ITEXAMTOOLS

Cisco Press Publications | Online Degrees | Pearson e-Books | Q&A and FREE downloads IT Learning Lessons

Building Web Applications in PHP

success Featured, PHP Edit Post

Easy IN-Page Links

Building Web Applications in PHP covers the following asapects to learn:

DISCLOSURE

ITExamtools.com Is learner supported & contains affiliate links, meaning when you make a purchase, we receive a small commission.

TOP online Courses

- Top 25 online courses in Mexico
- Top 25 online courses in Thailand
- Top 25 online courses in New Zealand
- Top 25 online courses in Pakistan
- Top 25 online courses in Nigeria
- Top 23 online courses in Canada

How a web browser interacts with a web server, request/response cycle, including GET/POST/Redirect. an introductory understanding of Hypertext Markup Language (HTML), as well as the basic syntax and data structures of the PHP language, variables, logic, iteration, arrays, error handling, and superglobal variables, among other elements. An introduction to Cascading Style Sheets (CSS) will allow you to style markup for web pages, & the skills and knowledge to install and use an integrated PHP/MySQL environment like XAMPP or MAMP.

First let us see the The Building Blocks of PHP

This starting class covers some of the nuts and bolts of the PHP scripting language, including a few PHP-specific features with regards to global variables, data types, and changing types.

Those of you new to programming might feel overwhelmed at times, but don't worry—you can always refer to this chapter later on. Concentrate on understanding the concepts, rather than memorizing the features covered, because these elements will be repeated throughout the scripts in this book. Eventually you'll get it, if not the first time!

If you're already an experienced programmer, you should at least skim this chapter because it covers a few PHP-specific features with regards to global variables, data types, and changing types.

In this chapter, you will learn

- Top 25 online courses in India
- Top 25 online courses in Oman
- Top 25 online courses in Qatar
- Top 25 online courses in South Africa
- Top 25 online courses in Malaysia
- Top 25 online courses in Australia

- About variables—what they are, why you need to use them, and how to use them
- How to define and access variables
- About data types
- About some of the more commonly used operators
- How to use operators to create expressions
- How to define and use constants

Variables

A *variable* is a special container that you can define, which will then “hold” a value, such as a number, string, object, array, or a Boolean. Variables are fundamental to programming. Without variables, you would be forced to hard-code each specific value used in your scripts. The following hard-coded statement adds two numbers together and prints the result, which solves a simple mathematics problem:

```
echo (2 + 4);
```

However, this snippet of code is useful only for people who specifically want to know the sum of 2 and 4. To get past this limitation, you could write a script for finding the sum of another set of numbers, say 3 and 5. However, this approach to programming is clearly absurd, and this is where variables come into play.

Variables allow you to create templates for operations, such as adding two numbers, without worrying about the specific values the variables represent. Values will be given to the variables when the script is run, possibly through user input, through a database query, or from the result of another action earlier in the script. In other words, variables should be used whenever the data in

data:image/gif;base64,R0lGODdhAQABAPAAAP///wAAACwAAAAAAQABAEACAkQBADs=
data:image/gif;base64,R0lGODdhAQABAPAAAP///wAAACwAAAAAAQABAEACAkQBADs=

your script is liable to change—either during the lifetime of the script, or when it is passed to another script for later use.

A variable consists of a name of your choosing, preceded by a dollar sign ($). Variable names can include letters, numbers, and the underscore character (_), but they cannot include spaces. Names must begin with a letter or an underscore. The following list shows some legal variables:

```
$a;

$a_longish_variable_name;

$2453;

$sleepyZZZZ;
```

By the Way

Your variable names should be meaningful as well as consistent in style. For example, if your script deals with name and password values, don't create a variable called $n for the name and $p for the password—those are not meaningful names for anyone other than you, at that particular moment. If you pick up that script weeks later, you might think that $n is the variable for "number" rather than "name" and that $p stands for "page" rather than "password." And what if a co-worker has to modify your script? How will that person know what $n and $p stood for? You can use whatever naming convention you want for variables in your scripts, as long as the names are descriptive and follow some sort of pattern that others can understand.

A semicolon (;)—also known as the *instruction terminator* —is used to end a PHP statement. The semicolons in the previous fragment of code are not part of the variable names, but are used to end the statement that declares the variable as “alive and kicking,” if you will. To declare a variable, you need only include it in your script. When you declare a variable, you usually assign a value to it in the same statement, as shown here:

```
$num1 = 8;
```

```
$num2 = 23;
```

The preceding lines declare two variables and use the assignment operator (=) to assign values to them. You will learn about assignment in more detail in the “Operators and Expressions” section later in this chapter. After you assign values to your variables, you can treat them exactly as if they were the values themselves. In other words

```
echo $num1;
```

is equivalent to

```
echo 8;
```

as long as $num1 is assigned a value of 8.

Globals and Superglobals

In addition to the rules for naming variables, there are rules regarding the availability of variables. In general, the assigned value of a variable is present only within the function or script where it resides. For example, if you have scriptA.php that holds a variable called $name with a value of joe, and you want to create scriptB.php that also uses a $name variable, you can assign to that second $name variable a value of jane without affecting the variable in scriptA.php. The value of the $name variable is *local* to each script, and the assigned values are independent of each other.

However, you can also define the $name variable as *global* within a script or function. If the $name variable is defined as a global variable in both scriptA.php and scriptB.php, and these scripts are connected to each other (that is, one script calls the other or includes the other), there will only be one value for the now-shared $name variable. Examples of global variable scope will be explained in more detail in Chapter 7, “Working with Functions.”

