Constants are like variables except that once they are defined they cannot be changed or undefined.Unit V:

PHP INTRODUCTION :

PHP is a recursive acronym for "PHP: Hypertext Preprocessor".

PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible.

Simplicity

Efficiency

Security

Flexibility

Familiarity

HOW WEB WORKS :

Web physically consists of the following components −

Your personal computer – This is the PC at which you sit to see the web.

A Web browser − A software installed on your PC which helps you to browse the Web.

An internet connection − This is provided by an ISP and connects you to the internet to reach to any Website.

A Web server − This is the computer on which a website is hosted.

Routers & Switches − They are the combination of software and hardware who take your request and pass to appropriate Web server.

The Web is known as **a client-server system.**

SETTINGS UP AN ENVIRONMENT (LAMP SERVER) :

In order to develop and run PHP Web pages three vital components need to be installed on your computer system.

Web Server − PHP will work with virtually all Web Server software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server. Download Apache for free here − https://httpd.apache.org/download.cgi

Database − PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database. Download MySQL for free here − https://www.mysql.com/downloads/

PHP Parser − In order to process PHP script instructions a parser must be installed to generate HTML output that can be sent to the Web Browser. This tutorial will guide you how to install PHP parser on your computer.

PHP Parser Installation

Before you proceed it is important to make sure that you have proper environment setup on your machine to develop your web programs using PHP.

Type the following address into your browser's address box.

http://127.0.0.1/info.php

If this displays a page showing your PHP installation related information then it means you have PHP and Webserver installed properly.

PHP VARIABLES :

Variables are "containers" for storing information.

Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

```
<?php

$txt = "Hello world!";

$x = 5;

$y = 10.5;

?>
```

Rules for PHP variables:

A variable starts with the $ 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 can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)

Variable names are case-sensitive ($age and $AGE are two different variables).

PHP CONSTANTS :

Constants are like variables except that once they are defined they cannot be changed or undefined.

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no $ sign before the constant name).

Note: Unlike variables, constants are automatically global across the entire script

Create a PHP Constant

To create a constant, use the define() function.

Syntax

define(name, value, case-insensitive)

Parameters:

name: Specifies the name of the constant

value: Specifies the value of the constant

case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false.

Example :

```
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

PHP strings:

A string is a sequence of characters, like "Hello world!".

PHP String Functions::

strlen() - Return the Length of a String

The PHP strlen() function returns the length of a string.

Example

Return the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

str_word_count() - Count Words in a String

The PHP str_word_count() function counts the number of words in a string.

Example

Count the number of word in the string "Hello world!":

```
<?php

echo str_word_count("Hello world!"); // outputs 2

?>
```

strrev() - Reverse a String

The PHP strrev() function reverses a string.

Example

Reverse the string "Hello world!":

```
<?php

echo strrev("Hello world!"); // outputs !dlrow olleH

?>.
```

strpos() - Search For a Text Within a String

The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

Example

Search for the text "world" in the string "Hello world!":

```
<?php

echo strpos("Hello world!", "world"); // outputs 6

?>
```

str_replace() - Replace Text Within a String

The PHP str_replace() function replaces some characters with some other characters in a string.

Example

Replace the text "world" with "Dolly":

```
<?php
```

```
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!

?>
```

PHP Arrays:

An array is a special variable, which can hold more than one value at a time.

Example

```
<?php

$cars = array("Volvo", "BMW", "Toyota");

echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";

?>
```

Create an Array in PHP

In PHP, the array() function is used to create an array:

