Introduction to PHP

Uday M
Assistant Professor
Department of Computer Science
Seshadripuram Degree College

History

- 1994 – Created by Rasmus Lerdorf as Personal Home Page (PHP) Tools to track website visitors
- 1995 – Released as PHP/FI (Forms Interpreter) with form handling and database support
- 1998 – PHP 3 launched by Zeev Suraski & Andi Gutmans; renamed PHP: Hypertext Preprocessor
- 2000 – PHP 4 introduced with Zend Engine for better performance
- 2004 – PHP 5 released with full Object-Oriented Programming, exceptions, PDO
- PHP 6 – Skipped due to Unicode issues
- 2015 – PHP 7 released; major performance improvement (2× faster)
- 2020 – PHP 8 introduced JIT, union types, match expression
- Today – PHP powers major platforms like WordPress, Facebook, Wikipedia

Year	Version	Major Contribution
1994	PHP Tools	Visitor tracking scripts
1995	PHP/FI	Form handling
1998	PHP 3	Official PHP language
2000	PHP 4	Zend Engine
2004	PHP 5	Full OOP support
2015	PHP 7	High performance
2020	PHP 8	JIT, modern features

Client-side Scripting

Client-side scripting refers to programs that are executed on the user’s web browser rather than on the web server.

Common Client-Side Scripting Languages

- JavaScript (most popular)
- VBScript (obsolete)

Server-side Scripting

A script is a set of instructions executed by another program

Web Servers:

- Apache
- IIS (Microsoft)

Server-side technologies:

- PHP
- ASP
- ColdFusion

Introduction

PHP (Hypertext Preprocessor) is a server-side scripting language mainly used for web development. It is embedded within HTML and executed on the server, generating dynamic web pages.

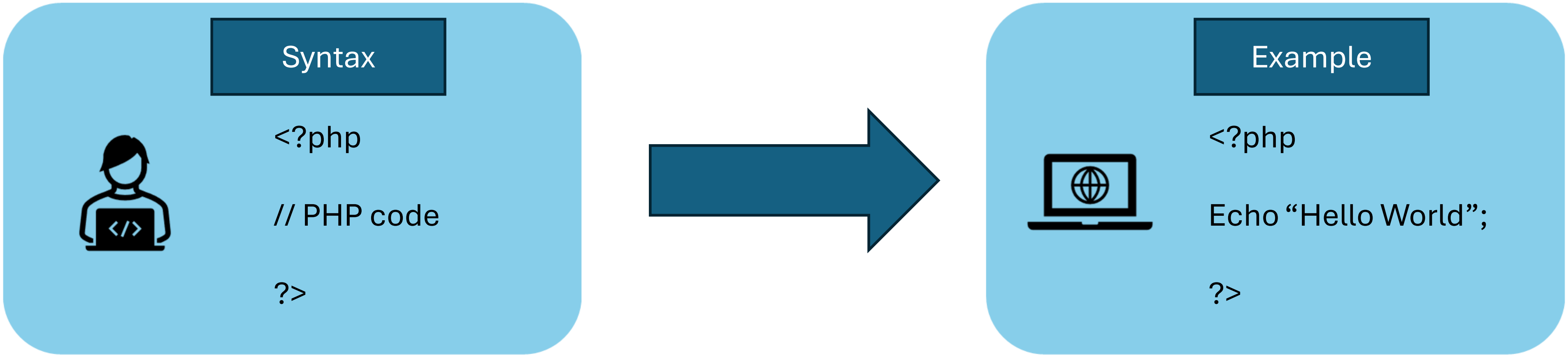
Advantages:

1. Websites Function Faster
2. Creates Dynamic Web Page
3. Provides Security Encryption
4. Cost Efficient
5. Compatibility

Note: According to w3techs, 79.2% web is still powered by PHP.
Which means PHP is installed on approximately 20 million websites and one million web servers.

PHP Syntax

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



Syntax

```
<?php

// PHP code

?>
```

Example

```
<?php

Echo “Hello World”;

?>
```

What is a PHP File?

- A PHP file containing PHP code which is used to create dynamic web pages.
- It contains HTML, CSS, JavaScript and PHP code together.
- It Runs on a web server with PHP installed (e.g., Apache, Nginx) and the output is sent to the browser as plain HTML.
- It is saved with a .php extension.

What Can PHP Do?

- **Generate Dynamic Content:** PHP creates dynamic web pages based on user input.
- **Manage Files:** PHP can create, open, read, write, and delete files on the server.
- **Process Form Data:** PHP collects and processes data from web forms.
- **Handle Cookies:** PHP can send and receive cookies to manage sessions.
- **Interact with Databases:** PHP adds, deletes, and modifies data in databases.
- **Control User Access:** PHP can manage login systems and user permissions.
- **Encrypt Data:** PHP can encrypt sensitive information for security.