In addition to global variables of your own creation, PHP has several predefined variables called *superglobals*. These variables are always present, and their values are available to all your scripts. Each of the following superglobals is actually an array of other variables:

- $_GET contains any variables provided to a script through the GET method.
- $_POST contains any variables provided to a script through the POST method.
- $_COOKIE contains any variables provided to a script through a cookie.
- $_FILES contains any variables provided to a script through file uploads.
- $_SERVER contains information such as headers, file paths, and script locations.
- $_ENV contains any variables provided to a script as part of the server environment.
- $_REQUEST contains any variables provided to a script via GET, POST, or COOKIE input mechanisms.
- $_SESSION contains any variables that are currently registered in a session.

The examples in this book will use superglobals wherever possible. Using superglobals within your scripts is important in creating secure applications because superglobals reduce the likelihood of user-injected input to your scripts. By coding your scripts to accept only what you want, in the manner defined by you (from a form using the POST method, or from a session, for example), you can eliminate some of the problems created by loosely written scripts.

Data Types

Different types of data take up different amounts of memory and may be treated differently when they are manipulated in a script. Some programming languages therefore demand that the programmer declare in advance which type of data a variable will contain. By contrast, PHP is *loosely typed*, meaning that it will determine the data type at the time data is assigned to each variable.

This automatic typing is a mixed blessing. On the one hand, it means that variables can be used flexibly—in one instance, a variable can hold a string and then later in the script it can hold an integer or some other data type. On the other hand, this flexibility can lead to problems in larger scripts if you are specifically expecting a variable to hold one data type when in fact it holds something completely different. For example, suppose that you have created code to manipulate an array variable. If the variable in question instead contains a number value and no array structure is in place, errors will occur when the code attempts to perform array-specific operations on the variable.

Table 5.1 shows the eight standard data types available in PHP.

Table 5.1. Standard Data Types

Type	Example	Description
Boolean	true	One of the special values true or false
Integer	5	A whole number
Float or double	3.234	A floating-point number
String	"hello"	A collection of characters
Object		An instance of a class
Array		An ordered set of keys and values
Resource		Reference to a third-party resource (a database, for example)
NULL		An uninitialized variable

Resource types are often returned by functions that deal with external applications or files. For example, you will see references to "the MySQL resource ID" in Chapter 18, "Interacting with MySQL Using PHP." The NULL type is reserved for variables that have been declared, but no value has been assigned to them.

PHP has several functions available to test the validity of a particular type of variable—one for each type, in fact. The is_* family of functions, such as is_bool(), tests whether a given value is a Boolean. Listing 5.1 assigns different data types to a single variable and then tests the variable with the appropriate is_* function. The comments in the code show you where the script is in the process.

By the Way

Site Navigation

- Home
- CISCO ALL
  - CCENT & CCNA R&S
  - CCNAv7
  - CCNA Data Center
  - CCNP Data Center
  - Cisco Certified CyberOps Associate. CBROPS 200-201
  - CCNA 200-301 (New CCNA)
- Cisco Career Certifications
  - Cisco Collaboration Exams
    - CCNP/CCIE 350-801[CLCOR] Q&A and More
    - CCNP/Speciali st 300-810 [CLICA] Q&A and more
    - 300-815 [CLACCM] Q&A and more
    - CCNP 300-820 [CLCEI] Q&A and more

You can read more about calling functions in Chapter 7.

Listing 5.1. Testing the Type of a Variable

```
<?php
$testing; // declare without assigning
echo "is null? ".is_null($testing); // checks if
echo "<br/>";
$testing = 5;
echo "is an integer? ".is_int($testing); // check
echo "<br/>";
$testing = "five";
echo "is a string? ".is_string($testing); // chec
echo "<br/>";
$testing = 5.024;
echo "is a double? ".is_double($testing); // chec
echo "<br/>";
$testing = true;
echo "is boolean? ".is_bool($testing); // checks
echo "<br/>";
$testing = array('apple', 'orange', 'pear');
echo "is an array? ".is_array($testing); // check
echo "<br/>";
echo "is numeric? ".is_numeric($testing); // chec
echo "<br/>";
echo "is a resource? ".is_resource($testing); //
echo "<br/>";
echo "is an array? ".is_array($testing); // check
echo "<br/>";
?>
```

Put these lines into a text file called testtype.php, and place this file in your web server document root. When you access this script through your web browser, it produces the following output:

```
is null? 1
is an integer? 1
is a string? 1
is a double? 1
is boolean? 1
is an array? 1
is numeric?
is a resource?
is an array? 1
```

- CCNP 300-835 [CLAUTO] Q&A and more
- Cisco CyberOps Exams
  - CyberOps Associate 200-201 CBROPS Q&A and more
  - CyberOps Professional 350-201 [CBRCOR] Q&A and more
  - CyberOps Professional 300-215 [CBRFIR] Q&A and more
- Cisco Data Center Exams
  - CCNP Data Center 350-601 [DCCOR] Cisco Certified Specialist
  - CCNP Data Center 300-610 [DCID] Cisco Certified Specialist. Q&A and more
  - CCNP Data Center 300-615 [DCIT] Cisco Certified Specialist
  - CCNP Data Center 300-620 [DCACI]Data

When the $testing variable is declared in line 2, no value is assigned to it, so when the variable is tested in line 3 to see whether it is null (using is_null()), the result is 1 (true). After this, values are assigned to $testing by using the = sign before testing it with the appropriate is_* function. An integer, assigned to the $testing variable in line 5, is a whole or real number. In simple terms, you can think of a whole number as a number without a decimal point. A string, assigned to the $testing variable in line 8, is a collection of characters. When you work with strings in your scripts, they should always be surrounded by double or single quotation marks (” or ‘). A double, assigned to the $testing variable in line 11, is a floating-point number (that is, a number that includes a decimal point). A Boolean, assigned to the $testing variable in line 14, can have one of two special values: true or false. In line 17, an array is created using the array() function, which you’ll learn more about in Chapter 8, “Working with Arrays.” This particular array contains three items, and the script dutifully reports $testing to have a type of “array.”

