// define variables to empty values
$nameErr = $emailErr = $mobilenoErr = $genderErr = $websiteErr = $agreeErr = "";
$name = $email = $mobileno = $gender = $website = $agree = "";


//Input fields validation
if ($_SERVER["REQUEST_METHOD"] == "POST") {


//String Validation
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
```

```
} else {
   $name = input_data($_POST["name"]);
      // check if name only contains letters and whitespace
      if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
         $nameErr = "Only alphabets and white space are allowed";
      }
}



//Email Validation
if (empty($_POST["email"])) {
      $emailErr = "Email is required";
} else {
      $email = input_data($_POST["email"]);
      // check that the e-mail address is well-formed
      if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
         $emailErr = "Invalid email format";
      }
 }

//Number Validation
if (empty($_POST["mobileno"])) {
      $mobilenoErr = "Mobile no is required";
} else {
      $mobileno = input_data($_POST["mobileno"]);
      // check if mobile no is well-formed
      if (!preg_match ("/^[0-9]*$/", $mobileno) ) {
      $mobilenoErr = "Only numeric value is allowed.";
      }
```

```
    //check mobile no length should not be less and greator than 10
    if (strlen ($mobileno) != 10) {
        $mobilenoErr = "Mobile no must contain 10 digits.";
        }
}

//URL Validation
if (empty($_POST["website"])) {
    $website = "";
} else {
        $website = input_data($_POST["website"]);
        // check if URL address syntax is valid
        if (!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-9+&@#\/%=~_|]/i",$website)) {
            $websiteErr = "Invalid URL";
        }
}

//Empty Field Validation
if (empty ($_POST["gender"])) {
        $genderErr = "Gender is required";
} else {
        $gender = input_data($_POST["gender"]);
}

//Checkbox Validation
if (!isset($_POST['agree'])){
        $agreeErr = "Accept terms of services before submit.";
} else {
        $agree = input_data($_POST["agree"]);
```

```
    }
}
function input_data($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>

<h2>Registration Form</h2>
<span class = "error">* required field </span>
<br><br>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]); ?>" >
   Name:
   <input type="text" name="name">
   <span class="error">* <?php echo $nameErr; ?> </span>
   <br><br>
   E-mail:
   <input type="text" name="email">
   <span class="error">* <?php echo $emailErr; ?> </span>
   <br><br>
   Mobile No:
   <input type="text" name="mobileno">
   <span class="error">* <?php echo $mobilenoErr; ?> </span>
   <br><br>
   Website:
   <input type="text" name="website">
   <span class="error"><?php echo $websiteErr; ?> </span>
```

```
   <br><br>
   Gender:
   <input type="radio" name="gender" value="male"> Male
   <input type="radio" name="gender" value="female"> Female
   <input type="radio" name="gender" value="other"> Other
   <span class="error">* <?php echo $genderErr; ?> </span>
   <br><br>
   Agree to Terms of Service:
   <input type="checkbox" name="agree">
   <span class="error">* <?php echo $agreeErr; ?> </span>
   <br><br>
   <input type="submit" name="submit" value="Submit">
   <br><br>
</form>

<?php
   if(isset($_POST['submit'])) {
   if($nameErr == "" && $emailErr == "" && $mobilenoErr == "" && $genderErr == "" &&
$websiteErr == "" && $agreeErr == "") {
      echo "<h3 color = #FF0001> <b>You have sucessfully registered.</b> </h3>";
      echo "<h2>Your Input:</h2>";
      echo "Name: " .$name;
      echo "<br>";
      echo "Email: " .$email;
      echo "<br>";
      echo "Mobile No: " .$mobileno;
      echo "<br>";
      echo "Website: " .$website;
      echo "<br>";
      echo "Gender: " .$gender;
```

```
    } else {
        echo "<h3> <b>You didn't filled up the form correctly.</b> </h3>";
    }
    }
?>