```
array();
```

In PHP, there are three types of arrays:

Indexed arrays - Arrays with a numeric index

Associative arrays - Arrays with named keys

Multidimensional arrays - Arrays containing one or more arrays

PHP OPERATORS :

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

Arithmetic operators

Assignment operators

Comparison operators

Increment/Decrement operators

Logical operators

String operators

Array operators

Conditional assignment operators

P**HP Arithmetic Operators**

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

Operator	Name	Example	
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y.	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiatio	$x ** $y	Result of raising $x to the $y'th power

PHP Operators

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

Arithmetic operators

Assignment operators

Comparison operators

Increment/Decrement operators

Logical operators

String operators

Array operators

Conditional assignment operators

PHP Arithmetic Operators

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

Operator	Name	Example	Result	Show it
+	Addition	$x + $y	Sum of $x and $y	
-	Subtraction	$x - $y	Difference of $x and $y	
*	Multiplication	$x * $y	Product of $x and $y	
/	Division	$x / $y	Quotient of $x and $y	
%	Modulus	$x % $y	Remainder of $x divided by $y	
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power	

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result	Show it
==	Equal	$x == $y	Returns true if $x is equal to $y	
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type	
!=	Not equal	$x != $y	Returns true if $x is not equal to $y	

<> Not equal $x <> $y Returns true if $x is not equal to $y

!== Not identical $x !== $y Returns true if $x is not equal to $y, or they are not of the same type

> Greater than $x > $y Returns true if $x is greater than $y

< Less than $x < $y Returns true if $x is less than $y

>= Greater than or equal to $x >= $y Returns true if $x is greater than or equal to $y

<= Less than or equal to $x <= $y Returns true if $x is less than or equal to $y

<=> Spaceship $x <=> $y Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Operators

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

Arithmetic operators

Assignment operators

Comparison operators

Increment/Decrement operators

Logical operators

String operators

Array operators

Conditional assignment operators

PHP Arithmetic Operators

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

Operator	Name	Example	Result	Show it
+	Addition	$x + $y	Sum of $x and $y	
-	Subtraction	$x - $y	Difference of $x and $y	
*	Multiplication	$x * $y	Product of $x and $y	
/	Division	$x / $y	Quotient of $x and $y	
%	Modulus	$x % $y	Remainder of $x divided by $y	
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power	

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description	Show it
x = y	x = y	The left operand gets set to the value of the expression on the right	
x += y	x = x + y	Addition	
x -= y	x = x - y	Subtraction	
x *= y	x = x * y	Multiplication	
x /= y	x = x / y	Division	
x %= y	x = x % y	Modulus	

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result	Show
==	Equal	$x == $y	Returns true if $x is equal to $y	
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type	

!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true

|| Or $x || $yTrue if either $x or $y is true

! Not !$x True if $x is not true

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator Name Example Result Show it

. Concatenation $txt1 . $txt2 Concatenation of $txt1 and $txt2

.= Concatenation assignment $txt1 .= $txt2 Appends $txt2 to $txt1

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator Name Example Result Show it

+ Union $x + $y Union of $x and $y

== Equality$x == $y Returns true if $x and $y have the same key/value pairs

=== Identity $x === $y Returns true if $x and $y have the same key/value pairs in the same order and of the same types

!= Inequality $x != $y Returns true if $x is not equal to $y

<> Inequality $x <> $y Returns true if $x is not equal to $y

!== Non-identity $x !== $y Returns true if $x is not identical to $y

PHP Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:

Operator Name Example Result Show it

?: Ternary $x = expr1 ? expr2 : expr3 Returns the value of $x.

The value of $x is expr2 if expr1 = TRUE.

The value of $x is expr3 if expr1 = FALSE

?? Null coalescing $x = expr1 ?? expr2 Returns the value of $x.

The value of $x is expr1 if expr1 exists, and is not NULL.

If expr1 does not exist, or is NULL, the value of $x is expr2.

Introduced in PHP 7

PHP control structures :

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;
}
```

Example

Output "Have a good day!" if the current time (HOUR) is less than 20:

```
<?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;
}
```

Example

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```
<?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 first condition is false and this condition is true;
} else {
```

```
  code to be executed if all conditions are false;
}
```

Example

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!":

```
<?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 PHP switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
```

```
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

PHP switch Statement

The switch statement is used to perform different actions based on different conditions.

The PHP switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

This is how it works: First we have a single expression n (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically. The default statement is used if no match is found.