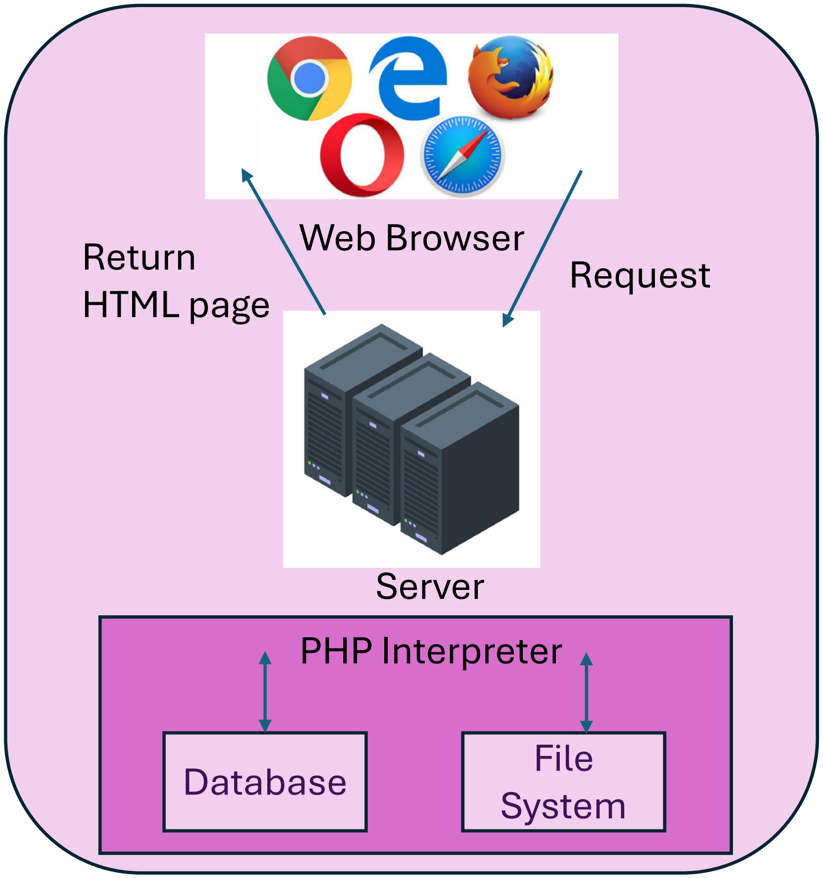
Features of PHP

- Simple
- Highly Flexible
- Platform Independent
- Interpreted
- Fast
- Open Source
- Secure
- Large PHP Community

Limitations of PHP

- **Security Risks:** If not used properly, PHP can be vulnerable to attacks like SQL injection, XSS, and CSRF. Developers must follow security best practices.
- **Not Ideal for Command-Line:** While PHP can be run from the command line, it's mainly designed for web development, so its performance for general-purpose tasks may not be as efficient as other languages.
- **Limited Support for Multi-threading:** While PHP can handle concurrent requests by asynchronous frameworks, multi-threaded support is limited compared to languages like Java or Node.js.

Working of PHP Pages



- The request is sent from browser to server in dot PHP extension
- the server sends the PHP page request to PHP interpreter
- PHP interpreter interprets the script to access specific data as requested
- The server then returns the HTML page according to the requirements

Major Compamies Using PHP

f
yahoo!
zoom
Magento®
An Adobe Company
Etsy

Installation & Configuration of PHP

SOFTWARES REQUIRED FOR TO WORK WITH PHP:

1. Browser - Internet Explorer
2. Server - Apache
3. Server-side script - PHP
4. Database - MYSQL

HOW TO DOWNLOAD & INSTALL XAMPP TOOL

1. Go to Google
2. XAMPP download in Google Search
3. Select appropriate version of XAMPP based on OS
4. Click on download

TOOLS TO WORK WITH PHP:

1. XAMPP - Cross Apache MYSQL PHP Perl
2. WAMPP - Window Apache MYSQL PHP Perl
3. LAMPP - Linux Apache MYSQL PHP Perl
4. MAMPP - Macintosh Apache MYSQL PHP Perl

Installation & Configuration of PHP

Installation

1. Double click on downloaded file
2. Click on the Next button
3. Select drive where you want to installed
4. Click on the Next button until you will get finished button
5. Click on Finish button

HOW TO START OR STOP APACHE SERVER?

1. Open XAMPP folder where we installed
2. Select Icon with name called XAMPP Hyper control
3. Double click on the Icon
4. It will open the window, click on start button related to Apache module.

HOW TO CHECK XAMPP S/W ARE SUCCESSUFULLY RUNNING OR NOT?

1. Open any browser
2. Type http://localhost in browser url address and press enter
3. If we can see welcome page XAMPP i.e., running successfully, so that we can start working with PHP

Installation & Configuration of PHP

How to write php program

1. Open ant Editor
2. Create a New file
3. Save the file name with extension .php inside "htdocs" folder C:/XAMPP/htdocs(Ex: abc.php)
4. We need to write PHP code inside saved file

   Ex: <?php

   ……..

   ?>
5. Save again php file

HOW TO EXECUTE PHP PROGRAM

1. Open any browser
2. Type in the url http://localhost/program.php name and press enter
3. You can see now output if we don't have any error

TYPES OF ERRORS

1. **Notice -** Notice that the error occurs when you try to access the undefined variable and then produce a notice error.

   Example:

   ```
   <?php
   $a=“Ram";
   echo "Notice Error !!";
   echo $b; ?>
   ```

TYPES OF ERRORS

2. **WARNING -** The main reason for warning errors are to include a missing file or using the incorrect number of parameters in a function.

Example:

```
<?php
echo "Warning Error!!";
include ("Welcome.php");
?>
```

TYPES OF ERRORS

3. **FATAL ERROR -** If you are trying to access the undefined functions, then the output is a fatal error.

