
Cyber Aces

Module 3 – System Administration

Web Scripting – PHP Operators

By Tom Hessman & Michael Coppola

Presented by Tim Medin

v15Q1

This tutorial is licensed for personal use exclusively for students and teachers to prepare for the Cyber Aces competition. You may not use any part of it in any printed or electronic form for other purposes, nor are you allowed to redistribute it without prior written consent from the SANS Institute.

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited.

1

Welcome to Cyber Aces, Module 3! This module provides an introduction to the Apache Web Server, HTML, PHP, and basic web security.

Course Roadmap

- Introduction
- Apache & HTML
- **PHP**
- Basic Web Security
- Conclusion

- PHP Syntax
- Variables
- Echo Statement
- Strings
- Math Operators
- Comparison Operators
- Logical Operators
- If...Else
- While loops
- For loops
- Using PHP from the CLI
- Include files

Course Roadmap

In this section, we will learn about PHP operators.

Operators

- Operators are used to manipulate data or variables
 - They are most commonly used with variables, but can also be used with literal (in-line) data
- In addition to mathematical operators (5 + 5), there are comparison operators, assignment operators, string operators, and more
- Note that if an operator is contained within a string, it will not be interpreted as an operator, but as its literal character
- We've already covered the assignment operator, "=", which is used to assign a value to a variable:
`$myVariable = "some value";`

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited.

3

Operators

Operators are used to manipulate data or variables. They are mostly used with variables, but can also be used with literal (in-line) data. In addition to mathematical operators (5 + 5), there are comparison operators, assignment operators, string operators, and more. Note that if an operator is contained within a string, it will not be interpreted as an operator, but as its literal character.

We've already covered the assignment operator, "=", which is used to assign a value to a variable:

```
$myVariable = "some value";
```

Further reading: <http://www.tizag.com/phpT/operators.php>

Arithmetic Operators

Operator	What it does	Example	Result
+	Addition	7 + 9	16
-	Subtraction	47 - 5	42
*	Multiplication	12 * 9	108
/	Division	42 / 6	7
%	Modulus	5 % 2	1

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited.

4

Arithmetic Operators

Arithmetic Operators are used to perform basic mathematical operations, such as addition (+), subtraction (-), multiplication, division (/), and modulus (%). Modulus is an operation that returns the remainder after a division operation, which has many uses in programming (such as determining if a number is odd or even by dividing it by two and seeing if the remainder is zero).

Comparison Operators

Operator	What it does	Example	Result
==	Equal To	1 == 1	TRUE
!=	Not Equal To	2 != 2	FALSE
<	Less Than	2 < 3	TRUE
>	Greater Than	2 > 3	FALSE
<=	Less Than or Equal To	2 <= 2	TRUE
>=	Greater Than or Equal To	3 >= 2	TRUE

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited.

5

Comparison Operators

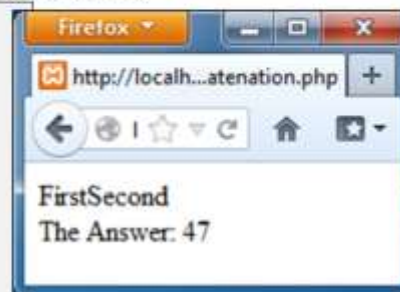
Comparison Operators are used to compare values. They return a boolean value of either "TRUE" or "FALSE", and are commonly used within conditional statements to test whether a condition has been met. The "==" operator checks if two values are equal to one another (don't confuse it with the assignment operator, "="). The "!=" operator checks if two values are NOT equal to one another. The < and > operators check if a value is less than or greater than another, respectively, and the <= and >= check if a value is less than or equal to or greater than or equal to another value, respectively.

Concatenation Operator

- The "." operator concatenates (combines) strings together
- It can also be used to combine multiple variables or expressions together into a single string

```
<?php
$x = "First";
$y = "Second";
$z = 42;

echo $x . $y;
echo "<br>";
$new_string = $x . $y . $z;
echo "The Answer: " . ($z + 5);
```



Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited.

6

Concatenation Operator

The concatenation operator, "." (a single period), combines strings together. It can also be used to combine multiple variables or expressions together into a single string.

In the example above, the first echo statement displays the strings "\$x" and "\$y" together, which would display "FirstSecond" to the browser. The next line combines \$x, \$y, and \$z into a single string and assigns it to the variable "\$new_string" (which will have the contents "FirstSecond42"). The last echo statement displays the text "The Answer: ", followed by the results of the mathematical expression "\$z + 5" (which evaluates to 47). Note that the parenthesis are necessary around the "\$z + 5" operation due to operator precedence (order of operations)...without them, the string "The Answer: " and the variable "\$z" would be concatenated together, and then added to 5, which results in an answer of 5.

Shorthand Assignment Operators

Operator	What it does	Example	Equivalent Operation
<code>+=</code>	Plus Equals	<code>\$x += 2;</code>	<code>\$x = \$x + 2;</code>
<code>-=</code>	Minus Equals	<code>\$x -= 2;</code>	<code>\$x = \$x - 2;</code>
<code>*=</code>	Multiply Equals	<code>\$x *= 2;</code>	<code>\$x = \$x * 2;</code>
<code>/=</code>	Divide Equals	<code>\$x /= 2;</code>	<code>\$x = \$x / 2;</code>
<code>%=</code>	Modulo Equals	<code>\$x %= 2;</code>	<code>\$x = \$x % 2;</code>
<code>.=</code>	Concatenate Equals	<code>\$str .= "hi";</code>	<code>\$str = \$str . "hi";</code>

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited.

