PHP PROGRAMMING

UNIT-1

Introduction

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.

1.PHP was created by Rasmus Lerdorf in 1994 but appeared in the market in 1995.

2.PHP stands for Hypertext Preprocessor.

3.PHP is an interpreted language, i.e., there is no need for compilation.

4.PHP is faster than other scripting languages, for example, ASP and JSP.

5.PHP is a server-side scripting language, which is used to manage the dynamic content of the website.

6.PHP can be embedded into HTML.

7.PHP is an object-oriented language.

8.PHP is an open-source scripting language.

9.PHP is simple and easy to learn language.

PHP Features

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

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

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

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

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

6.Database Support: PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.

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

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

9.Web servers Support:PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.

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

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

"Server side Scripting Vs Client Side Scripting"

"***Server side Scripting*** "

1.It helps work with the back end.

2.It doesn't depend on the client.

3.It runs on the web server.

4.It helps provide a response to every request that comes in from the user/client.

5.This is not visible to the client side of the application.

6.It requires the interaction with the server for the data to be process.

7.Server side scripting requires languages such as PHP, ASP.net, ColdFusion, Python, Ruby on Rails.

8.It is considered to be a secure way of working with applications.

9.It can be used to customize web pages.

10.It can also be used to provide dynamic websites.

"Client-side Scripting"

1.It helps work with the front end.

2.It is visible to the users.

3.The scripts are run on the client browser.

4.It runs on the user/client's computer.

5.It depends on the browser's version.

6.It doesn't interact with the server to process data.

7.Client side scripting involves languages such as HTML, CSS, JavaScript.

8.It helps reduce the load on the server.

9.It is considered to be less secure in comparison to client side scripting.

"PHP Variables"

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

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

After declaring a variable, it can be reused throughout the code.

Assignment Operator (=) is used to assign the value to a variable.

Syntax of declaring a variable in PHP is given below:

$variablename=value;

Rules for declaring PHP variable:

1.A variable must start with a dollar ($) sign, followed by the variable name.

2.It can only contain alpha-numeric character and underscore (A-z, 0-9, _).

3.A variable name must start with a letter or underscore (_) character.

4.A PHP variable name cannot contain spaces.

5.One thing to be kept in mind that the variable name cannot start with a number or special symbols.

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

"PHP Variable Scope"

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

PHP has three types of variable scopes:

1.Local variable

2.Global variable

3.Static variable

1."Local variable"

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

A variable declaration outside the function with the same name is completely different from the variable declared inside the function.

For Example:-

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


output:

Local variable declared inside the function is: 45

2.Global variable

The global variables are the variables that are declared outside the function. These variables can be accessed anywhere in the program. To access the global variable within a function, use the GLOBAL keyword before the variable. However, these variables can be directly accessed or used outside the

function without any keyword. Therefore there is no need to use any keyword to access a global variable outside the function.

For example:

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


Output:

Variable inside the function: smriddhi

Variable outside the function: smridhhi

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

3.Static variable

It is a feature of PHP to delete the variable, once it completes its execution and memory is freed. Sometimes we need to store a variable even after completion of function execution. Therefore, another important feature of variable scoping is static variable. We use the static keyword before the variable to define a variable, and this variable is called as static variable.

Static variables exist only in a local function, but it does not free its memory after the program execution leaves the scope.

Example:

File: static_variable.php

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

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


Output:

Static: 4

Non-static: 7

Static: 5

Non-static: 7

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.

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:

Using define() function

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.

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.

example:-

```
<?php

define("MESSAGE","Hello  PHP");

echo MESSAGE;

?>
```

output:

Hello PHP

Create a constant with case-insensitive name:

```
?php

define("MESSAGE","Hello PHP",true);//not case sensitive

echo MESSAGE, "</br>";

echo message;

?>
```

output:-

Hello PHP

Hello PHP

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.

example:

```
<?php

const MESSAGE="Hello const  PHP";

echo MESSAGE;

?>
```

output:-

Hello const 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)

example:-

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

output:-

mohit jangir

mohit jangir

"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: Special Types

There are 2 special data types in PHP.