Example:

```
<?php
function fun1() {
echo "Uday Kumar";
}
fun2();
echo "Fatal Error !!";
?>
```

TYPES OF ERRORS

4. **PARSE ERROR-** The parse error occurs if there is a syntax mistake in the script; the output is Parse errors.

   a) Unclosed quotes

   b) Missing or Extra parentheses

   c) Unclosed braces

   d) Missing semicolon

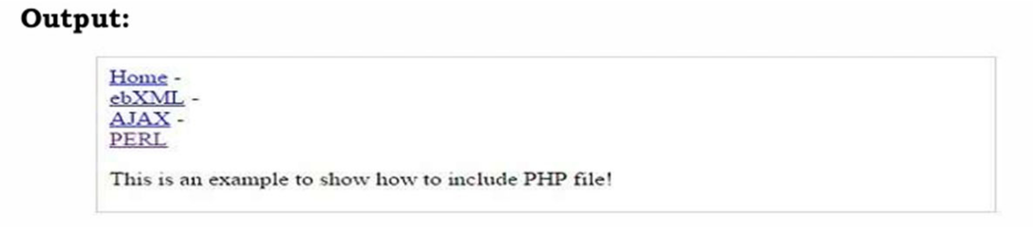
      Example:

```
<?php
echo "C#";
echo "PHP"
echo "C";
?>
```

INCLUDE MECHANISM IN PHP

- **include()** issues a warning (E_WARNING) if the file is not found, but the script will continue execution. This is useful for non-critical files like headers or footers, where the rest of the page can still be displayed if one section is missing.
- **require()** issues a fatal error (E_COMPILE_ERROR) and halts the script execution immediately if the file is not found. This is recommended for critical files, such as configuration files or database connections, that are essential for the application to function correctly.

```
<html>
<body>
<?php include("menu.php"); ?>
<p>This is an example to show how to
include PHP
file!</p>
</body>
</html>
```



Output:

Home -
ebXML -
AJAX -
PERL

This is an example to show how to include PHP file!

TAGS IN PHP

1. **Default syntax Or Universal Tags:** The default syntax starts with "<? php" and ends with "?>".
2. **Short open Tags:** The short tags starts with "<?" and ends with "?>". Short style tags are only available when they are enabled in php.ini configuration file on servers.
3. **HTML Script Tags:** HTML script tags look like this :

```
<script language="php">
echo "This is HTML script tags.";
</script>
```

4. **ASP Style Tags:** The ASP style tags start with "<%" and ends with "%>". ASP style tags are only available when they are enabled in php.ini configuration file on servers.

Comments in PHP

Comments are non-executable lines of text in the code that are ignored by the PHP interpreter.

1. **Single-Line Comments in PHP**

Single-line comments are used for brief explanations or notes that fit on a single line. The single line commments begin with either // or #.Using const keyword

2. **Multi-Line Comments**

Multi-line comments, also known as block comments, are used for longer explanations or when you need to comment out multiple lines of code. They begin with /* and end with */.

Sending Data to the Web Browser

GET Method: In the GET method, the data is sent as URL parameters that are usually strings of name and value pairs separated by ampersands (&). In general, a URL with GET data will look like this:

Example:

```
http://www.example.com/action.phpname=Sam&weight=55
```

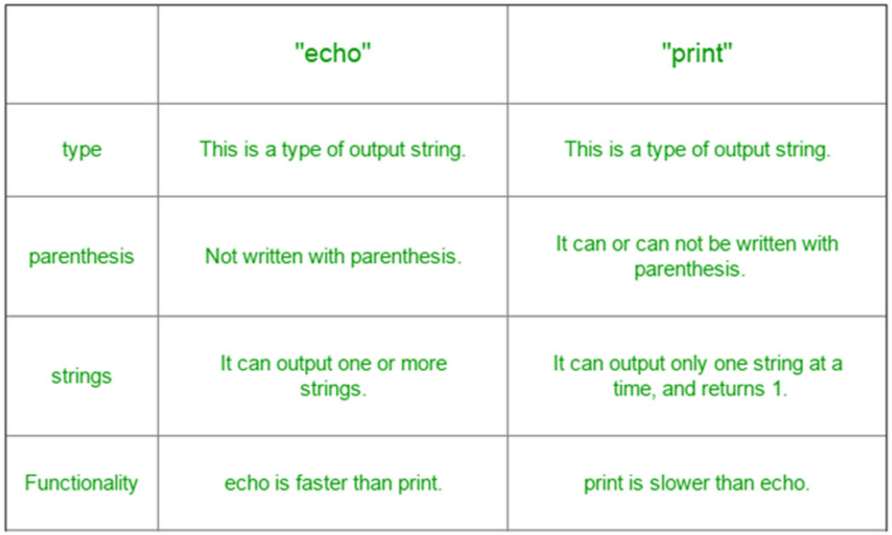
POST Method: In the POST method, the data is sent to the server as a package in a separate communication with the processing script. Data sent through the POST method will not be visible in the URL.

Example:

```
http://www.example.com/
```

OUTPUT FUNCTIONS IN PHP

1. **echo** for most general output tasks, such as displaying text, HTML, or simple variable values like strings and integers. It is slightly faster than print and the most common method for general display.
2. **print_r** primarily during development and debugging to inspect the contents and structure of complex variables like arrays and objects.



	"echo"	"print"
type	This is a type of output string.	This is a type of output string.
parenthesis	Not written with parenthesis.	It can or can not be written with parenthesis.
strings	It can output one or more strings.	It can output only one string at a time, and returns 1.
Functionality	echo is faster than print.	print is slower than echo.