</body>
</html>
```

Working with Different types of Mouse Events

Events that occur when the mouse interacts with the HTML document belongs to the MouseEvent

Object.

The MouseEvent inherits all the properties and methods from:

- The UiEvent
- The Event Object

Event Types

These event types belong to the MouseEvent Object

Event	Description
onclick	The event occurs when the user clicks on an element
oncontextmenu	The event occurs when the user right-clicks on an element to open a context menu
ondblclick	The event occurs when the user double-clicks on an element

onmousedown	The event occurs when the user presses a mouse button over an element
onmouseenter	The event occurs when the pointer is moved onto an element
onmouseleave	The event occurs when the pointer is moved out of an element
onmousemove	The event occurs when the pointer is moving while it is over an element
onmouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children
onmouseover	The event occurs when the pointer is moved onto an element, or onto one of its children
onmouseup	The event occurs when a user releases a mouse button over an element

Object Oriented Programming with PHP 5

Object-oriented programming is a programming model organized around Object rather than the actions and data rather than logic.

OOP stands for Object-Oriented Programming.

Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

Object Oriented Programming Principles

The three major principles of OOP are;

Encapsulation –

- this is concerned with hiding the implementation details and only exposing the methods. The main purpose of encapsulation is to
  Reduce software development complexity – by hiding the implementation details and only exposing the operations, using a class becomes easy.

- Protect the internal state of an object – access to the class variables is via methods such as get and set, this makes the class flexible and easy to maintain.
- The internal implementation of the class can be changed without worrying about breaking the code that uses the class.

Inheritance –

- this is concerned with the relationship between classes. The relationship takes the form of a parent and child. The child uses the methods defined in the parent class. The main purpose of inheritance is;
  - ✓ Re-usability–a number of children, can inherit from the same parent. This is very useful when we have to provide common functionality such as adding, updating and deleting data from the database.

Polymorphism –

- this is concerned with having a single form but many different implementation ways. The main purpose of polymorphism is;
  - ✓ Simplify maintaining applications and making them more extendable.

OOPs Concepts in PHP

PHP is an object oriented scripting language; it supports all of the above principles. The above principles are achieved via;

- **Encapsulation –** via the use of "get" and "set" methods etc.
- **Inheritance –** via the use of extends keyword
- **Polymorphism –** via the use of implements keyword

Now that we have the basic knowledge of OOP and how it is supported in PHP, let us look at examples that implement the above principles

Before diving deep into PHP, you have to learn about the basic OOPs concepts in PHP that are the pillars of OOPS:

S.No	Terminology	Definition
1	Class	The user-defined data type combines variables, local data, and functions.
2	Objects	These are local instances created by the developer to access the content of the class
3	Member Variable	These are none other than variables defined inside a class and are only accessed by member functions
4	Member Function	These are none other than functions defined inside a class and are generally used to access data objects
5	Inheritance	It is one of the main properties of object-oriented programming, where traits and properties of a superior class are given to a subclass. The subclass inherits member function and variables present in the parent class

6	Parent class	The main class generally inherits some of its traits and properties to its child class or other class. These types of classes are also known as superclasses or base classes.
7	Child class	A subclass inherits some of its traits and properties from its parent class or another class, these types of classes are also known as subclasses or derived classes.
8	Polymorphism	It is one of the base concepts of object-oriented programming, which states that a single function can be used for various other reasons. In this, the name of the function remains the same, but arguments can be different
9	Overloading	It is a type of polymorphism where a single function class is overloaded with different forms of implementation depending on certain types of arguments, or we can overload the same type of function using different implementations.
10	Data abstraction	It is a specific type of data where the data implementation details are hidden.
11	Encapsulation	Process in which all data and member functions are wrapped together to create a new object.
12	Constructor	A special function is automatically called when an object of the same class is formed.
13	Destructor	A special function is automatically called when an object goes out of scope or gets deleted.

- **Class:**
  The class is the blueprint that is used to hold the objects along with their behavior and properties. In PHP, the class name should be the same as that of the file name with which it has saved the program. A class is a user-defined data type that consists of two entities: Data Members and Member Functions.
  The class keyword is used to define a class in PHP. Below are the rules for creating a class in PHP.
  - ✓ The class name should start with a letter
  - ✓ The class name cannot be a PHP reserved word
  - ✓ The class name cannot contain spaces

  Syntax to Create Class in PHP
  Syntax:

```
<?php
  class myFirstClass {
    var $ var _ a ;
    var $ var _ b = " constant string " ;

    function classFunction ( $ parameter 1 , $ parameter 2 ) {
      [ . . . . . ]
```

```
    }
    [. . . . . . . . . . . . . . .]

  }

?>
```

Parameters: -

S.No	Parameters	description
1	Class	To declare a class, we have to use the keyword class and then the name of the class that we want to declare.
2	Variable declaration	To declare a variable, we have to declare keyword var followed by $ convention and then the name of the declared variable. These are also known as member variables. These are also called properties
3	Function declaration	To declare a function, we have to declare the keyword function following the name of the declared function. These are also known as member functions, but these functions are only accessible to class only. These are also called methods
4	Braces	We have to enclose our class with curly braces { }

In PHP, to see the contents of the class, use var_dump(). The var_dump() function is used to display the structured information (type and value) about one or more variables.

Syntax:

```
var_dump($obj);
```

- **Data Members:**
  Data Members are the variables that can be of the data type var in PHP. Data Members act as the data for the source code with which you can meddle around. Data members can have three types of visibility modes that decide the access permission of these members. These modes are private, protected, and public.
- **Member Functions:**

Those data members that have visibility mode as private, and cannot be accessed directly by the class objects. In such cases, member functions come into play. Those functions that are specifically created to access private data members are known as member functions.

- **Objects:**
  A class defines an individual instance of the data structure. We define a class once and then make many objects that belong to it. Objects are also known as an instance.
  An object is something that can perform a set of related activities.
  Syntax:

```
<?php
class MyClass
{
      // Class properties and methods go here
}
$obj = new MyClass;
var_dump($obj);
?>
```

Creating Objects in PHP to access the class

```
$employee_1 = new employees;

$employee_2 = new employees;

$employee_3 = new employees;
```

To access class employees' member variables and member function, we have created 3 objects and assigned them to class employees using a new keyword.

Calling member functions using objects

```
$employee_1 -> set_name (" JOHN " );
$employee_1 -> set_salary (" 10000 " );<
$employee_1 -> set_profile (" audit manager " );

$employee_2 -> set_name (" DOE " );
$employee_2 -> set_salary ( " 150000 " );
$employee_2 -> set_profile (" engineer " );

$employee_3 -> set_name ( " NINA " );
$employee_3 -> set_salary ( " 70000 " );
$employee_3 -> set_profile ( " accountant " );
```

Once after creating objects we can use these objects to call the member functions inside the class, and using these member functions we can assign the values to member variables

```
$employee_1->get_name();
$employee_1->get_salary();
$employee_1->get_profile();

$employee_2->get_name();
$employee_2->get_salary();
$employee_2->get_profile();

$employee_3->get_name();
$employee_3->get_salary();
$employee_3->get_profile();
```

We have called the other member function with the declared objects to print the output.

Define a class

To define a class, you specify the class keyword followed by a name like this:

```
<?php

class ClassName
{
   //...
}
```

For example, the following defines a new class called BankAccount:

<?php

```
class BankAccount
{
}
```

From the BankAccount class, you can create a new bank account object by using the new keyword like this:

```
<?php

class BankAccount
{
}

$account = new BankAccount();
```

In this syntax, the $account is a variable that references the object created by the BankAccount class. The parentheses that follow the BankAccount class name are optional. Therefore, you can create a new BankAccount object like this:

```
$account = new BankAccount;
```

The process of creating a new object is also called instantiation. In other words, you instantiate an object from a class. Or you create a new object from a class.

The BankAccount class is empty because it doesn't have any state and behavior.

Add properties to a class

To add properties to the BankAccount class, you place variables inside it. For example:

```
<?php


class BankAccount
{
   public $accountNumber;
   public $balance;
}
```

The BankAccount class has two properties $accountNumber and $balance. In front of each property, you see the public keyword.

The public keyword determines the visibility of a property. In this case, you can access the property from the outside of the class.

To access a property, you use the object operator (->) like this:

```
<?php
$object->property;
```

The following example shows how to set the values of the accountNumber and balance properties:

```
<?php
class BankAccount
{
   public $accountNumber;
   public $balance;
}
$account = new BankAccount();
```

```
$account->accountNumber = 1;

$account->balance = 100;
```

Besides the public keyword, PHP also has private and protected keywords which you'll learn in the access modifiers tutorial.

Add methods to a class

The following shows the syntax for defining a method in a class:

```
<?php


class ClassName

{

        public function methodName(parameter_list)

        {

                // implementation

        }

}
```

Like a property, a method also has one of the three visibility modifiers: public, private, and protected. If you define a method without any visibility modifier, it defaults to public.

The following example defines the deposit() method for the BankAccount class:

```
<?php


class BankAccount

{

        public $accountNumber;


        public $balance;


        public function deposit($amount)

        {

                if ($amount > 0) {

                        $this->balance += $amount;

                }
```

```
        }
}
```

The deposit() method accepts an argument $amount. It checks if the $amount is greater than zero before adding it to the balance.

To call a method, you also use the object operator (->) as follows:

```
$object->method(arguments)
```

The new syntax in the deposit() method is the $this variable. The $this variable is the current object of the BankAccount class.

For example, when you create a new object $account and call the deposit() method, the $this inside the method is the $account object:

```
$account = new BankAccount();
$account->accountNumber = 1;
$account->balance = 100;
$account->deposit(100);
```

Similarly, you can add the withdraw() method to the BankAccount class as follows:

```
<?php

class BankAccount
{
        public $accountNumber;

        public $balance;

        public function deposit($amount)
        {
                if ($amount > 0) {
                        $this->balance += $amount;
                }
        }

        public function withdraw($amount)
```

```
        {
                if ($amount <= $this->balance) {
                        $this->balance -= $amount;
                        return true;
                }
         return false;

        }
}
```

The withdraw() method checks the current balance.

If the balance is less than the withdrawal amount, the withdraw() method returns false.

Example: PHP program to display the use of PHP CLASS

```
<!DOCTYPE html>
<html lang = " en ">
<head>
 <meta charse t= " UTF ? 8 ">
 <meta http ? equiv = " X ? UA ? Compatible " content = " IE = edge ">
 <meta name = " viewport " content = " width = device - width, initial ? scale = 1 .0">
 <title> PHP </title>
</head>
<body>
<? Php
class cars {
 // Properties
 var $name;
 var $color;

 // Methods
 function set_name($name) {
  $this->name = $name;
 }
```

```
 function get_name() {
   return $this->name;
 }
   function set_color($color) {
   $this->color = $color;
 }
 function get_color() {
   return $this->color;
 }
}

$BMW = new cars();
$audi = new cars();
$volvo = new cars();
$BMW->set_name('BMW ');
$BMW->set_color('red');

$audi->set_name('audi ');
$audi->set_color('blue');

$volvo->set_name('volvo ');
$volvo->set_color('black');

echo $BMW->get_name();
echo " --> ";
echo $BMW->get_color();
echo "<br>";
echo $audi->get_name();
echo " --> ";
echo $audi->get_color();
echo "<br>";
```

```
echo $volvo->get_name();
echo " --> ";
echo $volvo->get_color();
?>
</body>
</html>
```

Here in this program, we have declared a class car, with member variables $ name and $ color. We have declared member function set _ name to add a name to a certain created object, set _ color to add color to that object, get _ name to print the object, and get _ color to print the object. To access the declared class, we have declared 3 objects $ BMW, $ Audi, and $ Volvo and called the class cars using these objects. To access the member functions declared inside the class we have used the created objects with the member functions inside the class using the value as parameters for example $ BMW - > set _ name (" BMW "), $ audi - > set _ color (" blue "), $ Volvo - > set _ name (" Volvo "),in order to print the result we have used the get_ name function with class objects like $ BMW - > get _ name ( ) , $ audi - > get _ color ( ).

$ this Keyword

$ this keyword refers to the present object, and this keyword can only be used inside a member function of a class. We can use $ this keyword in two ways

1. To add a value to the declared variable, we have to use $ this property inside the set _ name function

   For example: -

```
<!DOCTYPE html>
<html>
<body>
<?php
class employee {
  public $name;
  var $salary;
  function set_name($name) {
    $this->name = $name;
  }
    function set_salary($salary) {
    $this->salary = $salary;
  }
}
$employee_1 = new employee();
$employee_1->set_name("JOHN");
$employee_1->set_salary(" $ 2000000");

$employee_2 = new employee();
$employee_2->set_name("ROCK");
$employee_2->set_salary(" $ 1200000");
```

```
echo $employee_1->name;
echo $employee_1->salary;
echo "<br>";
echo $employee_2->name;
echo $employee_2->salary;
?>
```

In the above example, we have used $ this with set_name and set_salary to add a new value to the declared variable name and salary, and later, we can get the output using echo.

2. In order to add a value to declared variable, the property of variable can be changed directly.

For example:-

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

<?php
class employee {
  public $name;
  var $salary;

}
$employee_1 = new employee();
$employee_1->name = " JOHN ";
$employee_1->salary = " $ 2000000 ";

$employee_2 = new employee();
$employee_2->name = " ROCK ";
$employee_2->salary= " $ 1200000 ";

echo $employee_1->name;
echo $employee_1->salary;
echo "<br>";
echo $employee_2->name;
```

```
echo $employee_2->salary;
?>
</body>
</html>
```

In the above example, we have used $ this directly with name and salary to add new value to declared variable name and salary, and later, we can get the output using echo.

