

# Perl Operators

## *How to use operators in Perl*

---

Page Resource[CGI Resources](#) | [Perl Tutorials](#)

---

[Home/CGI/Perl/Operators](#)

Before we can go much further, we will need to look at how operators are used in Perl, and what they are. These will be important in the upcoming sections where comparison, logic, and some mathematics will be used in conditional blocks. So, let's take a look at the Perl Operators.

### Some Important Notes

In Perl, there are different versions of the operators for numbers and strings. For instance, if you want to compare a number, you would use a traditional symbol such as `<`, `>`, and so on. However, when you compare two strings, the less-than and greater-than signs are not used. Instead, a special version is used to compare strings. Less-than would be the two letters **lt** and greater-than would be **gt**. In the lists that follow, there will be separate listings for numerical and string operators where this is necessary.

### Arithmetic Operators

These operators are used to perform mathematical calculations on numbers. Keep in mind though, they are not used to combine strings. There are some special string operators for this. Note that the assignment operator does work both ways, we used it to assign values to variables in the last section. Here is the list:

Operator	Function
+	Addition
-	Subtraction, Negative Numbers, Unary Negation
*	Multiplication
/	Division
%	Modulus
**	Exponent

To use these, you will place them in your statements like a mathematical expression. So, if you want to store the sum of two variables in a third variable, you would write something like this:

```
$advenue=20;  
$sales=10;  
$total_revenue=$advenue+$sales;
```

You can use the other arithmetic operators in the same way, it is quite similar to other programming

languages.

## Assignment Operators

We have already used the = sign as an assignment operator to assign values to a variable. You can also use the = sign with another arithmetic operator to perform a special type of assignment. You can precede the = sign with the + operator, for example:

```
$revenue+=10;
```

What this does is create a shorthand for writing out the following statement:

```
$revenue=$revenue+10;
```

It takes the variable \$revenue and assigns it the value of \$revenue (itself) plus 10. So, if you had an initial value for \$revenue set at 5:

```
$revenue=5;
$revenue=$revenue+10;
```

After these statements, \$revenue is 15. It added 10 to the value it had before the new assignment.

The others work the same way, but perform the various different operations. Here a list of the arithmetic operators we used above when we place them with the assignment operator:

Operator	Function
=	Normal Assignment
+=	Add and Assign
-=	Subtract and Assign
*=	Multiply and Assign
/=	Divide and Assign
%=	Modulus and Assign
**=	Exponent and Assign

Remember, these are used for the sake of typing less. You can write the statements out the long way if it makes it more understandable when you read your code.

## Increment/Decrement

Another shorthand method is to use the increment and decrement operators, rather than writing out something like this:

```
$revenue=$revenue+1;
```

You can simply write something like this:

```
$revenue++;
```

However, using these operators you must remember that you could also write something like this:

```
++$revenue;
```

If you place the ++ before the variable name, it the variable adds one to itself before it is used or evaluated. For example, if you write:

```
$revenue=5;
$total= ++$revenue + 10;
```

The \$revenue variable is incremented before it is used in the calculation, so it is changed to 6 before 10 is added to it. Thus, \$total turns out to be 16 here.

If you want to increment the variable after it is used, you use the ++ after the variable name:

```
$revenue=5;
$total= $revenue++ + 10;
```

This way \$total is only 15 because \$revenue is used before being incremented, so it stays at 5 for this expression. If you use \$revenue again after this, it will have a value of 6.

With that in mind, here is the short list of the two operators:

Operator	Function
++	Increment (Add 1)
--	Decrement (Subtract 1)

The -- operator works the same way as ++, but it subtracts one from the value of the variable (decrements it).

## Adding Strings

Like I mentioned at the beginning of this section, there are different operators for strings under certain conditions. If you want to put two strings together (also called concatenate), you will want to use the dot operator. Unlike C and JavaScript (where it is used with objects), the dot operator in Perl concatenates two strings.

For example, if you want to place two strings together, you could do this:

```
$full_string="light" . "house";
```

This would make \$full\_string have the value of **lighthouse**. This is more useful if you are using variables for this:

```
$word1="light";
$word2="house";
$full_string=$word1 . $word2;
print "If I had a $word1 and a $word2, would I be able to make a $full_string?";
```

Yes, it prints out this silly little sentence:

```
If I had a light and a house, would I be able to make a lighthouse?
```

This can also be used with the assignment operator to do what we did with numbers earlier. In this case, it gives the string the value of itself put together with another string:

```
$word1="light";
$full_string=$word1 . "house";
```

Of course, we again get the value of lighthouse for the \$full\_string variable. Here is the list for these two string operators:

Operator	Function
.	Concatenate Strings
.=	Concatenate and Assign

Well, that's enough for one section. In the next section, we'll pick up with operators for comparison and more. Let's go on to [Perl Operators: Part 2](#)

---

## Partners

---

[CoolHomepages](#) | [Web Design Library](#) | [Website Content](#)

---

The tutorials and articles on these pages are © 1997-2005 by John Pollock and may not be reposted without written permission from the author, and may not be reprinted for profit.

---



[Previous](#)

By: [John Pollock](#)



[Next](#)

---

[Main Page](#) | [HTML](#) | [JavaScript](#) | [Graphics](#) | [DHTML/Style Sheets](#) | [ASP/PHP](#)  
[PutWeb/FTP](#) | [CGI/Perl](#) | [Promotion](#) | [Java](#) | [Design Articles](#)  
[Support Forums](#) | [Site Search](#) | [FAQs](#) | [Privacy](#) | [Contact](#)

---

Copyright © 1997-2005 [The Web Design Resource](#). All rights reserved.