VARIABLES

1. Variable is an identifier which holds data or another one variable and whose value can be changed at the execution time of script.
2. Syntax: $variablename=value;
3. A variable starts with the $ sign, followed by the name of the variable
4. A variable name must start with a letter or the underscore character
5. A variable name can't start with a number.
6. A variable name can only contain alpha-numeric characters and underscores(A-z, 0-9 and _)
7. Variable names are case-sensitive ($str and $STR both are two different)

TYPES OF VARIABLES

1. **Local variables:** The variables declared within a function are called local variables to that function and has its scope only in that particular function.

2. **Global variables:** The variables declared outside a function are called global variables. These variables can be accessed directly outside a function. To get access within a function we need to use the “global” keyword before the variable to refer to the global variable.

3. **Static variable:** It is the characteristic of PHP to delete the variable, ones it completes its execution and the memory is freed. To do this we use static keyword and the variables are then called as static variables.

CONSTANTS

Constants are name or identifier that can't be changed during the execution of the script.

1. Using define() function
2. Using const keyword

```
<?php

            define(“MSG”, “Hello World!”);
            echo msg;
?>
```

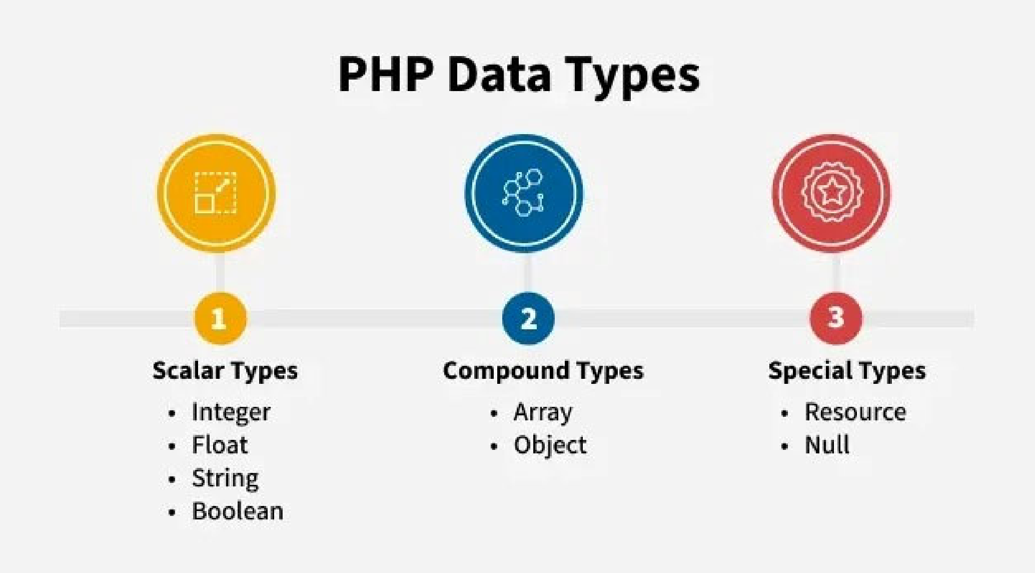
```
<?php

            const MSG="Hello world!";
            echo msg;
?>
```

Datatypes

A data type defines the kind of value a variable can hold, how it's stored in memory, and the operations (math, logic) that can be performed on it

Types of Datatypes:

1. Scalar Types
2. Compound Types
3. Special Types



Scaler Data Types

Scalar types hold single values. These are the basic and commonly used data types in PHP. They store only one value at a time, such as a number or a string of text.

1. **Integer**

An integer is a whole number (positive or negative) without decimals.

2. **Float (Double)**

A float is a number with a decimal point or in exponential form.

3. **String**

A string is a sequence of characters, enclosed in single or double quotes.

4. **Boolean**

A boolean represents either true or false. It is useful for conditional statements and logic control.

Compound Data Types

1. Array

Compound types hold multiple values or are used to structure data more meaningfully. These data types allow you to group several values together.

Example:

```
<?php
$colors = ["red", "green", "blue"];
echo $colors[1];  // Output: green
?>
```

2. Objects

An object is an instance of a class and is used to access methods and properties. Objects are explicitly declared and created from the new keyword.

Example:

```
<?php
class Car {
    public $brand = "Toyota";
}
$car = new Car();
echo $car->brand;  // Output: Toyota
?>
```

Special Data Types

1. NULL

A NULL value means the variable has no value. It is unset using unset().

Example:

```
<?php
$x = NULL;
var_dump($x);  // Output: NULL
?>
```

2. Resources

A resource is a special variable that holds a reference to an external resource, such as a file or database connection.

Example:

```
<?php
   $handle = fopen("file.txt", "r");
var_dump($handle);
?>
```

