

4

Using Numbers

Chapter 2, “Variables,” briefly discussed the various types of variables, how to assign values to them, and how they’re generally used. In this chapter, you’ll work specifically with number variables—both integers (whole numbers) and floating-point numbers (aka floats or decimals).

The chapter begins by creating an HTML form that will be used to generate number variables. Then you’ll learn how to perform basic arithmetic, how to format numbers, and how to cope with *operator precedence*. The last two sections of this chapter cover incrementing and decrementing numbers, plus generating random numbers. Throughout the chapter, you’ll also learn about other useful number-related PHP functions.

In This Chapter

Creating the Form	74
Performing Arithmetic	77
Formatting Numbers	81
Understanding Precedence	84
Incrementing and Decrementing a Number	86
Creating Random Numbers	88
Review and Pursue	90

Creating the Form

Most of the PHP examples in this chapter will perform various calculations based on an e-commerce premise. A form will take price, quantity, discount amount, tax rate, and shipping cost **A**, and the PHP script that handles the form will return a total cost. That cost will also be broken down by the number of payments the user wants to make in order to generate a monthly cost value **B**.

To start, let's create an HTML page that allows the user to enter the values.

To create the HTML form:

1. Begin a new HTML document in your text editor or IDE, to be named **calculator.php** (Script 4.1):

```
<!DOCTYPE html>
<html>

<head>

    <meta charset=utf-8"/>
<title>Product Cost Calculator
→ </title>

</head>
<body><!-- Script 4.1 -
→ calculator.php -->
<div><p>Fill out this form to
→ calculate the total cost:</p>
```

Fill out this form to calculate the total cost:

Price:

Quantity:

Discount:

Tax: (%)

Shipping method:

Number of payments to make:

A This form takes numbers from the user and sends them to a PHP page.

You have selected to purchase:
100 widget(s) at
\$5.00 price each plus a
\$5.00 shipping cost and a
7.5 percent tax rate.
After your **\$10.00** discount, the total cost is **\$532.13**.
Divided over **10** monthly payments, that would be **\$53.21** each.

B The PHP script performs a series of calculations on the submitted data and outputs the results. The results should look like this by the end of the chapter.

Script 4.1 This basic HTML form will be the origination of the numbers on which mathematical calculations will be performed in several PHP scripts.

```
1  <!DOCTYPE html>
2
3  <html>
4  <head>
5
6    <meta charset=utf-8"/>
7    <title>Product Cost Calculator
8    </title>
9
10   </head>
11   <body><!-- Script 4.1 -
12   calculator.html -->
13   <div><p>Fill out this form to
14   calculate the total cost:</p>
15
16   <form action="handle_calc.php"
17   method="post">
18
19   <p>Price: <input type="number"
20   name="price" size="5"></p>
21
22   <p>Quantity: <input type="number"
23   name="quantity" size="5" /></p>
24
25   <p>Discount: <input type="number"
26   name="discount" size="5"></p>
27
28   <p>Tax: <input type="number" name="tax"
29   size="3"> (%)</p>
30
31   <p>Shipping method: <select
32   name="shipping">
33   <option value="5.00">Slow and steady
34   </option>
35   <option value="8.95">Put a move on
36   it.</option>
37   <option value="19.36">I need it
38   yesterday!</option>
39   </select></p>
40
41   <p>Number of payments to make:
42   <input type="number"
43   name="payments" size="3" /></p>
44
45   <input type="submit" name="submit"
46   value="Calculate!" />
47
48   </form>
49
50   </div>
51   </body>
52   </html>
```

2. Create the initial **form** tag:

```
<form action="handle_calc.php"
→ method="post">
```

This **form** tag begins the HTML form. Its **action** attribute indicates that the form data will be submitted to a page named **handle_calc.php**. The tag's **method** attribute tells the page to use POST to send the data. See Chapter 3, "HTML Forms and PHP," for more details.

3. Create the inputs for the price, quantity, discount, and tax:

```
<p>Price: <input type="number"
→ name="price" size="5"></p>
<p>Quantity: <input
type="number" name="quantity"
size="5"></p> <p>Discount:
<input type="number"
→ name="discount" size="5"></p>
<p>Tax: <input type="number"
→ name="tax" size="3"> (%)</p>
```

HTML has input type for numbers, so create text boxes for these values. A parenthetical indicates that the tax should be entered as a percent.