1.resource

2.NULL

PHP Boolean

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

Example:

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

Output:

This condition is TRUE.

PHP Integer

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

Rules for integer:

1.An integer can be either positive or negative.

2.An integer must not contain decimal point.

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

4.The range of an integer must be lie between 2,147,483,648 and 2,147,483,647 i.e., -2^31 to 2^31.

Example:

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

```
    echo "Octal number: " .$oct1. "</br>";

    echo "HexaDecimal number: " .$hexa1. "</br>";

?>
```

Output:

Decimal number: 34

Octal number: 163

HexaDecimal number: 69

PHP Float

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

Example:

```
<?php

    $n1 = 19.34;

    $n2 = 54.472;

    $sum = $n1 + $n2;

    echo "Addition of floating numbers: " .$sum;

?>
```


Output:

Addition of floating numbers: 73.812

PHP String

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

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

Example:

```
<?php

    $company = "Javatpoint";

    //both single and double quote statements will treat different

    echo "Hello $company";
```

```
    echo "</br>";
    echo 'Hello $company';
?>
```

Output:

Hello Javatpoint

Hello $company

PHP Array

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

Example:

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


Output:

array(3) { [0]=> string(13) "Royal Enfield" [1]=> string(6) "Yamaha" [2]=> string(3) "KTM" }

Array Element1: Royal Enfield

Array Element2: Yamaha

Array Element3: KTM

PHP object

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

Example:

```
<?php
    class bike {
```

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

Output:

Bike Model: Royal Enfield

PHP Resource

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

PHP Null

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

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

Example:

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

Output:

PHP Operators

PHP Operator is a symbol i.e used to perform operations on operands. In simple words, operators are used to perform operations on variables or values. For example:

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

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

PHP Operators can be categorized in following forms:

1.Arithmetic Operators

2.Assignment Operators

3.Bitwise Operators

4.Comparison Operators

5.Incrementing/Decrementing Operators

6.Logical Operators

7.String Operators

8.Array Operators

9.Type Operators

10.Execution Operators

11.Error Control Operators

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

Unary Operators: works on single operands such as ++, -- etc.

Binary Operators: works on two operands such as binary +, -, *, / etc.

Ternary Operators: works on three operands such as "?:".

1.Arithmetic Operators

Arithmetic Operators

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

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

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

2.Assignment Operators

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

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

3.Bitwise Operators

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

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

4.Comparison Operators

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

Operator	Name	Example	Explanation
==	Equal	$a == $b	Return TRUE if $a is equal to $b
===	Identical	$a === $b	Return TRUE if $a is equal to $b, and they are of same data type
!==	Not identical	$a !== $b	Return TRUE if $a is not equal to $b, and they are not of same data type
!=	Not equal	$a != $b	Return TRUE if $a is not equal to $b
<>	Not equal	$a <> $b	Return TRUE if $a is not equal to $b
<	Less than	$a < $b	Return TRUE if $a is less than $b
>	Greater than	$a > $b	Return TRUE if $a is greater than $b
<=	Less than or equal to	$a <= $b	Return TRUE if $a is less than or equal $b
>=	Greater than or equal to	$a >= $b	Return TRUE if $a is greater than or equal $b
<=>	Spaceship	$a <=>$b	Return -1 if $a is less than $b Return 0 if $a is equal $b Return 1 if $a is greater than $b

5.Incrementing/Decrementing Operators

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

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

6.Logical Operators

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

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

7.String Operators

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

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

8.Array Operators

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

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

9.Type Operators

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

10. Execution Operators

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

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

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

11.Error Control Operators

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

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

PHP Regular Expressions

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.

Advantage and uses of Regular Expression

Regular expression is used almost everywhere in current application programming. Below some advantages and uses of regular expressions are given:

1.Regular expression helps the programmers to validate text string.