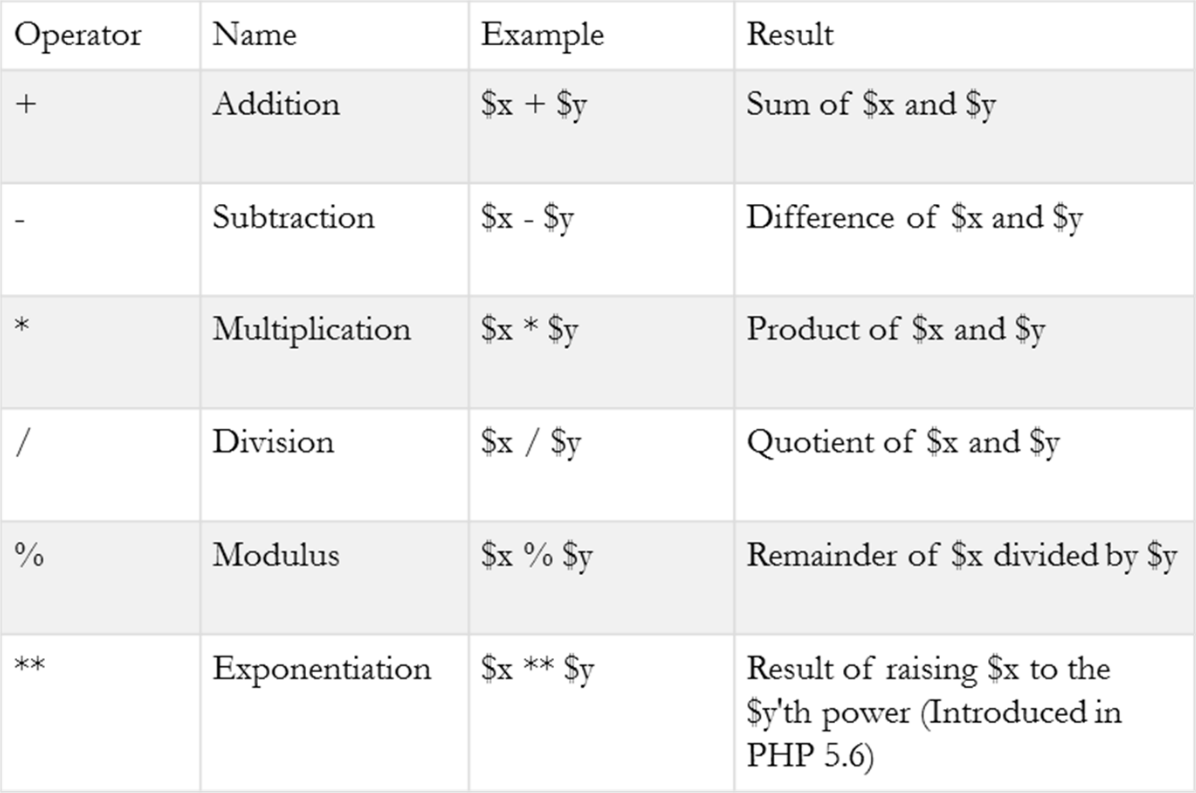
Operators

Operators are special symbols used to perform operations on variables and values. Operators help you perform a variety of tasks, such as mathematical calculations, string manipulations, logical comparisons, and more.

1. Arithmetic operators.
2. Assignment operators.
3. Comparison operators.
4. Increment/Decrement operators.
5. Logical operators.
6. String operators.
7. Array operators.

Arithmetic Operators

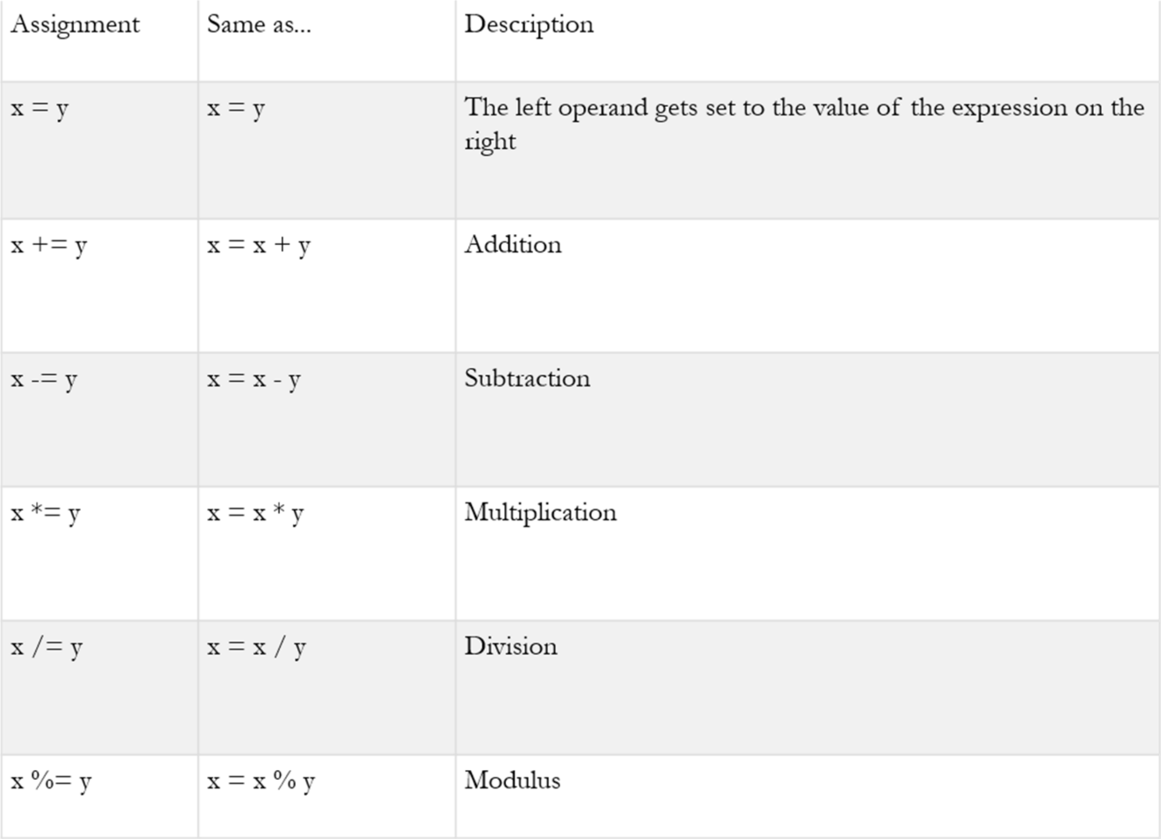
Arithmetic operators are used to perform basic arithmetic operations like addition, subtraction, multiplication, division, and modulus.