From line 20 through the end of the script, no value is reassigned to $testing—only the type is tested. Lines 20 and 22 test whether $testing is a numeric or resource type, respectively, and because it is not, no value is displayed to the user. In line 24, the script tests again to see whether $testing is an array, and because it is, the value of 1 is displayed.

Changing Type with settype()

PHP also provides the function settype(), which is used to change the type of a variable. To use settype(), you place the variable to change and the type to change it to

Center ACI Implementation

- Cisco Design Exam 352-001
- Cisco DevNet Exams
  - Cisco Certified DevNet Associate 200-901 DEVASC
- Cisco Enterprise Exams
- Cisco Field Tech.Exams
- Cisco Security Exams
- Cisco Service Provider Exams

- VMware
  - VMware Cloud Management and Automation exams
  - VMware Data Center Virtualization
  - VMware Desktop And Mobility
  - VMware Network Virtualization
  - VMware Digital Workspace
  - VMware Digital Business Transformation
- About
- Contact Us

between the parentheses and separate the elements with a comma, like this:

```
settype($variabletochange, 'new type');
```

Listing 5.2 converts the value 3.14 (a float) to each of the four standard types examined in this chapter.

Listing 5.2. Changing the Type of a Variable with settype()

```
1:  <?php
2:  $undecided = 3.14;
3:  echo "is ".$undecided." a double? ".is_double($un
4:  settype($undecided, 'string');
5:  echo "is ".$undecided." a string? ".is_string($un
6:  settype($undecided, 'integer');
7:  echo "is ".$undecided." an integer? ".is_integer(
8:  settype($undecided, 'double');
9:  echo "is ".$undecided." a double? ".is_double($un
10: settype($undecided, 'bool');
11: echo "is ".$undecided." a boolean? ".is_bool($und
12: ?>
```

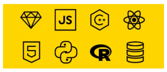
By the Way

Per the PHP Manual, “double” is returned in case of a float, and not simply “float”. Your eyes are not deceiving you.

In each case, we use the appropriate is_* function to confirm the new data type and to print the value of the variable $undecided to the browser using echo. When we convert the string “3.14” to an integer in line 6, any information beyond the decimal point is lost forever. That’s why $undecided contains 3 after we change it back to a double in line 8. Finally, in line 10, we convert $undecided to a Boolean. Any number other than 0 becomes true when converted to a Boolean. When printing a Boolean in PHP, true is represented

Posts

PHP



Which programming language is Important to learn first?

Thinking about learning to code but not sure where to start? One of the most common questions we hear is, “Which programming language is Important

Artificial Intelligence

where to get Best Deep Learning Engineer Jobs

Where to get Best Deep Learning Engineer Jobs

7 skills employers of the future will be looking for

Futurepro of your CV by

as 1 and false is represented as an empty string, so in line 11, $undecided is printed as 1.

Put these lines into a text file called settype.php and place this file in your web server document root. When you access this script through your web browser, it produces the following output:

```
is 3.14 a double? 1
is 3.14 a string? 1
is 3 an integer? 1
is 3 a double? 1
is 1 a boolean? 1
```

Changing Type by Casting

The principal difference between using settype() to change the type of an existing variable and changing type by *casting* is the fact that casting produces a copy, leaving the original variable untouched. To change type through casting, you indicate the name of a data type, in parentheses, in front of the variable you are copying. For example, the following line creates a copy of the $originalvar variable, with a specific type (integer) and a new name $newvar. The $originalvar variable will still be available, and will be its original type; $newvar is a completely new variable.

```
$newvar = (integer) $originalvar
```

Listing 5.3 illustrates changing data types through casting.

Listing 5.3. Casting a Variable

```
1:  <?php
2:  $undecided = 3.14;
3:  $holder = (double) $undecided;
4:  echo "is ".$holder." a double? ".is_double($holde
5:  $holder = (string) $undecided;
6:  echo "is ".$holder." a string? ".is_string($holde
 7:  $holder = (integer) $undecided;
 8:  echo "is ".$holder." an integer? ".is_integer($ho
 9:  $holder = (double) $undecided;
10:  echo "is ".$holder." a double? ".is_double($holde
11:  $holder = (boolean) $undecided;
12:  echo "is ".$holder." a boolean? ".is_bool($holder
13:  echo "<hr/>";
14:  echo "original variable type of $undecided: ";
15:  echo gettype($undecided); // double
16:  ?>
```

Listing 5.3 never actually changes the type of the $undecided variable, which remains a double throughout this script, as illustrated on line 15, where the gettype() function is used to determine the type of $undecided.

By the Way

Despite its usage here, don’t use gettype() to test for a certain type because it can be slow and is likely to be deprecated in future versions. Use the is_* family of functions to test type in production. This usage is simply for illustrative purposes.

In fact, casting $undecided creates a copy that is then converted to the type specified at the time of the cast, and stored in the variable $holder. This casting occurs first in line 3, and again in lines 5, 7, 9, and 11. Because the code is working with only a copy of $undecided and not the original variable, it never lost its original value, as the $undecided variable did in line 6 of Listing 5.2 when its type changed from a string to an integer.

Put the contents of Listing 5.3 into a text file called casttype.php and place this file in your web server document root. When you access this script through your web browser, it produces the following output:

the basic syntax and data structures of the PHP language, variables, logic, iteration, arrays, error handling, and superglobal variables, among other elements

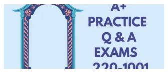
many many new horizons researchers are exploring. this is abut how the AI can be applied to the business models successfully and improve many stages of any business.

