PHP - Operators Types

What are Operators in PHP?

As in any programming language, PHP also has operators which are symbols (sometimes **keywords**) that are predefined to perform certain commonly required operations on one or more operands.

For example, using the expression "4 + 5" is equal to 9. Here "4" and "5" are called **operands** and "+" is called an **operator**.

We have the following types of operators in PHP −

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Assignment Operators
- String Operators
- Array Operators
- Conditional (or Ternary Operators)

This chapter will provide an overview of how you can use these operators in PHP. In the subsequent chapters, we will take a closer look at each of the operators and how they work.

Arithmetic Operators in PHP

We use Arithmetic operators to perform mathematical operations like addition, subtraction, multiplication, division, etc. on the given operands. Arithmetic operators (excluding the increment and decrement operators) always work on two operands, however the type of these operands should be the same.

The following table highligts the arithmetic operators that are supported by PHP. Assume variable "$a" holds 42 and variable "$b" holds 20 −

Operator	Description	Example

+	Adds two operands	$a + $b = 62
-	Subtracts second operand from the first	$a - $b = 22
*	Multiply both operands	$a * $b = 840
/	Divide numerator by de-numerator	$a / $b = 2.1
%	Modulus Operator and remainder of after an integer division	$a % $b = 2
++	Increment operator, increases integer value by one	$a ++ = 43
--	Decrement operator, decreases integer value by one	$a -- = 42

Comparison Operators in PHP

You would use Comparison operators to compare two operands and find the relationship between them. They return a Boolean value (either true or false) based on the outcome of the comparison.

The following table highligts the comparison operators that are supported by PHP. 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 > $b) is false
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	($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 false

<=	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

Logical Operators in PHP

You can use Logical operators in PHP to perform logical operations on multiple expressions together. Logical operators always return Boolean values, either true or false.

Logical operators are commonly used with conditional statements and loops to return decisions according to the Boolean conditions. You can also combine them to manipulate Boolean values while dealing with complex expressions.

The following table highligts the logical operators that are supported by PHP.

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

Assignment Operators in PHP

You can use Assignment operators in PHP to assign or update the values of a given variable with a new value. The right-hand side of the assignment operator holds the value and the left-hand side of the assignment operator is the variable to which the value will be assigned.

The data type of both the sides should be the same, else you will get an error. The associativity of assignment operators is from right to left. PHP supports two types of assignment operators −

- **Simple Assignment Operator** − It is the most commonly used operator. It is used to assign value to a variable or constant.
- **Compound Assignment Operators** − A combination of the assignment operator (=) with other operators like +, *, /, etc.

The following table highligts the assignment operators that are supported by PHP −

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

String Operators in PHP

There are two operators in PHP for working with string data types −

- The "." (dot) operator is PHP's concatenation operator. It joins two string operands (characters of right hand string appended to left hand string) and returns a new string.
- PHP also has the ".=" operator which can be termed as the concatenation assignment operator. It updates the string on its left by appending the characters of right hand operand.

```
$third = $first . $second;
$leftstring .= $rightstring;
```

Array Operators in PHP

PHP defines the following set of symbols to be used as operators on array data types –

Symbol	Example	Name	Result
+	$a + $b	Union	Union of $a and $b.
==	$a == $b	Equality	TRUE if $a and $b have the same key/value pairs.
===	$a === $b	Identity	TRUE if $a and $b have the same key/value pairs in the same order and of the same types.
!=	$a != $b	Inequality	TRUE if $a is not equal to $b.
<>	$a <> $b	Inequality	TRUE if $a is not equal to $b.
!==	$a !== $b	Non identity	TRUE if $a is not identical to $b.

Conditional Operators in PHP

There is one more operator in PHP which is called conditional operator. It is also known as ternary operator. It first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

Operator	Description	Example

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

Operator Categories in PHP

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.

Operator Precedence in PHP

Precedence of operators decides the order of execution of operators in an expression. For example, in "2+6/3", division of 6/3 is done first and then the addition of "2+2" takes place because the division operator "/" has higher precedence over the addition operator "+".

To force a certain operator to be called before other, parentheses should be used. In this example, (2+6)/3 performs addition first, followed by division.

Some operators may have the same level of precedence. In that case, the order of associativity (either left or right) decides the order of operations. Operators of same precedence level but are non-associative cannot be used next to each other.

The following table lists the PHP operators in their decreasing order of precedence –

Operators	**Purpose**
clone new	clone and new
**	exponentiation
++ --	increment/decrement
~(int) (float) (string) (array) (object) (bool)	casting

instanceof	types
!	logical
* /	multiplication/division
%	modulo
+ - .	arithmetic and string
<< >>	bitwise shift
< <= > >=	comparison
== != === !== <> <=>	comparison
&	bitwise and/references
^	bitwise XOR
|	bitwise OR
&&	logical and
||	logical or
??	null coalescing
? :	ternary
= += -= *= **= /= .= %= &= |= ^= <<= >>= ??=	assignment operators
yield from	yield from
yield	yield
print	print
and	logical
xor	logical
or	logical