Arithmetic Operators

- Arithmetic operators are used to perform basic mathematical operations.

```php
<?php
// Arithmetic Operators
// Addition (+)
// Adds two numbers.
$result = 5 + 3; // Result is 8

// Subtraction (-)
// Subtracts the second number from the first.
$result = 5 - 3; // Result is 2

// Multiplication (*)
// Multiplies two numbers.
$result = 5 * 3; // Result is 15

// Division (/)
// Divides the first number by the second.
$result = 5 / 2; // Result is 2.5

// Modulus (%)
// Returns the remainder of division.
$result = 5 % 2; // Result is 1

// Exponentiation (**)
// Raises the first number to the power of the second.
$result = 5 ** 3; // Result is 125

?>
```

Logical Operators

- Logical operators are used to combine or modify conditional statements.

```php

<?php

$a = true;
$b = false;

// Logical AND (&&)
// Returns true if both conditions are true.
```

```
$result = $a && $b; // Result is false

// Logical OR (||)
// Returns true if at least one condition is true.
$result = $a || $b; // Result is true

// Logical NOT (!)
// Reverses the boolean value of a condition.
$result = !$a; // Result is false

?>
```

Comparison Operators

- Comparison operators are used to compare two values.

```
$c = 5;
$d = '5';

// Equal (==)
// Checks if two values are equal.
$result = ($c == $d); // Result is true

// Identical (===)
// Checks if two values are equal and of the same type.
$result = ($c === $d); // Result is false

// Not Equal (!= or <>)
// Checks if two values are not equal.
$result = ($c != $d); // Result is false

// Not Identical (!==)
// Checks if two values are not equal or not of the same type.
$result = ($c !== $d); // Result is true

// Greater Than (>)
// Checks if the first value is greater than the second.
$result = ($c > 3); // Result is true

// Less Than (<)
// Checks if the first value is less than the second.
$result = ($c < 3); // Result is false

// Greater Than or Equal To (>=)
// Checks if the first value is greater than or equal to the second.
$result = ($c >= 5); // Result is true

// Less Than or Equal To (<=)
```

```php
// Checks if the first value is less than or equal to the second.
$result = ($c <= 3); // Result is false

?>
```

`$x++` (Post-Increment)

- **Definition:** The current value of `$x` is used in the expression, and the increment happens **afterwards**.

Example 1: Using `$x++`

```
<?php
$x = 5;
echo "Before increment: $x<br>"; // Outputs: 5
$y = $x++; // Assigns the current value of $x (5) to $y, then increments $x.
echo "Value of y: $y<br>";       // Outputs: 5
echo "After increment: $x<br>"; // Outputs: 6
?>

//Output
Before increment: 5
Value of y: 5
After increment: 6
```

Explanation:

1. `$x` starts at 5.
2. `$y = $x++;` assigns `5` (current value of `$x`) to `$y`.
3. After assignment, `$x` is incremented to `6`.

`++$x` (Pre-Increment)

- **Definition:** The value of `$x` is incremented **first**, and the new value is used in the expression.

Example 2: Using `++$x`

```

<?php
$x = 5;
echo "Before increment: $x<br>"; // Outputs: 5
$y = ++$x; // Increments $x first, then assigns the new value (6) to $y.
echo "Value of y: $y<br>";       // Outputs: 6
echo "After increment: $x<br>"; // Outputs: 6
?>

//Output
```

```
Before increment: 5
Value of y: 6
After increment: 6
```

Explanation:

1. `$x` starts at 5.
2. `$y = ++$x;` increments `$x` to `6` and assigns `6` to `$y`.