Operator	Name	Example	Result
+	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 (Introduced in PHP 5.6)

Assignment Operators

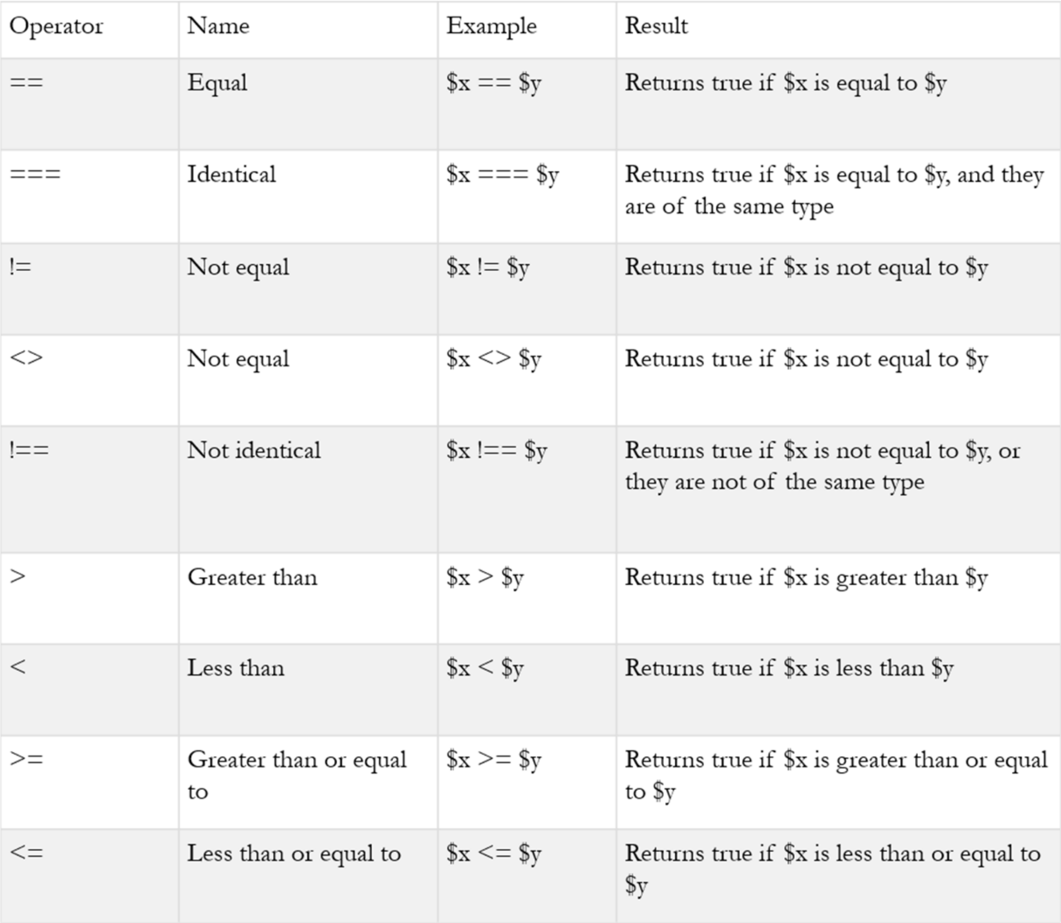
Assignment operators are used to assign values to variables. These operators allow you to assign a value and perform operations in a single step.



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

Comparison Operators

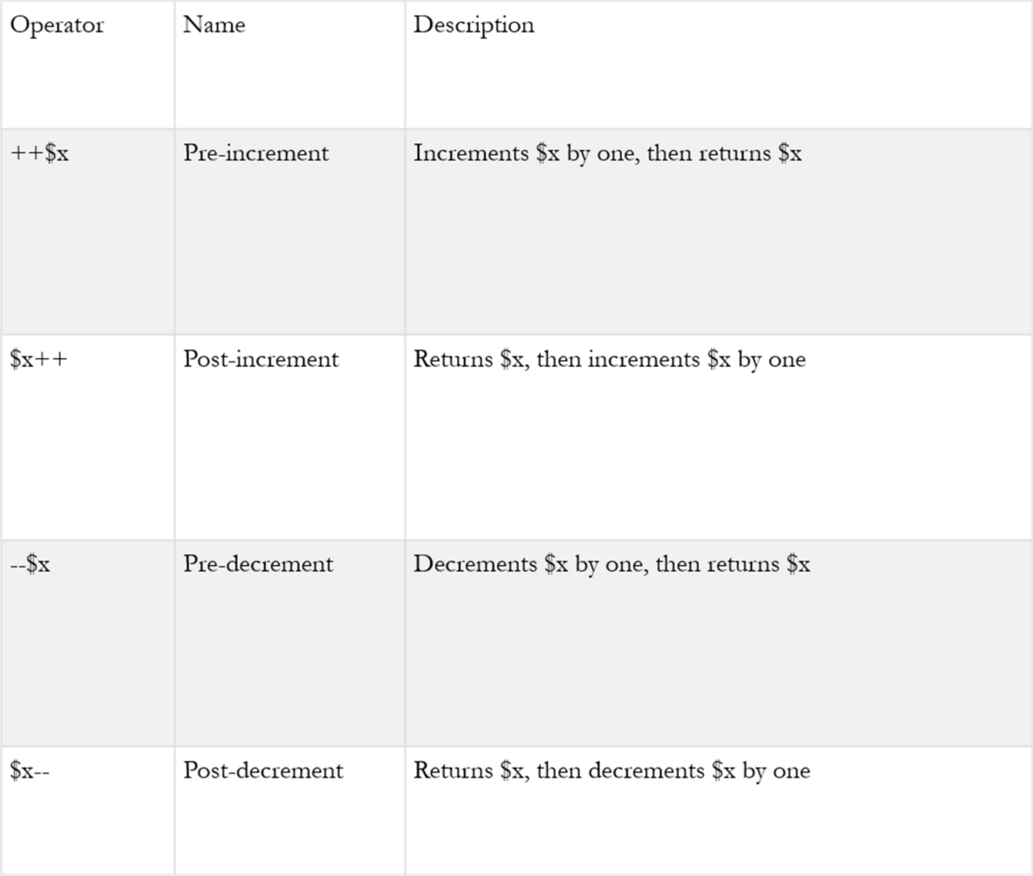
Comparison operators are used to compare two values and return a Boolean result (true or false).



