Arithmetic Operators

In the following lesson, you will be introduced to arithmetic operators.

We'll cover the following

A List of Operators

Arithmetic operators are operators that perform arithmetic operations such as addition and subtraction. Below is a list of the arithmetic operators supported by Dart.

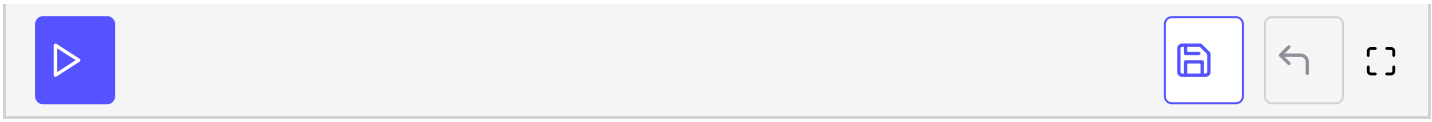
Operator	Use
`+`	Adds two operands
`-`	Subtracts the second operand from the first
`-`expr	Reverses the sign of the expression (unary minus)
`*`	Multiplies both operands

`/`	Divides the first operand by the second operand
`~/`	Divides the first operand by the second operand and returns an integer value
`%`	Gets the remainder after division of one number by another

Taking the first operand to be **10** and the second operand to be 7, let's look at an example for each operator.

```
main() {
  var operand1 = 10;
  var operand2 = 7;

  print(operand1 + operand2);
  print(operand1 - operand2);
  print(- operand1);
  print(operand1 * operand2);
  print(operand1 / operand2);
  print(operand1 ~/ operand2);
  print(operand1 % operand2);
}
```



The output we see when we press run is an expected output.

Most of the operators are ones we see regularly in mathematics. The only one that might be new is `~/`. All `~/` does is divide both operands, remove the fractional part from the decimal value, and keeps the whole number. The difference is clear on **line 9** and **line 10** of the code above.

Prefix and Postfix Operators

Dart also supports both prefix and postfix increment and decrement operators.

Below is a list of the arithmetic prefix and postfix increment and decrement operators supported by Dart.

operators supported by Dart.

Operator	Use
`++` var	var = var + 1
var `++`	var = var + 1
`--` var	var = var - 1
var `--`	var = var - 1

Let's look at how each of these operators work with a few examples.

++var

The expression value of `++var` is `var+1`. When we insert the expression in a print statement, the compiler first increments the variable by **1** and then prints the value of the variable.

```
main() {
  var prefixIncrement = 5;

  print(++prefixIncrement);
}
```

Since the value displayed is after the increment, **6**, i.e., **5+1**, is displayed as the output.

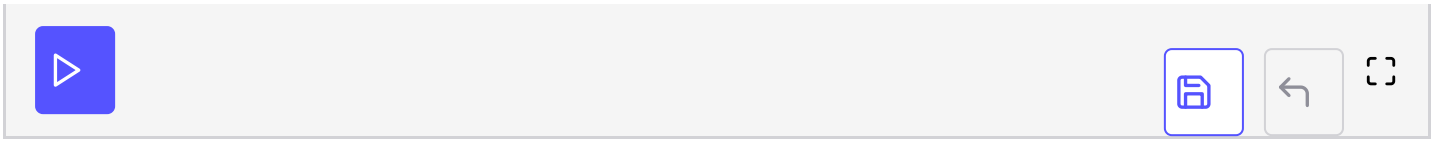
var++

The expression value of `var++` is `var`. When we insert the expression in a print statement, the compiler first prints the value of the variable and then increments it by **1**.

```
main() {
  var postfixIncrement = 5;

  print(postfixIncrement++);
  print(postfixIncrement);
```

```
}
```



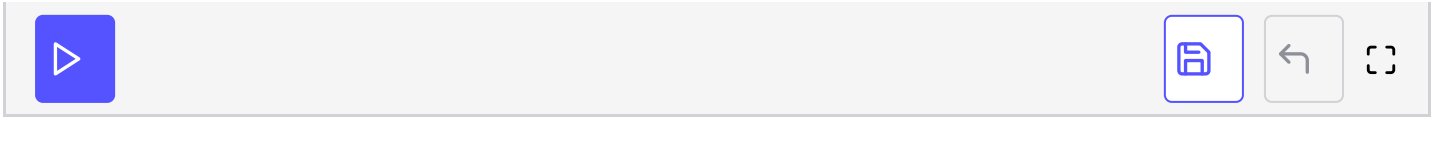
Since, in this case, the `print` statement will display the variable value first and then increment it, **5** is displayed on **line 4**, whereas when the `print` statement is called again the value has already been decremented hence, **line 5** displays **6**.

- -var

The expression value of `--var` is `var-1`. When we insert the expression in a print statement, the compiler first decrements the variable by **1** and then prints the value of the variable.

```
main() {
  var prefixDecrement = 5;

  print(--prefixDecrement);
}
```



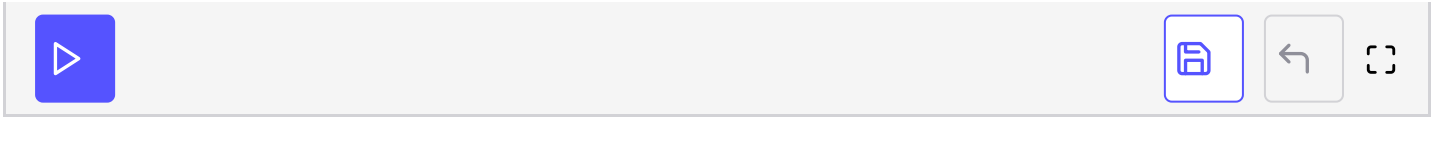
Since the value displayed is after the decrement, **4**, i.e., **5-1**, is displayed as the output.

var- -

The expression value of `var--` is `var`. When we insert the expression in a print statement, the compiler first prints the value of the variable and then decrements it by **1**.

```
main() {
  var postfixDecrement = 5;

  print(postfixDecrement--);
  print(postfixDecrement);
}
```



Since, in this case, the `print` statement will display the variable value first and then decrement it, **5** is displayed on **line 4**, whereas when the `print` statement is called again the value has already been decremented hence, **line 5** displays **4**.

That sums up arithmetic operators. Let’s move on to equality and relational operators in the next lesson.