Constructors

These are a special types of functions that are automatically called whenever an object is created. These function are created by using the _construct ( ) keyword

Syntax:

```
function __construct( $ parameter 1, $ parameter 2 ) {
  $this->name = $ parameter 1;
  $this->state = $ parameter 2;
}
```

Example:

```
<!DOCTYPE html>
<html lang = " en ">
<head>
 <meta charse t= " UTF ? 8 ">
 <meta http ? equiv = " X ? UA ? Compatible " content = " IE = edge ">
 <meta name = " viewport " content = " width = device - width, initial ? scale = 1 .0">
 <title> PHP </title>
</head>
<body>
<?php
class newconstructorclass
{
   // Constructor
   function __construct($new){
      echo 'the constructor class has been initiated' ."<br/>" ;
```

```
        $this->name = $new;
    }
        function get_name(){
        echo $this->name ."<br/>";
     }



}

// Create a new object
$obj = new newconstructorclass( " john " );
$obj -> get_name();

$obj2 = new newconstructorclass( " doe " );
$obj2 -> get_name();
$obj3 = new newconstructorclass( " mat " );
$obj3 -> get_name();

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

Here in this program, we have declared a class newconstructorclass, and inside the class, we have declared a new constructor using the _construct ( ) function. The main advantage of having a constructor class is that now we don't have to call the set function separately to add values to all the member variables. Now we can do the same when creating the object itself.

Destructors

These are special functions that are automatically called whenever an object goes out of scope or gets deleted. These functions are created by using the _destruct ( ) keyword

Syntax:

```
function __destruct() {
[ . . . . . .]
```

[. . . . . .]

}

Example:

```
<!DOCTYPE html>
<html lang = " en ">
<head>
 <meta charse t= " UTF ? 8 ">
 <meta http ? equiv = " X ? UA ? Compatible " content = " IE = edge ">
 <meta name = " viewport " content = " width = device - width, initial ? scale = 1 .0">
 <title> PHP </title>
</head>
<body>
<?php

class newdestructorclass
{
   // Destructor
   public function __destruct(){
      echo 'The class ' . __CLASS__ . '" was automatically destroyed when the created object does not have a scope to initiate';
   }

}

// Create a new object
$obj = new newdestructorclass;
?>
</body>
</html>
```

Here in this program, we have declared a class newdestructorclass. We have declared a destructor inside the class using the _destruct( ) function. It does not contain any argument. It will automatically invoke whenever the object does not have scope to work with.

ABSTRACT CLASS

An abstract class is a mix between an interface and a class. It can be define functionality as well as interface.

- Classes extending an abstract class must implement all of the abstract methods defined in the abstract class.
- An abstract class is declared the same way as classes with the addition of the 'abstract' keyword.

SYNTAX:

```
abstract class MyAbstract
{
   //Methods
}
//And is attached to a class using the extends keyword.
class Myclass extends MyAbstract
{
   //class methods
}
```

Example 1

```
<?php
abstract class a
{
abstract public function dis1();
abstract public function dis2();
}
class b extends a
{
public function dis1()
   {
      echo "nsti mumbai";
```

```
   }
   public function dis2()
   {
      echo "nsti jodhpur";
   }
}
$obj = new b();
$obj->dis1();
$obj->dis2();
?>
```

Example 2

```
<?php
abstract class Animal
{
   public $name;
   public $age;
public function Describe()
      {
            return $this->name . ", " . $this->age . " years old";
      }
abstract public function Greet();
   }
class Dog extends Animal
{
public function Greet()
      {
            return "Woof!";
      }

      public function Describe()
      {
```

```
        return parent::Describe() . ", and I'm a dog!";
    }
}
$animal = new Dog();
$animal->name = "Bob";
$animal->age = 7;
echo $animal->Describe();
echo $animal->Greet();
?>
```

Data Abstraction: Abstraction is referred to the concept of giving access to only those details that are required to perform a specific task, giving no access to the internal details of that task.

Example 1

```
<?php
abstract class Animal
{
public $name;
public $age;
public function Describe()
{
return $this->name . ", " . $this->age . " years old";
      }
abstract public function Greet();
}
class cat extends Animal
{
public function Greet()
{
return "Lion!";
}
public function Describe()
{
return parent::Describe() . ", and I'm a cat!";
}
}
$animal = new cat();
$animal->name = "Seru";
$animal->age = 5;
echo $animal->Describe();
echo $animal->Greet();

?>
```

Inheritance

It is a concept of accessing the features of one class from another class. If we inherit the class features into another class, we can access both class properties. We can extends the features of a class by using 'extends' keyword.

- It supports the concept of hierarchical classification.
- Inheritance has three types, single, multiple and multilevel Inheritance.
- PHP supports only single inheritance, where only one class can be derived from single parent class.
- We can simulate multiple inheritance by using interfaces.