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!";
}
?>
```

PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

while - loops through a block of code as long as the specified condition is true

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

for - loops through a block of code a specified number of times

foreach - loops through a block of code for each element in an array

The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

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

Examples

The example below displays the numbers from 1 to 5:

Example

```
<?php
$x = 1;
while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
  code to be executed;
```

} while (condition is true);

Examples

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php

$x = 1;

do {

  echo "The number is: $x <br>";

  $x++;

} while ($x <= 5);

?>
```

The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init counter; test counter; increment counter) {

  code to be executed for each iteration;

}
```

Parameters:

init counter: Initialize the loop counter value

test counter: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.

increment counter: Increases the loop counter value

Examples

The example below displays the numbers from 0 to 10:

Example

```
<?php

for ($x = 0; $x <= 10; $x++) {
```

```
  echo "The number is: $x <br>";
}
?>
```

The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

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

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

Examples

The following example will output the values of the given array ($colors):

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

PHP Functions :

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

PHP Built-in Functions

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

A function is a block of statements that can be used repeatedly in a program.

A function will not execute automatically when a page loads.

A function will be executed by a call to the function.

Create a User Defined Function in PHP

A user-defined function declaration starts with the word function:

Syntax

```
function functionName() {
  code to be executed;
}
```

Note: A function name must start with a letter or an underscore. Function names are NOT case-sensitive.

Tip: Give the function a name that reflects what the function does!

In the example below, we create a function named "writeMsg()". The opening curly brace ({) indicates the beginning of the function code, and the closing curly brace (}) indicates the end of the function. The function outputs "Hello world!". To call the function, just write its name followed by brackets ():

Example

```
<?php
function writeMsg() {
  echo "Hello world!";
}
writeMsg(); // call the function
?>
```

Reading data from Web pages :

PHP Form Handling

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

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

PHP Get Form

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

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

File: form1.html

<form action="welcome.php" method="get">

Name: <input type="text" name="name"/>

<input type="submit" value="visit"/>

</form>

File: welcome.php

<?php

$name=$_GET["name"];//receiving name field value in $name variable

echo "Welcome, $name";

?>

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

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

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

File: form1.html

<form action="login.php" method="post"> Let’s have a look at a very simple example, which displays a message using PHP code. Create the index.php file with the following contents under your document root.

1

2

3

4

5

6

7

8

9

```
<!DOCTYPE html>
<html>
<head>
<title>How to put PHP in HTML - Simple Example</title>
</head>
<body>
<h1><?php echo "This message is from server side." ?></h1>
</body>
</html>
<table>
<tr><td>Name:</td><td> <input type="text" name="name"/></td></tr>
<tr><td>Password:</td><td> <input type="password" name="password"/></td></tr>
<tr><td colspan="2"><input type="submit" value="login"/>  </td></tr>
</table>
</form>
```

File: login.php

```
<?php
$name=$_POST["name"];//receiving name field value in $name variable
$password=$_POST["password"];//receiving password field value in $password variable

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

Embedding php with in html :

Let’s have a look at a very simple example, which displays a message using PHP code. Create the index.php file with the following contents under your document root.

```
<!DOCTYPE html>

<html>

<head>

<title>How to put PHP in HTML - Simple Example</title>

</head>

<body>

<h1><?php echo "This message is from server side." ?></h1>

</body>

</html>
```

Connectivity with MySQL databases :

PHP 5 and later can work with a MySQL database using:

MySQLi extension (the "i" stands for improved)

PDO (PHP Data Objects)

MySQL Examples in Both MySQLi and PDO Syntax

In this, and in the following chapters we demonstrate three ways of working with PHP and MySQL:

MySQLi (object-oriented)

MySQLi (procedural)

PDO

Open a Connection to MySQL

Before we can access data in the MySQL database, we need to be able to connect to the server:

Example (MySQLi Object-Oriented)

```
<?php

$servername = "localhost";

$username = "username";

$password = "password";
```

```
// Create connection
$conn = new mysqli($servername, $username, $password);
// Check connection
if ($conn->connect_error) {
  die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```

Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
// Create connection
$conn = mysqli_connect($servername, $username, $password);
// Check connection
if (!$conn) {
  die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```

Example (PDO)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
try {
```

```
  $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);
  // set the PDO error mode to exception
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  echo "Connected successfully";
} catch(PDOException $e) {
  echo "Connection failed: " . $e->getMessage();
}
?>
```