Operator	Name	Example	Result
==	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

Increment/Decrement Operators

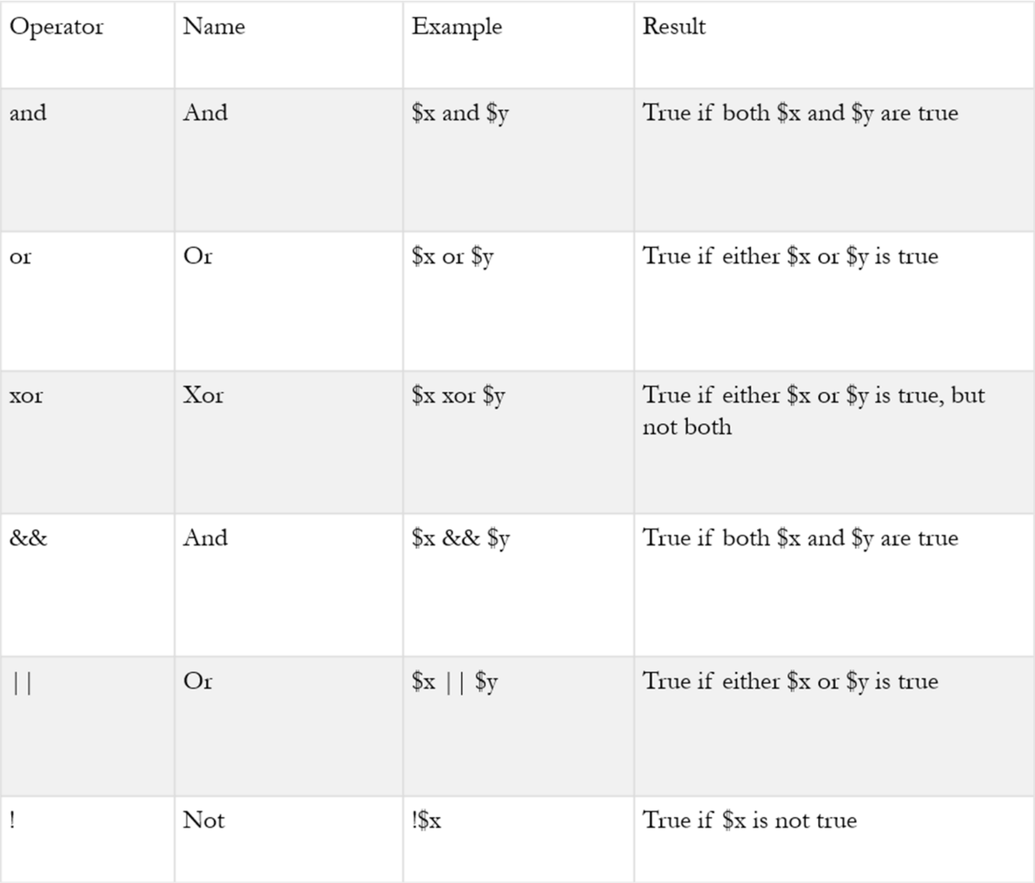
These are called the unary operators as they work on single operands. These are used to increment or decrement values.