Posts

data science

COMPTIA A+

How HADOOP is useful to data



CompTIA A+ Practice Questions Core

```
is 3.14 a double? 1
is 3.14 a string? 1
is 3 an integer? 1
is 3.14 a double? 1
is 1 a boolean? 1
original variable type of 3.14: double
```

Now that you've seen how to change the contents of a variable from one type to another either by using settype() or by casting, consider why this might be useful. It is not a procedure that you will have to use often because PHP automatically casts your variables for you when the context of the script requires a change. However, such an automatic cast is temporary, and you might want to make a variable persistently hold a particular data type—thus, the ability to specifically change types.

For example, the numbers that a user types into an HTML form will be made available to your script as the string type. If you try to add two strings together because they contain numbers, PHP will helpfully convert these strings into numbers while the addition is taking place. So

```
"30cm" + "40cm"
```

results in an answer of 70.

By the Way

The generic term *number* is used here to mean integers and floats. If the user input is in float form, and the strings added together were "3.14cm" and "4.12cm", the answer provided would be 7.26.

During the casting of a string into an integer or float, PHP will ignore any non-numeric characters. The string will be truncated, and any characters from the location of the first non-numeric character onward are ignored. So, whereas “30cm” is transformed into “30”, the string “6ft2in” becomes just 6 because the rest of the string evaluates to zero.

You might want to clean up the user input yourself and use it in a particular way in your script. Imagine that the user has been asked to submit a number. We can simulate this by declaring a variable and assigning the user’s input to it:

```
$test = "30cm";
```

As you can see, the user has added units to his number—instead of entering “30”, the user has entered “30cm”. You can make sure that the user input is clean by casting it as an integer:

```
$newtest = (integer) $test;
echo "Your imaginary box has a width of $newtest cent
```

The resulting output would be

```
Your imaginary box has a width of 30 centimeters.
```

Had the user input not been cast, and the value of the original variable, $test, been used in place of $newtest when printing the statement regarding the width of a box, the result would have been

```
Your imaginary box has a width of 30cm centimeters.
```

This output looks strange; in fact, it looks like parroted user input that hadn’t been cleaned up (which is exactly what it is).

Why Test Type?

the most common questions we hear is, “Which programming language is Important to learn first?”

how to Become a Facebook Data Engineer? just Start Here.

With over 2.7 billion active monthly users, Facebook is an endless source of big data. if you have

what is involved with studying for the CompTIA A Plus certification exam. The CompTIA A+ certification exam has been..

Why might it be useful to know the type of a variable? There are often circumstances in programming in which data is passed to you from another source. In Chapter 7, you will learn how to create functions in your scripts, and data is often passed between one or more functions because they can accept information from calling code in the form of arguments. For the function to work with the data it is given, it is a good idea to first verify that the function has been given values of the correct data type. For example, a function expecting data that has a type of "resource" will not work well when passed a string.

Operators and Expressions

With what you have learned so far, you can assign data to variables, and you can even investigate and change the data type of a variable. A programming language isn't very useful, though, unless you can manipulate the data you have stored. *Operators* are symbols used to manipulate data stored in variables, to make it possible to use one or more values to produce a new value, or to check the validity of data to determine the next step in a condition, and so forth. A value operated on by an operator is referred to as an *operand*.

By the Way

An *operator* is a symbol or series of symbols that, when used in conjunction with values, performs an action and usually produces a new value.

An *operand* is a value used in conjunction with an operator. There are usually two or more operands to one operator.

In this simple example, two operands are combined with an operator to produce a new value:

```
(4 + 5)
```

The integers 4 and 5 are operands. The addition operator (+) operates on these operands to produce the integer 9. Operators almost always sit between two operands, although you will see a few exceptions later in this chapter.

The combination of operands with an operator to produce a result is called an *expression*. Although operators and their operands form the basis of expressions, an expression need not contain an operator. In fact, an expression in PHP is defined as anything that can be used as a value. This includes integer constants such as 654, variables such as $user, and function calls such as gettype(). The expression (4 + 5), for example, consists of two expressions (4 and 5) and an operator (+). When an expression produces a value, it is often said to *resolve to* that value. That is, when all subexpressions are taken into account, the expression can be treated as if it were a code for the value itself. In this case, the expression (4 + 5) resolves to 9.

By the Way

An *expression* is any combination of functions, values, and operators that resolves to a value. As a rule of thumb, if you can use it as if it were a value, it is an expression.

Now that you have the principles out of the way, it's time to take a tour of the operators commonly used in PHP programming.

give yourself all the best chances in succeeding.

7 skills employers of the future will be looking for

Futurepro of your CV by learning about the skills employers of tomorrow will most likely be looking for, and the online courses

The Assignment Operator

You have seen the assignment operator in use each time a variable was declared in an example; the assignment operator consists of the single character: =. The assignment operator takes the value of the right-side operand and assigns it to the left-side operand:

```
$name = "jimbo";
```

The variable $name now contains the string “jimbo”. This construct is also an expression. Although it might seem at first glance that the assignment operator simply changes the variable $name without producing a value, in fact, a statement that uses the assignment operator always resolves to a copy of the value of the right operand. Thus

```
echo $name = "jimbo";
```

prints the string “jimbo” to the browser while it also assigns the value “jimbo” to the $name variable.

Arithmetic Operators

The arithmetic operators do exactly what you would expect—they perform arithmetic operations. Table 5.2 lists these operators along with examples of their usage and results.

Table 5.2. Arithmetic Operators

Operator	Name	Example	Sample Result
+	Addition	10+3	13
−	Subtraction	10−3	7
/	Division	10/3	3.3333333333333
*	Multiplication	10*3	30
%	Modulus	10%3	1

you can take to train up.

Data science interview preparation: How to answer top questions

When it comes to interview preparation, for data science technical savviness and communication skills are of equal importance.