2.It offers a powerful tool to analyze and search a pattern as well as to modify the text string.

3.By using regexes functions, simple and easy solutions are provided to identify the patterns.

4.Regexes are helpful for creating the HTML template system recognizing tags.

5.Regexes are widely used for browser detection, form validation, spam filtration, and password strength checking.

6.It is helpful in user input validation testing like email address, mobile number, and IP address.

7.It helps in highlighting the special keywords in file based upon the search result or input.

8.Metacharacters allow us to create more complex patterns.PHP

Conditional Statements

Very often when you write code, you want to perform different actions for different conditions.

You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

• if statement - executes some code if one condition is true

• if...else statement - executes some code if a condition is true and another code if that

condition is false

• if...elseif....else statement - executes different codes for more than two conditions

• switch statement - selects one of many blocks of code to be executed

PHP - The if Statement

The if statement executes some code if one condition is true.

Syntax

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

The example below will output "Have a good day!" if the current time (HOUR) is less than 20:

Example

```
<?php
$t = date("H");
if ($t < "20") {
 echo "Have a good day!";
}
?>
```


PHP - The if...else Statement

The if....else statement executes some code if a condition is true and another code if that condition is false.

Syntax

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

The example below will output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

Example

```
<?php
$t = date("H");
```

```
if ($t < "20") {
echo "Have a good day!";
} else {
 echo "Have a good night!";
}
?>
```

PHP - The if...elseif....else Statement

The if....elseif...else statement executes different codes for more than two conditions.

Syntax

```
if (condition) {
 code to be executed if this condition is true;
} elseif (condition) {
 code to be executed if this condition is true;
} else {
 code to be executed if all conditions are false;
}
```


The example below will output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

Example

```
<?php
$t = date("H");
if ($t < "10") {
 echo "Have a good morning!";
} elseif ($t < "20") {
 echo "Have a good day!";
} else {
 echo "Have a good night!";
```

```
}
?>
```

The Switch Statement

If you want to select one of many blocks of code to be executed, use the Switch statement.

The switch statement is used to avoid long blocks of if..elseif..else code.

Syntax

```
switch (expression){
 case label1:
 code to be executed if expression = label1;
 break;

 case label2:
 code to be executed if expression = label2;
 break;
 default:

 code to be executed
 if expression is different
 from both label1 and label2;
}
```


Example

```
<?php
$favcolor = "red";
switch ($favcolor) {
 case "red":
 echo "Your favorite color is red!";
```

```
 break;
case "blue":
 echo "Your favorite color is blue!";
 break;
case "green":
 echo "Your favorite color is green!";
 break;
default:
 echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

" Loop"

Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.

• for – loops through a block of code a specified number of times.

• while – loops through a block of code if and as long as a specified condition is true.

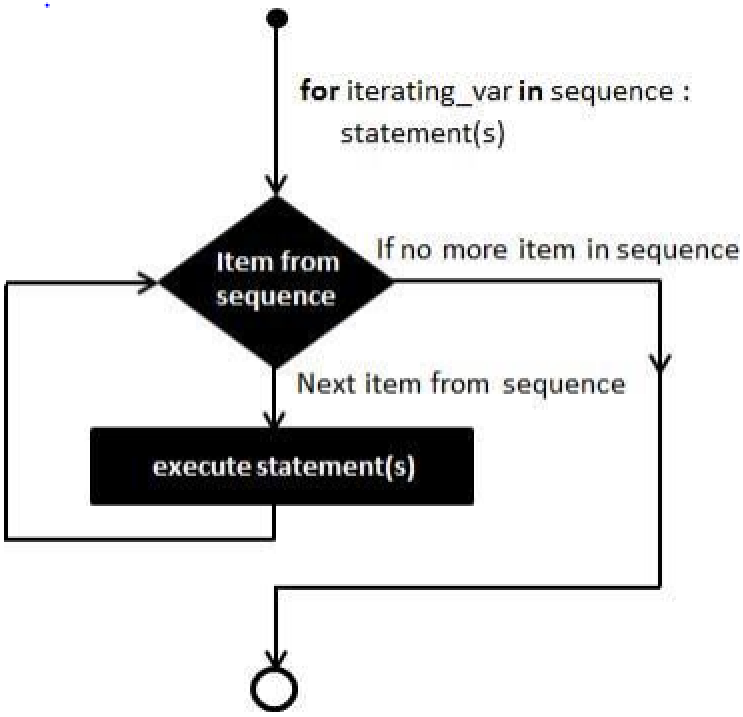
• do...while – loops through a block of code once, and then repeats the loop as long as a special condition is true.

• foreach – loops through a block of code for each element in an array.

We will discuss about continue and break keywords used to control the loops execution.

The for loop statement

The for statement is used when you know how many times you want to execute a statement or a block of statements.



Syntax

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

The initializer is used to set the start value for the counter of the number of loop iterations. A variable may be declared here for this purpose and it is traditional to name it $i.

Example

The following example makes five iterations and changes the assigned value of two variables on each pass of the loop −

Live Demo

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