```
<?php
class a
  {
    function fun1()
    {
      echo "";
    }
  }
  class b extends a
  {
    function fun2()
    {
      echo "SSSIT";
    }
  }
  $obj= new b();
  $obj->fun1();
?>
```

Interface

- An interface is similar to a class except that it cannot contain code.
- An interface can define method names and arguments, but not the contents of the methods.
- Any classes implementing an interface must implement all methods defined by the interface.
- A class can implement multiple interfaces.
- An interface is declared using the "interface" keyword.
- Interfaces can't maintain Non-abstract methods.

Example 1

```
<?php
  interface a
  {
    public function dis1();
  }
  interface b
  {
    public function dis2();
  }

class demo implements a,b
{
```

```
   public function dis1()
   {
      echo "method 1...";
   }
   public function dis2()
   {
      echo "method2...";
   }
}
$obj= new demo();
$obj->dis1();
$obj->dis2();

?>
```

Difference between Abstract class and Interfaces.

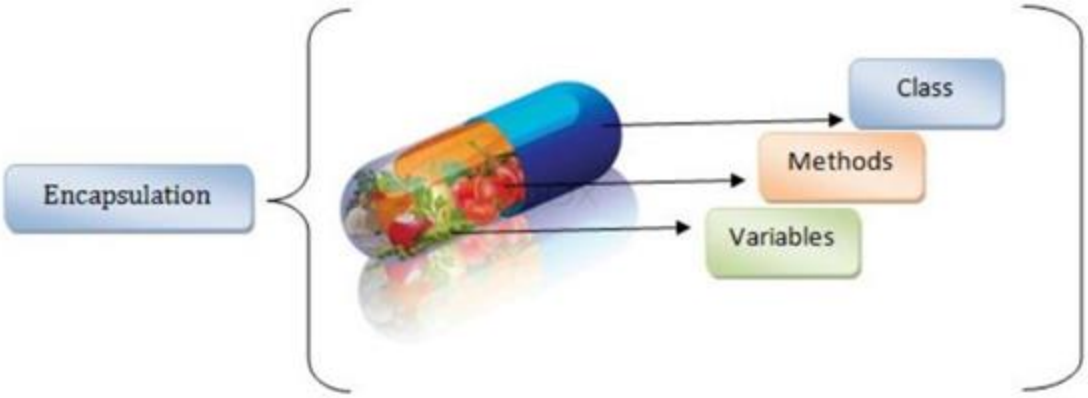
Abstract class:

- Abstract class comes under partial abstraction.
- Abstract classes can maintain abstract methods and non abstract methods.
- In abstract classes, we can create the variables.
- In abstract classes, we can use any access specifier.
- By using 'extends' keyword we can access the abstract class features from derived class.
- Multiple inheritance is not possible.

Interface:

- Interface comes under fully abstraction.
- Interfaces can maintain only abstract methods.
- In interfaces, we can't create the variables.
- In interface, we can use only public access specifier.
- By using 'implement' keyword we can get interface from derived class.
- By using interfaces multiple inheritance is possible.

Encapsulation in PHP



- Encapsulation is a concept where we encapsulate all the data and member functions together to form an object.
- Wrapping up data member and method together into a single unit is called Encapsulation.
- Encapsulation also allows a class to change its internal implementation without hurting the overall functioning of the system.
- Binding the data with the code that manipulates it.
- It keeps the data and the code safe from external interference.

Example 1

```
<?php
class person
{
public $name;
public $age;
function __construct($n, $a)
{
$this->name=$n;
$this->age=$a;
}
public function setAge($ag)
 {

$this->ag=$ag;

}
```

```
public function display()

{

echo  "welcome ".$this->name."<br/>";

return $this->age-$this->ag;

}

}

$person=new person("sonoo",28);

$person->setAge(1);

echo "You are ".$person->display()." years old";

?>
```

Final Keyword

- In PHP, Final keyword is applicable to only class and class methods. We cannot declare as Final in PHP.
- So if we declare class method as a Final then that method cannot be override by the child class.
- Same as method if we declare class as a Final then that class cannot be extended any more.

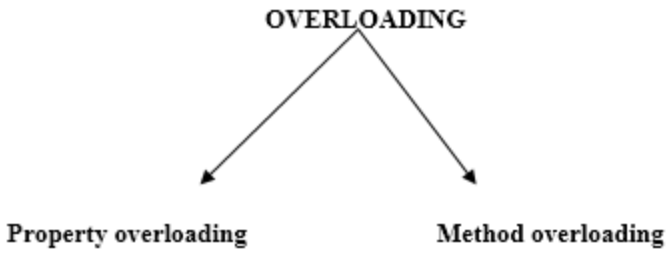
Example:

```
<?php


  class base

  {
```

```
    final public function dis1()
    {
        echo "Base class..";
    }
}
class derived extends base
{
    public function dis1()
    {
        echo "derived class";
    }
}
$obj = new derived();
$obj->dis1();