Posts

The addition operator adds the right-side operand to the left-side operand. The subtraction operator subtracts the right-side operand from the left-side operand. The division operator divides the left-side operand by the right-side operand. The multiplication operator multiplies the left-side operand by the right-side operand. The modulus operator returns the remainder of the left-side operand divided by the right-side operand.

The Concatenation Operator

The concatenation operator is represented by a single period (.). Treating both operands as strings, this operator appends the right-side operand to the left-side operand. So

```
"hello"." world"
```

returns

```
"hello world"
```

Note that the resulting space between the words occurs because there is a leading space in the second operand (” world” instead of “world”). The concatenation operator literally smashes together two strings without adding any padding. So, if you tried to concatenate two strings without leading or trailing spaces, such as

```
"hello"."world"
```

you would get this as your result:

```
"helloworld"
```

Regardless of the data types of the operands used with the concatenation operator, they are treated as strings, and the result will always be of the string type. You will

CISCO

CCNA 200-301 (Cisco Certified Network Associate) – Complete

CCNA 200-301 (Cisco Certified Network Associate) – Complete The Cisco Certified Network Associate (CCNA 200-301) exam is a 120-minute…

What’s new in

Featured

How HADOOP is useful to data scientists

Hadoop allows the users to store all forms of data, and provides massive storage for any kind of data.…

How to whizz Your Next Remote Interview

Remote interviews

encounter concatenation frequently throughout this book when the results of an expression of some kind must be combined with a string, as in

```
$cm = 212;
echo "the width is ".($cm/100)." meters";
```

Combined Assignment Operators

Although there is only one true assignment operator, PHP provides a number of combination operators that transform the left-side operand and return a result, while also modifying the original value of the variable. As a rule, operators use operands but do not change their original values, but combined assignment operators break this rule. A combined assignment operator consists of a standard operator symbol followed by an equal sign. Combination assignment operators save you the trouble of using two operators in two different steps within your script. For example, if you have a variable with a value of 4, and you want to increase this value to 4 more, you might see:

```
$x = 4;
$x = $x + 4; // $x now equals 8
```

However, you can also use a combination assignment operator (+=) to add and return the new value, as shown here:

```
$x = 4;
$x += 4; // $x now equals 8
```

Each arithmetic operator, as well as the concatenation operator, also has a corresponding combination assignment operator. Table 5.3 lists these new operators and shows an example of their usage.

Table 5.3. Some Combined Assignment Operators

Operator	Example	Equivalent To
+=	$x += 5	$x = $x + 5
−=	$x −= 5	$x = $x − 5
/=	$x /= 5	$x = $x / 5
*=	$x *= 5	$x = $x * 5
%=	$x %= 5	$x = $x % 5
.=	$x .= " test"	$x = $x." test"

Each of the examples in Table 5.3 transforms the value of $x using the value of the right-side operand. Subsequent uses of $x will refer to the new value. For example

```
$x = 4;
$x += 4; // $x now equals 8
$x += 4; // $x now equals 12
$x -= 3; // $x now equals 9
```

These operators will be used throughout the scripts in the book. You will frequently see the combined concatenation assignment operator when you begin to create dynamic text; looping through a script and adding content to a string, such as dynamically building the HTML code to represent a table, is a prime example of the use of a combined assignment operator.

Automatically Incrementing and Decrementing an Integer Variable

When coding in PHP, you will often find it necessary to increment or decrement a variable that is an integer type. You will usually need to do this when you are counting the iterations of a loop. You have already learned two ways of

playing a crucial part in today's job market. Find out all the most important remote interview tips to give yourself all the best chances in succeeding.

CCNP Data Center 300-610 [DCID] Describing High Availability on Layer 2

browser interacts with a web server, request/response cycle, including GET/POST/Redirect. an introductory understanding of Hypertext Markup Language (HTML), as well as the basic syntax and data structures of the PHP language, variables, logic, iteration, arrays, error handling, and doing this—either by incrementing the value of $x using the addition operator

```
$x = $x + 1; // $x is incremented by 1
```

or by using a combined assignment operator

```
$x += 1; // $x is incremented by 1
```

In both cases, the new value is assigned to $x. Because expressions of this kind are common, PHP provides some special operators that allow you to add or subtract the integer constant 1 from an integer variable, assigning the result to the variable itself. These are known as the *post-increment* and *post-decrement* operators. The post-increment operator consists of two plus symbols appended to a variable name:

```
$x++; // $x is incremented by 1
```

This expression increments the value represented by the variable $x by one. Using two minus symbols in the same way will decrement the variable:

```
$x--; // $x is decremented by 1
```

If you use the post-increment or post-decrement operators in conjunction with a conditional operator, the operand will be modified only after the first operation has finished:

```
$x = 3;
$y = $x++ + 3;
```

In this instance, $y first becomes 6 (the result of 3 + 3) and then $x is incremented.

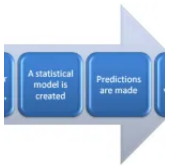
In some circumstances, you might want to increment or decrement a variable in a test expression before the test

In this Learning session, we'll be looking at features of Cisco network platforms that provide for high availability, in this case, high availability at layer 2.

Cisco CCNA practice test: Try these 20 exam questions

Use this CCNA latest practice test as study

superglobal variables, among other elements



Modeling Techniques in Predictive Analytics: Analytics and Data Science

Thomas W. Miller While earning a degree in philosophy may not be the best career move (unless a student…

Master of Applied

is carried out. PHP provides the pre-increment and pre-decrement operators for this purpose. These operators behave in the same way as the post-increment and post-decrement operators, but they are written with the plus or minus symbols preceding the variable:

```
++$x; // $x is incremented by 1
--$x; // $x is decremented by 1
```

If these operators are used as part of a test expression, incrementing occurs before the test is carried out. For example, in the next fragment, $x is incremented before it is tested against 4.

```
$x = 3;
++$x < 4; // false
```

The test expression returns false because 4 is not smaller than 4.