7

Shorthand Assignment Operators

While programming, it is common to have to alter the value of a variable by a fixed amount, such for a counter. These shorthand assignment operators were created to make these repetitive tasks easier. The most commonly used shorthand assignment operator is "plus equals", to increment a variable by a certain amount. In the example above, the variable "\$x" is incremented by 2, which is the same as the operation `$x = $x + 2;`. "Concatenate equals" is also very common in PHP, used to keep adding more data into a string, such as to build up a string of display data and then display it to the user.

Increment/Decrement Operators

- Incrementing or decrementing a variable by 1 is very common, and has its own operators, ++ and --
 - `$x++;` is equivalent to `$x = $x + 1;`
 - `$x--;` is equivalent to `$x = $x - 1;`
- If the ++ or -- is placed after the variable name, the variable will be changed *after* the current value is returned
 - The following example will display "3" to the browser:

```
<?php
$x = 3;
echo $x++;
```
- If the ++ or -- is placed before the variable name, the variable will be changed *before* the current value is returned
 - The following example will display "4" to the browser:

```
<?php
$x = 3;
echo ++$x;
```

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited.

8

Increment/Decrement Operators

Incrementing or decrementing a variable by 1 is a very common task in programming, so it has its own operators, ++ and --. The "++" operation is equivalent to setting a variable equal to itself plus one, and the "--" operation is equivalent to setting a variable equal to itself minus one.

If the ++ or -- is placed after the variable name, the variable will be incremented or decremented *after* its value is returned in the current expression. In the first example above, the value "3" is displayed to the browser, because PHP returns the value of the variable before it increments it by 1.

If the ++ or -- is placed before the variable name, the variable will be incremented or decremented *before* the current value is returned in the current expression. In the second example above, the value "4" is displayed to the browser, because PHP increments the variable by 1 before it returns its value.

Most of the time, you'll probably be incrementing the variable separately from other tasks, so this distinction won't matter. However, there are use cases where it is convenient to combine statements together, and then this distinction becomes important!

Logical Operators

Operator	Name	Example	Result
and &&	And	\$a and \$b \$a && \$b	TRUE if both \$a and \$b are TRUE
or 	Or	\$a or \$b \$a \$b	TRUE if either \$a or \$b are TRUE (or if they're both TRUE)
xor	Exclusive OR (XOR)	\$a xor \$b	TRUE if either \$a or \$b is TRUE, but not both
!	Not	! \$a	TRUE if \$a is FALSE

Cyber Aces Module 3 - ©2015 The SANS Institute. Redistribution Prohibited.

9

Logical Operators

Logical Operators are used when multiple conditions need to be checked in an expression. For example, you may need to check if someone's first name is "Chuck", AND that their last name is "Bartowski". The "and" operator returns TRUE when both expressions are TRUE. The "or" operator returns TRUE if either expression is TRUE (or if both are TRUE). The "xor" (exclusive or) operator returns TRUE if one expression is true but the other is not (not if both are TRUE). The "!" (not) operator inverts a comparison, checking if something is not TRUE.

The "&&" operator is essentially identical to the "and" operator, but "&&" has a higher precedence (meaning it gets evaluated first). The same is true of "||" and "or".

Review

- What will the following code snippet display to the user?

```
<?php  
echo 5 + 5 * 2;
```

- a. 15
- b. 25
- c. 20
- d. 5 + 5 * 2

- What will the following code snippet display to the user?

```
<?php  
$x = 40;  
$x += 2;  
echo $x++;
```

- a. 42
- b. 40
- c. 43
- d. Nothing, because the code snippet is invalid.

Review

What will the following code snippet display to the user?

```
<?php
```

```
echo 5 + 5 * 2;
```

- a. 15
- b. 25
- c. 20
- d. 5 + 5 * 2

What will the following code snippet display to the user?

```
<?php
```

```
$x = 40;
```

```
$x += 2;
```

```
echo $x++;
```

- a. 42
- b. 40
- c. 43
- d. Nothing, because the code snippet is invalid.

Answers

- What will the following code snippet display to the user?

```
<?php  
echo 5 + 5 * 2;
```

The answer is A: 15

PHP follows the same order of operations (PEMDAS) as mathematics, so the multiplication occurs before the addition (though parenthesis can be used to change this)
- What will the following code snippet display to the user?

```
<?php  
$x = 40;  
$x += 2;  
echo $x++;
```

The answer is A: 42

\$x is initially set to 40, and is then incremented by 2. Since the ++ is after the variable name, the current value (42) gets returned to the echo statement before the variable is incremented by 1.

Answers

What will the following code snippet display to the user?

```
<?php  
echo 5 + 5 * 2;
```

The answer is A: 15

PHP follows the same order of operations (PEMDAS) as mathematics, so the multiplication occurs before the addition (though parenthesis can be used to change this)

What will the following code snippet display to the user?

```
<?php  
$x = 40;  
$x += 2;  
echo $x++;
```

The answer is A: 42

\$x is initially set to 40, and is then incremented by 2. Since the ++ is after the variable name, the current value (42) gets returned to the echo statement before the variable is incremented by 1.

Tutorial Complete!

- This concludes the introduction to PHP Operators
- Next, you'll learn about Flow Control in PHP

This concludes the first section of Module 3. We've learned the basics PHP operators. Next, we'll learn about PHP's Flow Control.