?>
```

Overloading

- Overloading in PHP provides means to dynamically create properties and methods.
- These dynamic entities are processed via magic methods, one can establish in a class for various action types.
- All overloading methods must be defined as Public.
- After creating object for a class, we can access set of entities that are properties or methods not defined within the scope of the class.
- Such entities are said to be overloaded properties or methods, and the process is called as overloading.
- For working with these overloaded properties or functions, PHP magic methods are used.
- Most of the magic methods will be triggered in object context except __callStatic() method which is used in static context.



Property overloading

- PHP property overloading allows us to create dynamic properties in object context.

- For creating those properties no separate line of code is needed.
- A property which is associated with class instance, and not declared within the scope of the class, is considered as overloaded property.

Some of the magic methods which is useful for property overloading.

- __set(): It is triggered while initializing overloaded properties.
- __get(): It is utilized for reading data from inaccessible Properties.
- __isset(): This magic method is invoked when we check overloaded properties with isset() function.
- __unset(): This function will be invoked on using PHP unset() for overloaded properties.

Static Methods

Static methods can be called directly - without creating an instance of the class first.

Static methods are declared with the static keyword:

Syntax

```
<?php
class ClassName {
  public static function staticMethod() {
    echo "Hello World!";
  }
}


?>
```

To access a static method use the class name, double colon (::), and the method name:

Syntax

```
ClassName::staticMethod();
```

Example

```
<?php
class greeting {
  public static function welcome() {
    echo "Hello World!";
  }
}
// Call static method
```

```
greeting::welcome();
?>
```

Example Explained

Here, we declare a static method: welcome(). Then, we call the static method by using the class name, double colon (::), and the method name (without creating an instance of the class first).

A class can have both static and non-static methods. A static method can be accessed from a method in the same class using the self keyword and double colon (::):

Example

```
<?php
class greeting {
  public static function welcome() {
    echo "Hello World!";
  }

  public function __construct() {
    self::welcome();
  }
}

new greeting();
?>
```

Static methods can also be called from methods in other classes. To do this, the static method should be public:

Example

```
<?php
class A {
  public static function welcome() {
    echo "Hello World!";
  }
}
```

```
class B {
  public function message() {
    A::welcome();
  }
}

$obj = new B();
echo $obj -> message();
?>
```

Access Modifiers

Properties and methods can have access modifiers which control where they can be accessed.

There are three access modifiers:

- **Public -** class members with this access modifier will be publicly accessible from anywhere, even from outside the scope of the class.
- **Private -** class members with this keyword will be accessed within the class itself. It protects members from outside class access with the reference of the class instance.
- **Protected -** same as private, except by allowing subclasses to access protected superclass members.

EXAMPLE 1: Public

```
<?php
class demo
{
public $name="Ajeet";
function disp()
{
echo $this->name."<br/>";
}
}
class child extends demo
{
```

```
function show()
{
echo $this->name;
}
}
$obj= new child;
echo $obj->name."<br/>";
$obj->disp();
$obj->show();
?>
```

EXAMPLE 2: Private

```
<?php
Class ABC
{
private $name="Sonoo";
private function show()
{
echo "This is private method of parent class";
}
}
class child extends ABC
{
function show1()
{
echo $this->name;
}
}
$obj= new child;
$obj->show();
$obj->show1();
?>
```

EXAMPLE 3: Protected

```
<?php
Class ABC
{
protected $x=500;
protected $y=100;
   function add()
{
echo $sum=$this->x+$this->y."<br/>";
}
   }
class child extends ABC
{
function sub()
{
echo $sub=$this->x-$this->y."<br/>";
}


}
$obj= new child;
$obj->add();
$obj->sub();


?>
```

EXAMPLE 4: Public,Private and Protected

```
<?php
Class ABC
{
public $name="Ajeet";
protected $profile="HR";
```

```
private $salary=5000000;
public function show()
{
echo "Welcome : ".$this->name."<br/>";
echo "Profile : ".$this->profile."<br/>";
echo "Salary : ".$this->salary."<br/>";
}
}
classchilds extends ABC
{
public function show1()
{
echo "Welcome : ".$this->name."<br/>";
echo "Profile : ".$this->profile."<br/>";
echo "Salary : ".$this->salary."<br/>";
}
}
$obj= new childs;
$obj->show1();
?>
```

PHP - Static Properties

Static properties can be called directly - without creating an instance of a class.

Static properties are declared with the static keyword:

Syntax

```
<?php
class ClassName {
  public static $staticProp = "W3Schools";
}
?>
```

To access a static property use the class name, double colon (::), and the property name:

Syntax