Comparison Operators

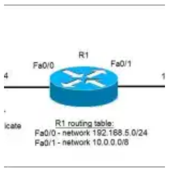
Comparison operators perform comparative tests using their operands and return the Boolean value true if the test is successful or false if the test fails. This type of expression is useful when using control structures in your scripts, such as if and while statements. This book covers if and while statements in Chapter 6, “Flow Control Functions in PHP.”

For example, to test whether the value contained in $x is smaller than 5, you can use the less-than operator as part of your expression:

```
$x < 5
```

If $x contains the value 3, this expression will have the value true. If $x contains 7, the expression resolves to false.

material to prepare for the Cisco CCNA Routing and Switching exam. With 20 questions …

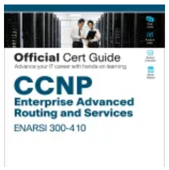


IP Routing in the LAN

Almost all enterprise networks use VLANs. To route IP packets in and out of those VLANs, some devices (either

Data Science

University of Michigan Online Master Degree Learn from the #1 public research university in the U.S. and join the next generation of data scientists. Completely online.



Official Cert Guide

CCNP Enterprise Advanced Routing and Services

ENARSI 300-410

CCNP Enterprise Advanced Routing ENARSI

Table 5.4 lists the comparison operators.

Table 5.4. Comparison Operators

Operator	Name	Returns True If…	Example ($x Is 4)	Result
==	Equivalence	Left is equivalent to right	$x == 5	false
!=	Non-equivalence	Left is not equivalent to right	$x != 5	true
===	Identical	Left is equivalent to right and they are the same type	$x === 4	true
	Non-equivalence	Left is equivalent to right but they are not the same type	$x === “4”	false
>	Greater than	Left is greater than right	$x > 4	false
>=	Greater than or equal to	Left is greater than or equal to right	$x >= 4	true
<	Less than	Left is less than right	$x < 4	false

routers or Layer 3 switches) need to have an IP address in each subnet and have a connected route to each of those subnets. Then the IP addresses on those routers or Layer 3 switches can serve as the default gateways in those subnets.

300-410 Official Cert Guide

CCNP Enterprise Advanced Routing ENARSI 300-410 Official Cert Guide. Published Mar 19, 2020 by Cisco Press. Part of the Official Cert Guide series.

Operator	Name	Returns True If…	Example ($x Is 4)	Result
<=	Less than or equal to	Left is less than or equal to right	$x <= 4	true

These operators are most commonly used with integers or doubles, although the equivalence operator is also used to compare strings. Be very sure to understand the difference between the == and = operators. The == operator tests equivalence, whereas the = operator assigns value. Also, remember that === tests equivalence with regards to both value and type.

Creating Complex Test Expressions with the Logical Operators

Logical operators test combinations of Boolean values. For example, the or operator, which is indicated by two pipe characters (||) or simply the word or, returns the Boolean value true if either the left or the right operand is true:

```
true || false
```

This expression returns true.

The and operator, which is indicated by two ampersand characters (&&) or simply the word and, returns the Boolean value true only if both the left and right operands are true:

```
true && false
```

This expression returns the Boolean value false. It's unlikely that you will use a logical operator to test

Boolean constants because it makes more sense to test two or more expressions that resolve to a Boolean. For example

```
($x > 2) && ($x < 15)
```

returns the Boolean value true if $x contains a value that is greater than 2 and smaller than 15. Parentheses are used when comparing expressions to make the code easier to read and to indicate the precedence of expression evaluation. Table 5.5 lists the logical operators.

Table 5.5. Logical Operators

Operator	**Name**	**Returns True If…**	**Example**	Result
||	Or	Left or right is true	true || false	true
or	Or	Left or right is true	true or false	true
xor	Xor	Left or right is true but not both	true xor true	false
&&	And	Left and right are true	true && false	false
and	And	Left and right are true	true and false	false
!	Not	The single operand is not true	! true	false

You might wonder why are there two versions of both the or and the and operators, and that’s a good question.

The answer lies in operator precedence, which you will examine next.

Operator Precedence

When you use an operator within an expression, the PHP engine usually reads your expression from left to right. For complex expressions that use more than one operator, though, the PHP engine could be led astray without some guidance. First, consider a simple case:

```
4 + 5
```

There's no room for confusion here—PHP simply adds 4 to 5. But what about the following fragment, with two operators:

```
4 + 5 * 2
```

This presents a problem. Should PHP find the sum of 4 and 5, and then multiply it by 2, providing the result 18? Or does it mean 4 plus the result of 5 multiplied by 2, resolving to 14? If you were simply to read from left to right, the former would be true. However, PHP attaches different precedence to different operators, and because the multiplication operator has higher precedence than the addition operator, the second solution to the problem is the correct one: 4 plus the result of 5 multiplied by 2.

However, you can override operator precedence by putting parentheses around your expressions. In the following fragment, the addition expression will be evaluated before the multiplication expression:

```
(4 + 5) * 2
```

Whatever the precedence of the operators in a complex expression, it is a good idea to use parentheses to make your code clearer and to save you from bugs such as applying sales tax to the wrong subtotal in a shopping cart situation. The following is a list of the operators covered in this chapter in precedence order (those with highest precedence are listed first):

- ++, –, (*cast*)
- /, *, %
- +, –
- <, <=, =>, >
- ==, ===, !=
- &&
- ||
- =, +=, -=, /=, *=, %=, .=
- and
- xor
- or

As you can see, or has a lower precedence than ||, and and has a lower precedence than &&, so you can use the lower-precedence logical operators to change the way a complex test expression is read. In the following fragment, the two expressions are equivalent, but the second is much easier to read:

```
$x and $y || $z
```

```
$x && ($y || $z)
```

Taking it one step further, the following fragment is easier still:

```
$x and ($y or $z)
```

However, all three examples are equivalent.