```
<?php
$a = 0;
$b = 0;

for( $i = 0; $i<5; $i++ ) {
$a += 10;
$b += 5;
}

echo ("At the end of the loop a = $a and b = $b" );
?>

</body>
</html>
```

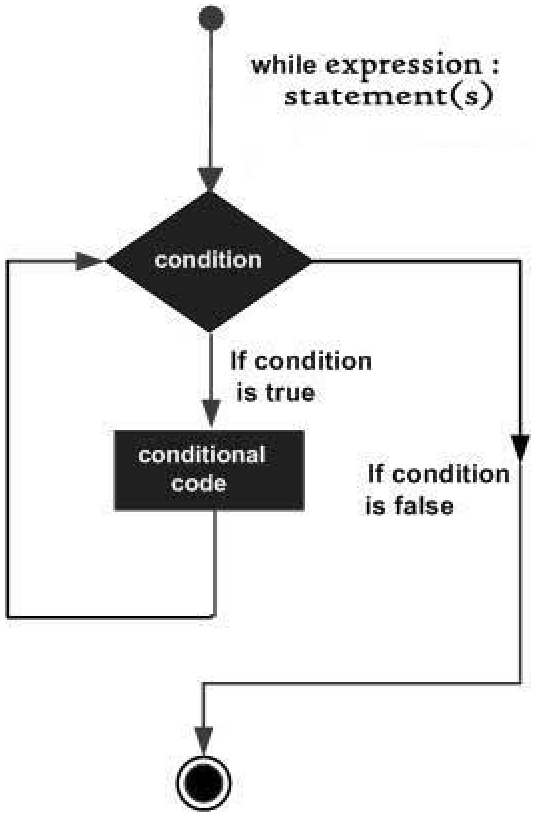

This will produce the following result −

At the end of the loop a = 50 and b = 25

The while loop statement

The while statement will execute a block of code if and as long as a test expression is true.

If the test expression is true then the code block will be executed. After the code has executed the test expression will again be evaluated and the loop will continue until the test expression is found to be false.



Syntax

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

Example

This example decrements a variable value on each iteration of the loop and the counter increments until it reaches 10 when the evaluation is false and the loop ends.

Live Demo

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

```
<?php
$i = 0;
$num = 50;

while( $i < 10) {
$num--;
$i++;
}

echo ("Loop stopped at i = $i and num = $num" );
?>

</body>
</html>
```


This will produce the following result −

```
Loop stopped at i = 10 and num = 40
```

The do...while loop statement

The do...while statement will execute a block of code at least once - it then will repeat the loop as long as a condition is true.

Syntax

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

Example

The following example will increment the value of i at least once, and it will continue

incrementing the variable i as long as it has a value of less than 10 −

Live Demo