```
ClassName::$staticProp;
```

Let's look at an example:

Example

```
<?php
class pi {
  public static $value = 3.14159;
}

// Get static property
echo pi::$value;
?>
```

Here, we declare a static property: $value. Then, we echo the value of the static property by using the class name, double colon (::), and the property name (without creating a class first).

A class can have both static and non-static properties. A static property can be accessed from a method in the same class using the self keyword and double colon (::):

Example

```
<?php
class pi {
  public static $value=3.14159;
  public function staticValue() {
    return self::$value;
  }
}

$pi = new pi();
echo $pi->staticValue();
?>
```

To call a static property from a child class, use the parent keyword inside the child class:

Example

```
<?php
```

```
class pi {
  public static $value=3.14159;
}

class x extends pi {
  public function xStatic() {
    return parent::$value;
  }
}

// Get value of static property directly via child class
echo x::$value;

// or get value of static property via xStatic() method
$x = new x();
echo $x->xStatic();
?>
```

PHP Namespaces

Namespaces are qualifiers that solve two different problems:

i. They allow for better organization by grouping classes that work together to perform a task
ii. They allow the same name to be used for more than one class

For example, you may have a set of classes which describe an HTML table, such as Table, Row and Cell while also having another set of classes to describe furniture, such as Table, Chair and Bed. Namespaces can be used to organize the classes into two different groups while also preventing the two classes Table and Table from being mixed up.

Declaring a Namespace

Namespaces are declared at the beginning of a file using the namespace keyword:

Syntax

Declare a namespace called Html:

```
<?php
namespace Html;
?>
```

Note: A namespace declaration must be the first thing in the PHP file. The following code would be invalid:

```
<?php
echo "Hello World!";
namespace Html;
...
?>
```

Constants, classes and functions declared in this file will belong to the Html namespace:

Example

Create a Table class in the Html namespace:

```
<?php
namespace Html;
class Table {
  public $title = "";
  public $numRows = 0;
  public function message() {
    echo "<p>Table '{$this->title}' has {$this->numRows} rows.</p>";
  }
}
$table = new Table();
$table->title = "My table";
$table->numRows = 5;
?>

<!DOCTYPE html>
<html>
<body>

<?php
$table->message();
```

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

Iterables

PHP - What is an Iterable?

An iterable is any value which can be looped through with a foreach() loop.

The iterable pseudo-type was introduced in PHP 7.1, and it can be used as a data type for function arguments and function return values.

PHP - Using Iterables

The iterable keyword can be used as a data type of a function argument or as the return type of a function:

Example

Use an iterable function argument:

```
<?php
function printIterable(iterable $myIterable) {
  foreach($myIterable as $item) {
    echo $item;
  }
}

$arr = ["a", "b", "c"];
printIterable($arr);
?>
```

Example

Return an iterable:

```
<?php
function getIterable():iterable {
  return ["a", "b", "c"];
}
```

```
$myIterable = getIterable();
foreach($myIterable as $item) {
  echo $item;
}
?>
```

PHP - Creating Iterables

Arrays

All arrays are iterables, so any array can be used as an argument of a function that requires an iterable.

Iterators

Any object that implements the Iterator interface can be used as an argument of a function that requires an iterable.

An iterator contains a list of items and provides methods to loop through them. It keeps a pointer to one of the elements in the list. Each item in the list should have a key which can be used to find the item.

An iterator must have these methods:

- current() - Returns the element that the pointer is currently pointing to. It can be any data type
- key() Returns the key associated with the current element in the list. It can only be an integer, float, boolean or string
- next() Moves the pointer to the next element in the list
- rewind() Moves the pointer to the first element in the list
- valid() If the internal pointer is not pointing to any element (for example, if next() was called at the end of the list), this should return false. It returns true in any other case

Example

Implement the Iterator interface and use it as an iterable:

```
<?php
// Create an Iterator
class MyIterator implements Iterator {
  private $items = [];
  private $pointer = 0;
  public function __construct($items) {
    // array_values() makes sure that the keys are numbers
```

```
    $this->items = array_values($items);
  }

  public function current() {
    return $this->items[$this->pointer];
  }

  public function key() {
    return $this->pointer;
  }

  public function next() {
    $this->pointer++;
  }
  public function rewind() {
    $this->pointer = 0;
  }
  public function valid() {
    // count() indicates how many items are in the list
    return $this->pointer < count($this->items);
  }
}
// A function that uses iterables
function printIterable(iterable $myIterable) {
  foreach($myIterable as $item) {
    echo $item;
  }
}
// Use the iterator as an iterable
$iterator = new MyIterator(["a", "b", "c"]);
printIterable($iterator);
```

```
?>
```