The order of precedence is the only reason that both && and and are available in PHP. The same is true of || and or. In most circumstances, the use of parentheses makes for clearer code and fewer bugs than code that takes advantage of the difference in the precedence of these operators. This book will tend to use the more common || and && operators and rely on parenthetical statements to set specific operator precedence.

Along with this basic PHP coding skills,

SKILLS YOU NEED TO GAIN Building Web Applications in PHP

Hypertext Preprocessor (PHP)

Html

Cascading Style Sheets (CCS)

Expanding each of them,

Introduction to Dynamic Web Content: the basic structure of a web application and how a web browser interacts with a web server. We explore the Request-Response Cycle that is the basis of the Hypertext Transfer Protocol (HTTP).

HyperText Markup Language (HTML) : the basics of the HyperText Markup Language (HTML) that is the markup for web pages.

Cascading Style Sheets (CSS) : the basics of cascading Style Sheets (CSS) that allow us to style the markup for web pages.

Installing PHP and SQL : The first technical task is to work through the installation steps including installing a text editor, installing MAMP or XAMPP (or equivalent), creating a MySql Database, and writing a PHP program.

Introduction to PHP : We begin learning PHP

PHP Arrays : unique aspects of arrays in the PHP language.

PHP Functions : unique aspects of functions in PHP.

PHP and HTML Forms : how HTML forms are created and processed in the PHP language.

Introducing to a course for Building Web Applications in PHP By University of Michigan

This course is part of the Web Applications for Everybody Specialization

A course by **Charles Russell Severance** and having 4.8 ratings and 458 reviews!

In this course, you'll explore the basic structure of a web application, and how a web browser interacts with a web server. You'll be introduced to the request/response cycle, including GET/POST/Redirect. You'll also gain an introductory understanding of Hypertext Markup Language (HTML), as well as the basic syntax and data structures of the PHP language, variables, logic, iteration, arrays, error handling, and superglobal variables, among other elements. An introduction to Cascading Style Sheets (CSS) will allow you to style markup for webpages. Lastly, you'll gain the skills and knowledge to install and use an integrated PHP/MySQL environment like XAMPP or MAMP.

And along with that, you will learn everything that explained earlier in this class above.

University of Michigan

The mission of the University of Michigan is to serve the people of Michigan and the world through preeminence in creating, communicating, preserving and applying knowledge, art, and academic values, and in developing leaders and citizens who will challenge the present and enrich the future.

Enrollment options

you can Enroll this course for FREE and can learn totally free for next 7 days and if you feel it is worth to enhance your knowledge and career, you can go further for getting the certification.

Start FREE Learning Today

for 7 days.

Financial aid available

- This Course Plus the Full Specialization
- Shareable Certificates
- Self-Paced Learning Option

- Course Videos & Readings
- Practice Quizzes
- Graded Assignments with Peer Feedback
- Graded Quizzes with Feedback
- Graded Programming Assignments

Enroll for Free
Starts Jul 08

63,451 already enrolled already.

Let us continue the Free lesson here giving you more Idea about the course.

PHP Constants

Variables offer a flexible way of storing data because you can change their values and the type of data they store at any time during the execution of your scripts. However, if you want to work with a value that must remain unchanged throughout your script's execution, you can define and use a *constant*. You must use PHP's built-in define() function to create a constant, which subsequently cannot be changed unless you specifically define() it again. To use the define() function, place the name of the constant and the value you want to give it, within parentheses and separated by a comma:

```
define("YOUR_CONSTANT_NAME", 42);
```

The value you want to set can be a number, a string, or a Boolean. By convention, the name of the constant should be in capital letters. Constants are accessed with the constant name only; no dollar symbol is required. Listing 5.4 shows you how to define and access a constant.

Listing 5.4. Defining and Accessing a Constant

```
1: <?php
2: define("THE_YEAR", "2008");
3: echo "It is the year ".THE_YEAR;
4: ?>
```

Did You Know

Constants can be used anywhere in your scripts, including in functions stored in external files.

Notice that in line 3 the concatenation operator is used to append the value held by the constant to the string “It is the year ” because PHP does not distinguish between a constant and a string within quotation marks.

Put these few lines into a text file called constant.php and place this file in your web server document root. When you access this script through your web browser, it produces the following output:

```
It is the year 2008
```

The define() function can also accept a third Boolean argument that determines whether the constant name should be case sensitive. By default, constant names are case sensitive. However, by passing true to the define() function, you can change this behavior, so if you were to set up our THE_YEAR constant as

```
define("THE_YEAR", "2008", true);
```

you could access its value without worrying about case:

```
echo the_year;
echo ThE_YeAr;
echo THE_YEAR;
```

The preceding three expressions are equivalent, and all would result in an output of 2008. This feature can make scripts a little friendlier for other programmers who work with our code because they will not need to consider case when accessing a constant we have already defined. On the other hand, given the fact that other constants *are* case sensitive, this might make for more, rather than less, confusion as programmers forget which constants to treat in which way. Unless you have a compelling reason to do otherwise, the safest course is to keep your constants case sensitive and define them using uppercase characters, which is an easy-to-remember (not to mention standard) convention.

Predefined Constants

PHP automatically provides some built-in constants for you. For example, the constant __FILE__ returns the name of the file that the PHP engine is currently reading. The constant __LINE__ returns the current line number of the file. These constants are useful for generating error messages. You can also find out which version of PHP is interpreting the script with the PHP_VERSION constant. This constant can be useful if you need version information included in script output when sending a bug report.