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

Logical Operators

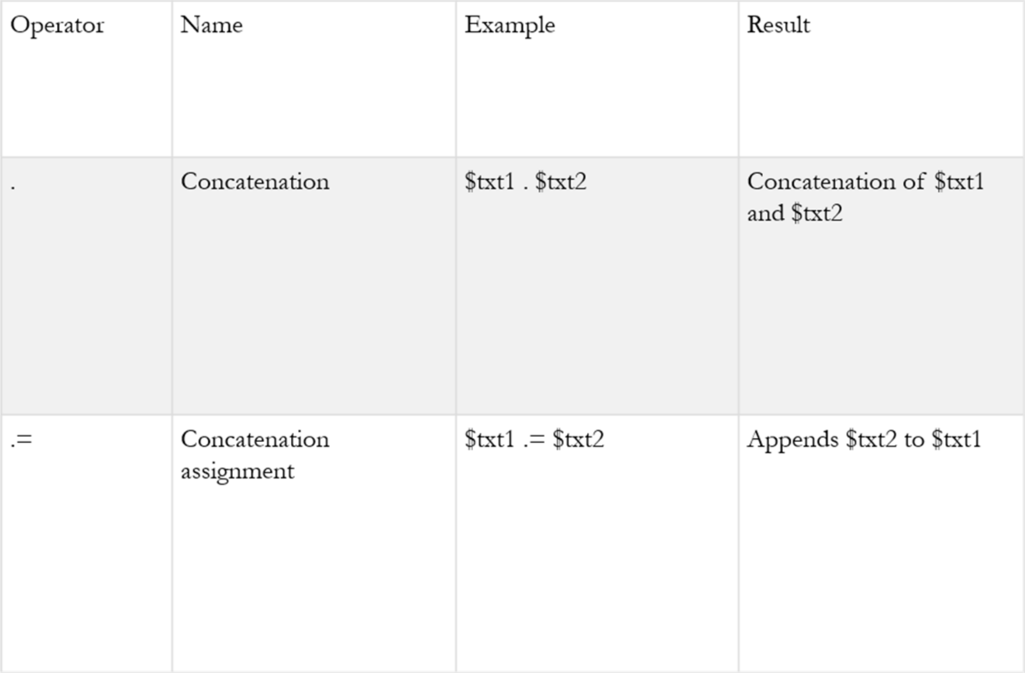
Logical operators are used to operate with conditional statements. These operators evaluate conditions and return a Boolean result (true or false).



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 || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

String Operators

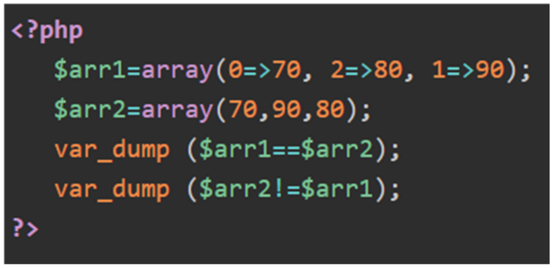
This operator is used for the concatenation of 2 or more strings using the concatenation operator ('.'). We can also use the concatenating assignment operator ('.=') to append the argument on the right side to the argument on the left side.



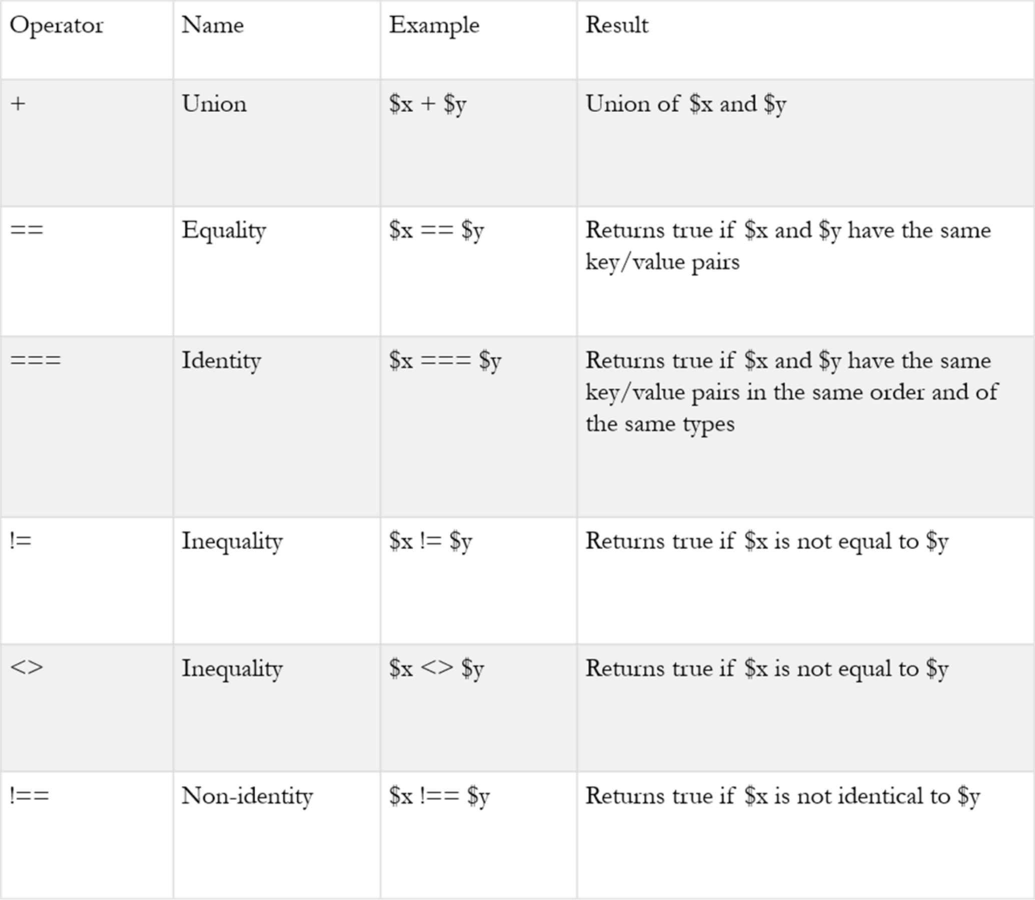
Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

Array Operators

These operators are used in the case of arrays. Here are the array operators, along with their syntax and operations, that PHP provides for the array operation.



```
<?php
   $arr1=array(0=>70, 2=>80, 1=>90);
   $arr2=array(70,90,80);
   var_dump ($arr1==$arr2);
   var_dump ($arr2!=$arr1);
?>
```



Operator	Name	Example	Result
+	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