PHP Operators

In this tutorial you will learn how to manipulate or perform the operations on variables and values using operators in PHP.

What is Operators in PHP

Operators are symbols that tell the PHP processor to perform certain actions. For example, the addition (`+`) symbol is an operator that tells PHP to add two variables or values, while the greater-than (`>`) symbol is an operator that tells PHP to compare two values.

The following lists describe the different operators used in PHP.

PHP Arithmetic Operators

The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc. Here's a complete list of PHP's arithmetic operators:

Operator	Description	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

The following example will show you these arithmetic operators in action:

Example **Run this code »**

```php
<?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
?>
```

PHP Assignment Operators

The assignment operators are used to assign values to variables.

Operator	Description	Example	Is The Same As
=	Assign	`$x = $y`	`$x = $y`
+=	Add and assign	`$x += $y`	`$x = $x + $y`
-=	Subtract and assign	`$x -= $y`	`$x = $x - $y`
*=	Multiply and assign	`$x *= $y`	`$x = $x * $y`
/=	Divide and assign quotient	`$x /= $y`	`$x = $x / $y`
%=	Divide and assign modulus	`$x %= $y`	`$x = $x % $y`

The following example will show you these assignment operators in action:

Example

Run this code »

```php
<?php
$x = 10;
echo $x; // Outputs: 10

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

$x = 50;
$x -= 20;
```

PHP Comparison Operators

The comparison operators are used to compare two values in a Boolean fashion.

Operator	Name	Example	Result
==	Equal	`$x == $y`	True if $x is equal to $y
===	Identical	`$x === $y`	True if $x is equal to $y, and they are of the same type
!=	Not equal	`$x != $y`	True if $x is not equal to $y
<>	Not equal	`$x <> $y`	True if $x is not equal to $y
!==	Not identical	`$x !== $y`	True if $x is not equal to $y, or they are not of

			the same type
<	Less than	`$x < $y`	True if $x is less than $y
>	Greater than	`$x > $y`	True if $x is greater than $y
>=	Greater than or equal to	`$x >= $y`	True if $x is greater than or equal to $y
<=	Less than or equal to	`$x <= $y`	True if $x is less than or equal to $y

The following example will show you these comparison operators in action:

Example **Run this code »**

```
<?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
?>
```

PHP Incrementing and Decrementing Operators

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

Operator	Name	Effect
`++$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

The following example will show you these increment and decrement operators in action:

Example **Run this code »**

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

```
echo ++$x; // Outputs: 11
echo $x;   // Outputs: 11

$x = 10;
echo $x++; // Outputs: 10
echo $x;   // Outputs: 11

$x = 10;
echo --$x; // Outputs: 9
echo $x;   // Outputs: 9

$x = 10;
echo $x--; // Outputs: 10
echo $x;   // Outputs: 9
?>
```

PHP Logical Operators

The logical operators are typically 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 || $y`	True if either $$x or $y is true
`!`	Not	`!$x`	True if $x is not true

The following example will show you these logical operators in action:

Example Run this code »

```
<?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.";
}
?>
```

PHP String Operators

There are two operators which are specifically designed for strings.

Operator	Description	Example	Result
`.`	Concatenation	`$str1 . $str2`	Concatenation of $str1 and $str2
`.=`	Concatenation assignment	`$str1 .= $str2`	Appends the $str2 to the $str1

The following example will show you these string operators in action:

Example | Run this code »

```
<?php
$x = "Hello";
$y = " World!";
echo $x . $y; // Outputs: Hello World!

$x .= $y;
echo $x; // Outputs: Hello World!
?>
```

PHP Array Operators

The array operators are used to compare arrays:

Operator	Name	Example	Result
`+`	Union	`$x + $y`	Union of $x and $y
`==`	Equality	`$x == $y`	True if $x and $y have the same key/value pairs
`===`	Identity	`$x === $y`	True if $x and $y have the same key/value pairs in the same order and of the same types
`!=`	Inequality	`$x != $y`	True if $x is not equal to $y
`<>`	Inequality	`$x <> $y`	True if $x is not equal to $y
`!==`	Non-identity	`$x !== $y`	True if $x is not identical to $y

The following example will show you these array operators in action:

Example Run this code »

```
<?php
$x = array("a" => "Red", "b" => "Green", "c" => "Blue");
$y = array("u" => "Yellow", "v" => "Orange", "w" => "Pink");
$z = $x + $y; // Union of $x and $y
var_dump($z);
var_dump($x == $y);   // Outputs: boolean false
var_dump($x === $y);  // Outputs: boolean false
var_dump($x != $y);   // Outputs: boolean true
var_dump($x <> $y);   // Outputs: boolean true
var_dump($x !== $y);  // Outputs: boolean true
?>
```

PHP Spaceship Operator PHP 7

PHP 7 introduces a new spaceship operator (`<=>`) which can be used for comparing two expressions. It is also known as combined comparison operator.

The spaceship operator returns `0` if both operands are equal, `1` if the left is greater, and `-1` if the right is greater. It basically provides three-way comparison as shown in the following table:

Operator	<=> Equivalent
`$x < $y`	`($x <=> $y) === -1`
`$x <= $y`	`($x <=> $y) === -1 \|\| ($x <=> $y) === 0`
`$x == $y`	`($x <=> $y) === 0`
`$x != $y`	`($x <=> $y) !== 0`
`$x >= $y`	`($x <=> $y) === 1 \|\| ($x <=> $y) === 0`
`$x > $y`	`($x <=> $y) === 1`

The following example will show you how spaceship operator actually works:

Example Run this code »

```
<?php
// Comparing Integers
echo 1 <=> 1; // Outputs: 0
echo 1 <=> 2; // Outputs: -1
echo 2 <=> 1; // Outputs: 1
```

```

// Comparing Floats
echo 1.5 <=> 1.5; // Outputs: 0
echo 1.5 <=> 2.5; // Outputs: -1
echo 2.5 <=> 1.5; // Outputs: 1

// Comparing Strings
echo "x" <=> "x"; // Outputs: 0
echo "x" <=> "y"; // Outputs: -1
echo "y" <=> "x"; // Outputs: 1
?>
```