Summary

This chapter covered some of the basic features of the PHP language. You learned about variables and how to assign values to them using the assignment operator, as well as received an introduction to the scope of variables and built-in superglobals. You got an introduction to operators and learned how to combine some of the most common of these into expressions. Finally, you learned how to define and access constants.

Now that you have mastered some of the fundamentals of PHP, the next chapter will really put you in the driver's seat. You will learn how to make scripts that can make decisions and repeat tasks, with help from variables, expressions, and operators.

Q&A

Q.	***Why is it useful to know the type of data that a variable holds?***
A.	Often the data type of a variable constrains what you can do with it. For example, you can't perform array-related functions on simple strings. Similarly, you might want to make sure that a variable contains an integer or a float before using it in a mathematical calculation, even though PHP will often help you by changing data types for you in this situation.
Q.	***Should I obey any conventions when naming variables?***
A.	Your goal should always be to make your code easy to read and understand. A variable such as $ab123245 tells you nothing about its role in your script and invites typos. Keep your variable names short and descriptive.A variable named $f is unlikely to mean much to you when you return to your code after a month or so. A variable named $filename, on the other hand, should make more sense.
Q.	***Should I learn the operator precedence table?***
A.	There is no reason you shouldn't, but I would save the effort for more useful tasks. By using parentheses in your expressions, you can make your code easy to read while defining your own order of precedence.

Workshop

The workshop is designed to help you anticipate possible questions, review what you've learned, and begin putting your knowledge into practice.

Quiz

1.	Which of the following variable names are not valid? $a_value_submitted_by_a_user $666666xyz $xyz666666 $_____counter_____ $the first $file-name
2.	What will the following code fragment output?$num = 33; (boolean) $num; echo $num;
3.	What will the following statement output?echo gettype("4");
4.	What will be the output from the following code fragment?$test_val = 5.5466; settype($test_val, "integer"); echo $test_val;
5.	Which of the following statements does not contain an expression?4; gettype(44); 5/12;
6.	Which of the statements in question 5 contains an operator?
7.	What value will the following expression return?5 < 2What data type will the returned value be?

Answers

1.	The variable name $666666xyz is not valid because it does not begin with a letter or an underscore character. The variable name $the first is not valid because it contains a space. $file-name is also invalid because it contains a nonalphanumeric character (-).
2.	The fragment will print the integer 33. The cast to Boolean produces a converted copy of the value stored in $num. It does not alter the value actually stored there.
3.	The statement will output the string "string".
4.	The code will output the value 5. When a float is

	converted to an integer, any information beyond the decimal point is lost.
5.	They are all expressions because they all resolve to values.
6.	The statement 5/12; contains a division operator.
7.	The expression will resolve to false, which is a Boolean value.

Activities

1.	Create a script that contains at least five different variables. Populate them with values of different data types and use the gettype() function to print each type to the browser.
2.	Assign values to two variables. Use comparison operators to test whether the first value isThe same as the secondLess than the secondGreater than the secondLess than or equal to the secondPrint the result of each test to the browser.Change the values assigned to your test variables and run the script again.

Enroll for Free
Starts Jul 08

More Titles to Refer for Learning & Building Web Applications in PHP

Leave a comment Building Web Applications in PHP

Read More Posts

How HADOOP is useful to data scientists

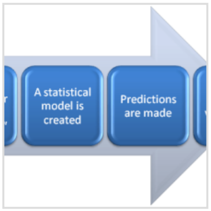
Hadoop allows the users to store all forms of data, and provides massive storage for any kind of data. It can handle a large amount of tasks, but here the…

Which programming language is Important to learn first?

Thinking about learning to code but not sure where to start? One of the most common questions we hear is, “Which programming language is Important to learn first?”

How to whizz Your Next Remote Interview

Remote interviews are now playing a crucial part in today’s job market. Find out all the most important remote interview tips to give yourself all the best chances in succeeding.

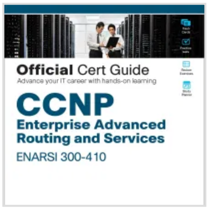


Modeling Techniques in Predictive Analytics: Analytics and Data Science

Thomas W. Miller While earning a degree in philosophy may not be the best career move (unless a student plans to teach philosophy, and few of these positions are available),…

Master of Applied Data Science

University of Michigan Online Master Degree Learn from the #1 public research university in the U.S. and join the next generation of data scientists. Completely online.


Official Cert Guide
Advance your IT career with hands-on learning
CCNP
Enterprise Advanced Routing and Services
ENARSI 300-410

CCNP Enterprise Advanced Routing ENARSI 300-410 Official Cert Guide

CCNP Enterprise Advanced Routing ENARSI 300-410 Official Cert Guide. Published Mar 19, 2020 by Cisco Press. Part of the Official Cert Guide series.

Leave a Reply

Logged in as success. Log out?

Comment

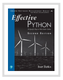
Post Comment

This site uses Akismet to reduce spam. Learn how your comment data is processed.

Recent Posts

- Udemy Budget Affordable Discount sale 2021
- Discovery Dojo: The School Of Getting To Know Yourself
- A Hands-On Guide to Splunk Enterprise Security
- Apache Kafka For Absolute Beginners
- Hadoop Developer Course with MapReduce and Java
- Google BigQuery & PostgreSQL : Big Query for Data Analysis

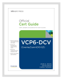
Latest Attractions


Effective Python

iOS Programming

Agile Game Development

DevOps for VMware Administrators


VCP6-DCV Official Cert Guide

Security+ SY0-401

design & shape world-class videos

design & shape world-class videos

Learn Ethical
Hacking -CEHv10
from scratch

Sitemap ○ Disclosure ○ Privacy Policy ○ TOS