```
<html>
 <body>

 <?php
 $i = 0;
 $num = 0;

 do {
 $i++;
 }

 while( $i < 10 );
 echo ("Loop stopped at i = $i" );
 ?>

 </body>
</html>
```


This will produce the following result —

Loop stopped at i = 10

The foreach loop statement

The foreach statement is used to loop through arrays. For each pass the value of the current array element is assigned to $value and the array pointer is moved by one and in the next pass next element will be processed.

Syntax

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

Example

Try out following example to list out the values of an array.

Live Demo

```
<html>
 <body>

 <?php
 $array = array( 1, 2, 3, 4, 5);

 foreach( $array 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

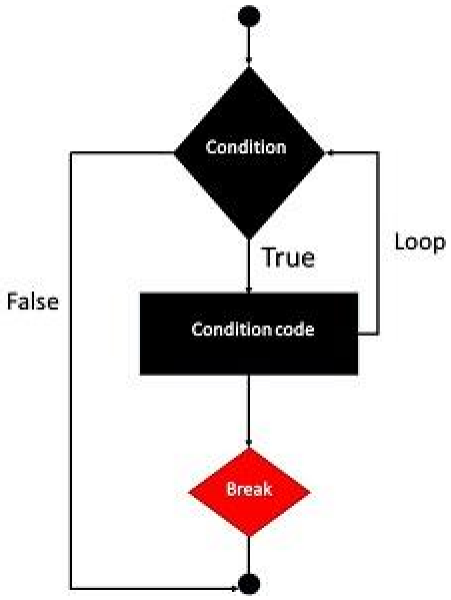
Jumping Statements

1.The break statement

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

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

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



Example

In the following example condition test becomes true when the counter value reaches 3 and loop terminates.

Live Demo

```
<html>
 <body>

 <?php
 $i = 0;

 while( $i < 10) {
 $i++;
 if( $i == 3 )break;
```

```
}
echo ("Loop stopped at i = $i" );
?>

</body>
</html>
```

This will produce the following result —

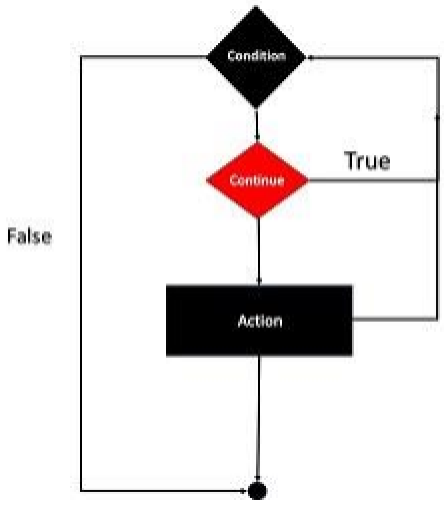
Loop stopped at i = 3

The continue statement

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

the continue statement is used within looping and switch control structure when you immediately jump to the next iteration.

The continue statement can be used with all types of loops such as - for, while, do-while, and foreach loop. The continue statement allows the user to skip the execution of the code for the specified condition.



Example

In the following example loop prints the value of array but for which condition becomes true it

just skip the code and next value is printed.

Live Demo

```
<html>
 <body>

 <?php
 $array = array( 1, 2, 3, 4, 5);

 foreach( $array as $value ) {
 if( $value == 3 )continue;
 echo "Value is $value <br />";
 }
 ?>

 </body>
</html>
```


This will produce the following result −

```
Value is 1
Value is 2
Value is 4
Value is 5
```

PHP Arrays

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

Advantage of PHP Array

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

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

3.Sorting: We can sort the elements of array.

PHP Array Types

There are 3 types of array in PHP.

1.Indexed Array(also called numeric array)

2.Associative Array

3.Multidimensional Array

PHP Indexed Array

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

or

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.

Live Demo

```
<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
```


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

Live Demo

```
<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
```

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.

Live Demo

```
<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