Precedence and Associativity

This lesson discusses two important characteristics of arithmetic operators, i.e., precedence and associativity.

WE'LL COVER THE FOLLOWING

- Precedence
- Associativity
  - Left Associativity
  - Right Associativity
- Operators with the same Precedence and Associativity
- Conclusion

Precedence

Operator precedence determines which operator is performed first in an expression with more than one operators. Operator precedence in PHP is similar to that of regular arithmetic operators.

- `*`, `/`, `%` operators have **equal** precedence.
- `+`, `-` operators have **equal** precedence.
- `*`, `/`, `%` have a **higher** precedence than `+`, `-`.

The operator with higher precedence is executed first in the expression.

Associativity

Associativity is used when two operators of the same precedence appear in an expression.

Left Associativity

Left associativity occurs when an expression is evaluated from left to right. For example `*` and `/` have the same precedence and their associativity is left

to right, so the expression

```
100 / 10 * 10
```

is treated as:

```
(100 / 10) * 10
```

Right Associativity

Right associativity occurs when an expression is evaluated from right to left. Like most programming languages, `=` and `print` in PHP have right associativity.

Run the following code to see a few examples:

```php
<?php
echo "5 - 3 + 2 = " . (5 - 3 + 2); // 5-3+2 is treated as (5-3)+2
echo "\n";
echo "5 + 3 * 2 = " . (5 + 3 * 2); // 5+3*2 is treated as 5+(3*2)
echo "\n";
echo "15 / 3 * 5 = " . (15 / 3 * 5); // 15/3*5 is treated as (15/3)*5
echo "\n";
echo "42 + 7 % 2 = " . (42 + 7 % 2); // 42+7%2 is treated as 42+(7%2)
?>
```

Operators with the same Precedence and Associativity

Naturally, you must be thinking, what will happen if there are two operators with the same precedence and associativity in an expression.

In this case, the expression is evaluated from left to right, i.e., it evaluates the operator that comes first (on the left).

For instance, `%` and `*` have the same precedence and associativity. Thus, given an expression `$a = 5 * 3 % 2;` the value of `$a` will be 1 because

$$(5 * 3)\%2 = 15\%2 = 1$$

And given an expression `$a = 5 % 3 * 2;` the value of `$a` will become 4

because

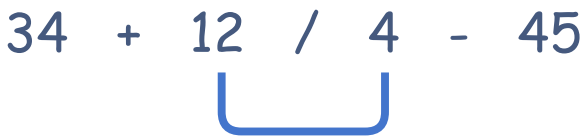
$$(5\%3) * 2 = (2 * 2) = 4$$

Run the code snippet below to see how this works:

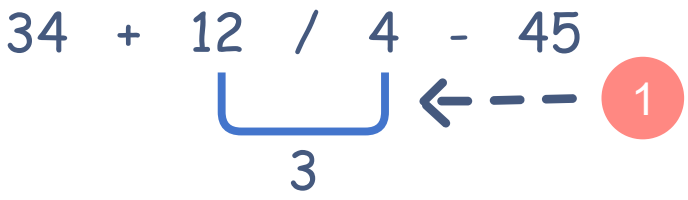
```php
<?php
$a = 5 * 3 % 2; // $a now is (5 * 3) % 2 => (15 % 2) => 1
echo $a . "\n";
$a = 5 % 3 * 2; // $a now is (5 % 3) * 2 => (2 * 2) => 4
echo $a . "\n";
?>
```

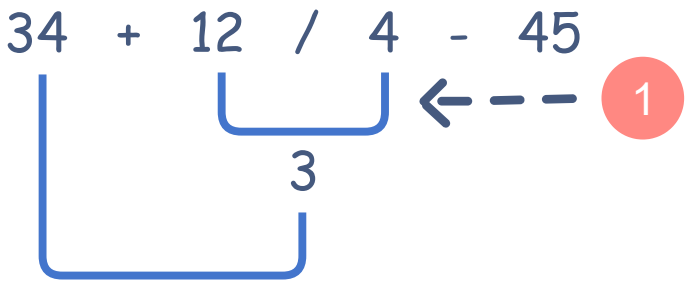
Conclusion

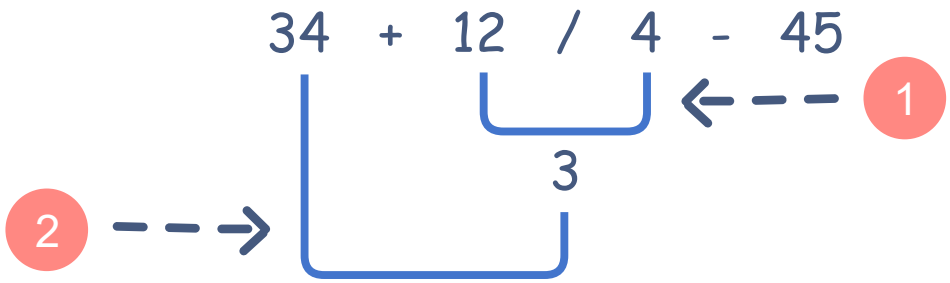
Thus, precedence and associativity are two characteristics of operators that determine the evaluation order of subexpressions in absence of brackets. This is illustrated in the figure below:

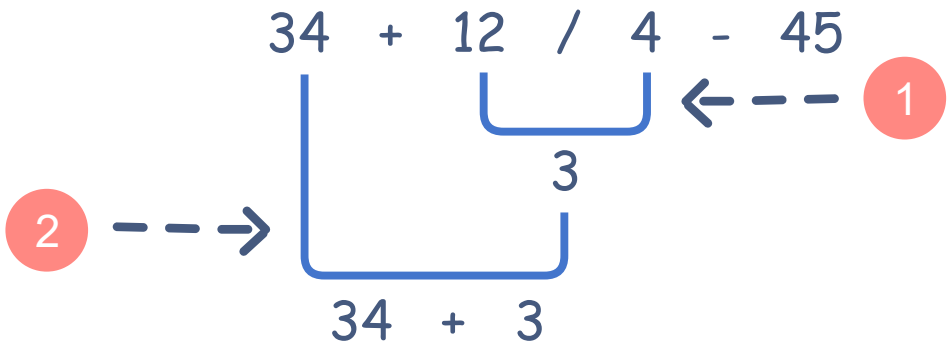

34 + 12 / 4 - 45

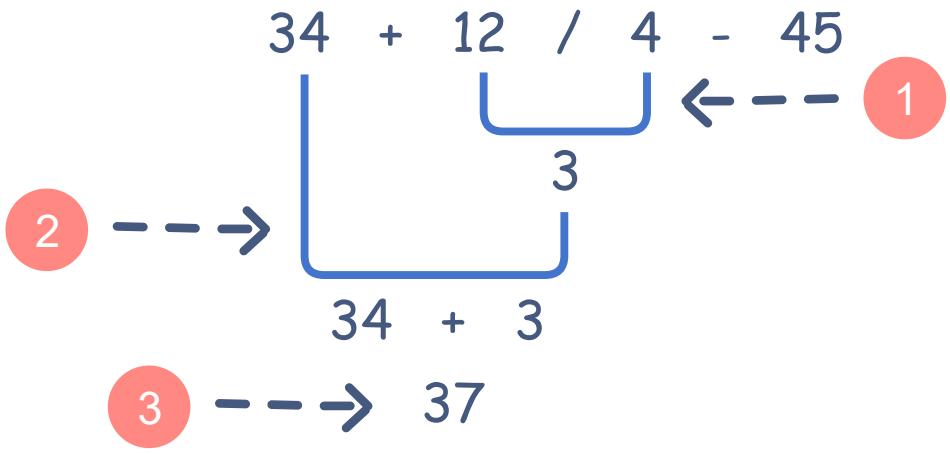
34 + 12 / 4 - 45
1

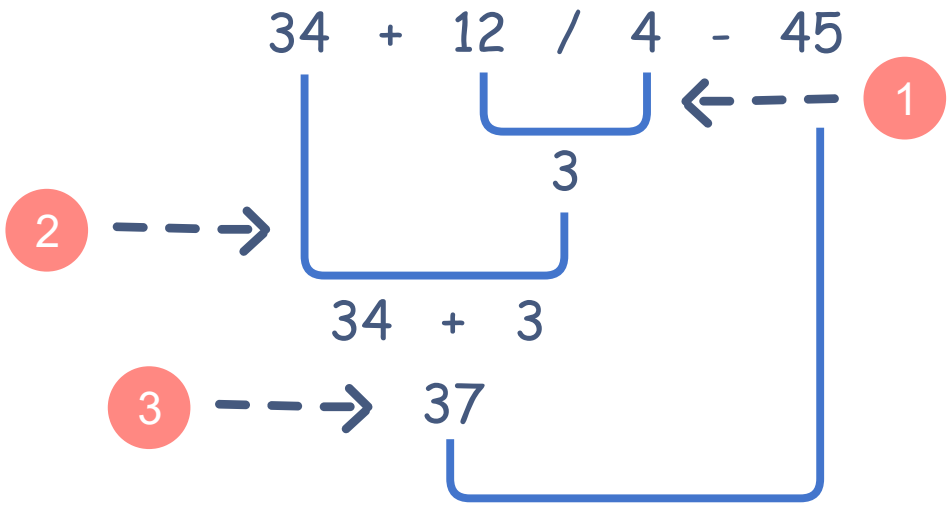

34 + 12 / 4 - 45
1
3

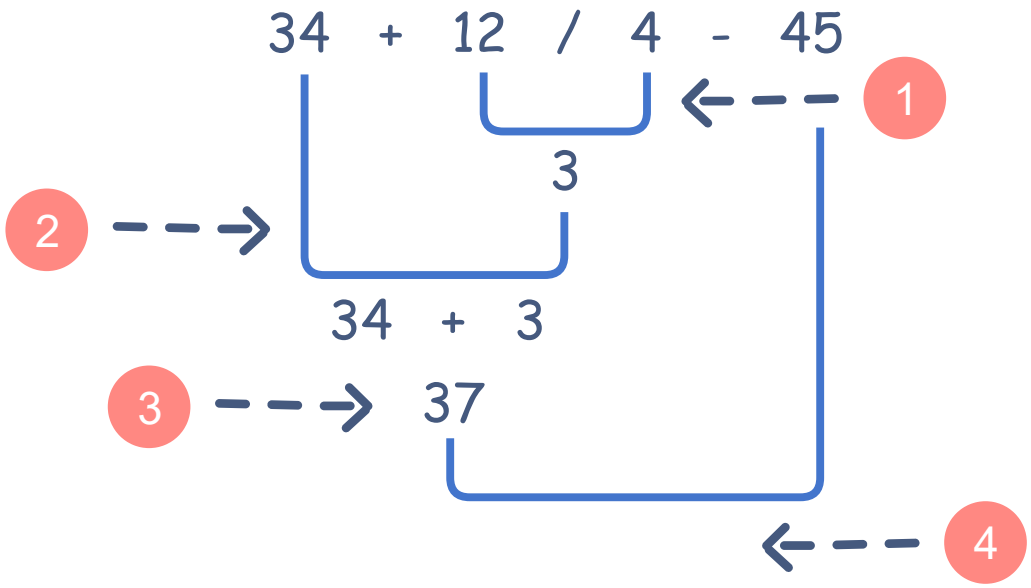

34 + 12 / 4 - 45
1
3

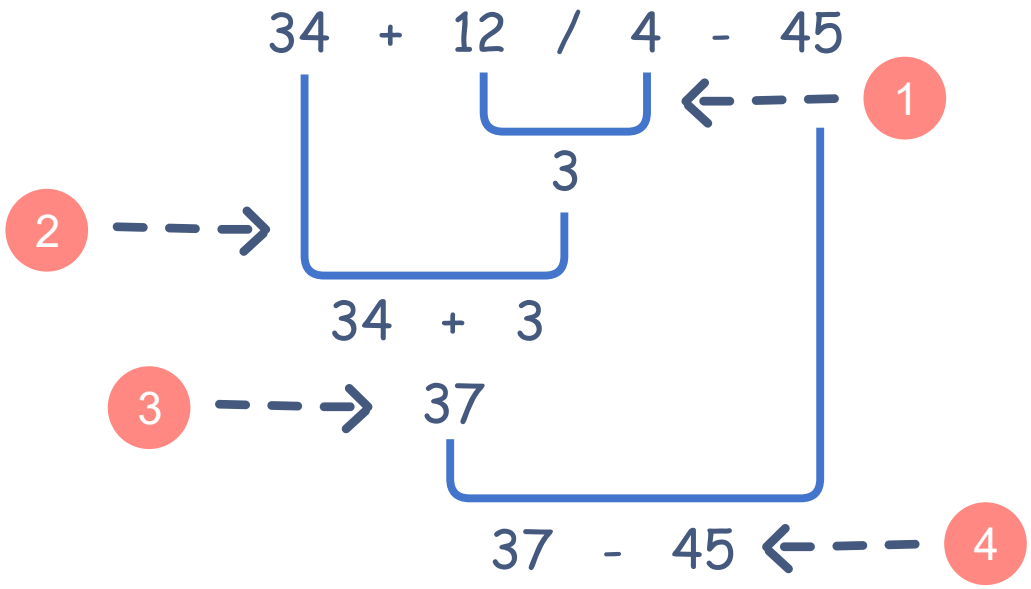

34 + 12 / 4 - 45
1
3
2

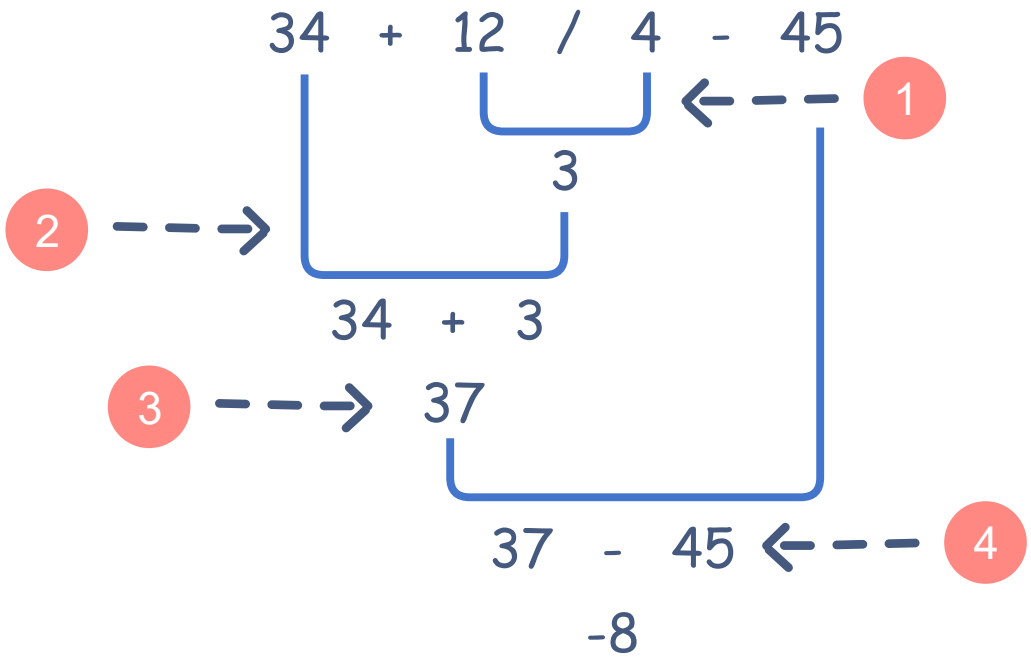

34 + 12 / 4 - 45
1
3
2
34 + 3

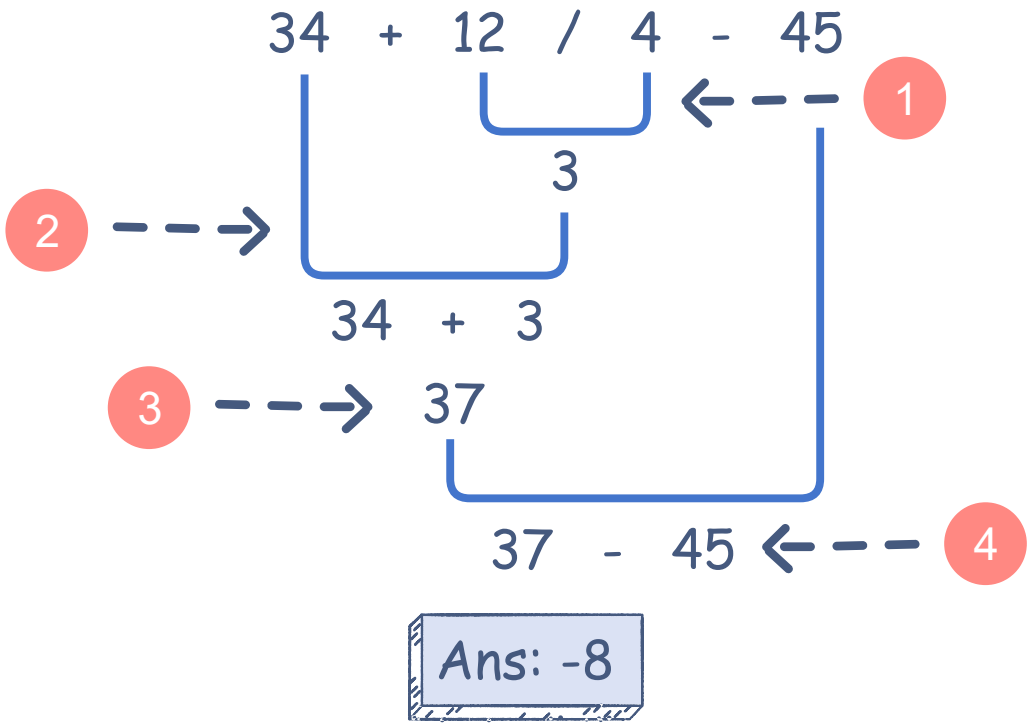

34 + 12 / 4 - 45
1
3
2
34 + 3
3
37


34 + 12 / 4 - 45
1
3
2
34 + 3
3
37


34 + 12 / 4 - 45
1
3
2
34 + 3
3
37
4


34 + 12 / 4 - 45
1
3
2
34 + 3
3
37
37 - 45
4





Remember: It is good to know precedence and associativity rules, but the best thing is to use brackets. Brackets increase readability of the code

> the best thing is to use brackets. Brackets increase readability of the code as the reader doesn't have to see the table to find out the order.

Now that you know all about arithmetic operations and how they behave, let's learn about another type of operators, the comparison operators.