Also remember that the names used for the inputs have to correspond to valid PHP variable names (letters, numbers, and the underscore only; doesn't start with a number, and so forth).

continues on next page

Input Type: number

The number type is used for input fields that should contain a numeric value.

You can also set restrictions on what numbers are accepted.

4. Add a field in which the user can select a shipping method:

```
<p>Shipping method: <select  
→ name="shipping">  
<option value="5.00">Slow and  
→ steady</option>  
<option value="8.95">Put a move  
→ on it.</option>  
<option value="19.36">I need it  
→ yesterday!</option>  
</select></p>
```

The shipping selection is made using a drop-down menu. The value of the selected option is the cost for that option. If the user selects, for example, the *Put a move on it.* option, the value of `$_POST['shipping']` in `handle_calc.php` will be 8.95.

5. Complete the HTML form:

```
<p>Number of payments to make:  
→ <input type="number"  
→ name="payments" size="3"></p>  
<input type="submit"  
name="submit" → value="Calculate!">  
</form>
```

The final two input types take a number for how many payments are required and then create a submit button (labeled *Calculate!*). The closing `form` tag marks the end of the form section of the page.

6. Complete the HTML page:

```
</div>  
</body>  
</html>
```

7. Save the script as `calculator.php` and view it in your Web browser.

Because this is an HTML page, you can view it directly in a Web browser.

Order of Operations - *PEMDAS (Please excuse my Dear Aunt Sally)*

Script 4.2 This PHP script executes all the standard mathematical calculations using the numbers submitted from the form.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset=utf-8">
5
6      <title>Product Cost Calculator</title>
7      <style type="text/css" media="screen">
8          .number { font-weight: bold; }
9      </style>
10 </head>
11 <body>
12 <?php // Script 4.2 - handle_calc.php
13 /* This script takes values from
14    calculator.html and performs
15    total cost and monthly payment
16    calculations. */
17
18 // Address error handling, if you want.
19 // Get the values from the $_POST array:
20 $price = $_POST['price'];
21 $quantity = $_POST['quantity'];
22 $discount = $_POST['discount'];
23 $tax = $_POST['tax'];
24 $shipping = $_POST['shipping'];
25 $payments = $_POST['payments'];
26
27 // Calculate the total:
28 $total = $price * $quantity;
29 $total = $total + $shipping;
30 $total = $total - $discount;
31
32 // Determine the tax rate:
33 $taxrate = $tax/100;
34 $taxrate = $taxrate + 1;
35
36 // Factor in the tax rate:
37 $total = $total * $taxrate;
38
39 // Calculate the monthly payments:
40 $monthly = $total / $payments;
```

code continues on next page

Performing Arithmetic

Just as you learned in grade school, basic mathematics involves the principles of addition, subtraction, multiplication, and division. These are performed in PHP using the most obvious operators:

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)

To use these operators, you'll create a PHP script that calculates the total cost for the sale of some widgets. This handling script could be the basis of a shopping cart application—a very practical Web page feature (although in this case the relevant number values will come from **calculator.html**).

When you're writing this script, be sure to note the use of comments (**Script 4.2**) to illuminate the different lines of code and the reasoning behind them.

To create your sales cost calculator:

1. Create a new document in your text editor or IDE, to be named **handle_calc.php** (Script 4.2):

```
<!DOCTYPE html>

<html>
<head>

    <meta charset=utf-8">
<title>Product Cost Calculator
→ </title>
```

continues on next page

```

<style type="text/css"
→ media="screen">
    .number { font-weight: bold; }
</style>
</head>
<body>

```

The head of the document defines one CSS class called *number*. Any element within the page that has that class value will be given extra font weight. In other words, when the numbers from the form, and the results of the various calculations, are printed in the script's output, they'll be made more obvious.

2. Insert the PHP tags and address error handling, if desired:

```

<?php // Script 4.2 -
→ handle_calc.php

```

Depending on your PHP configuration, you may or many not want to add a couple of lines that turn on **display_errors** and adjust the level of error reporting. See Chapter 3 for specifics.

3. Assign the `$_POST` elements to local variables:

```

$price = $_POST['price'];
$quantity = $_POST['quantity'];
$discount = $_POST['discount'];
$tax = $_POST['tax'];
$shipping = $_POST['shipping'];
$payments = $_POST['payments'];

```

The script will receive all the form data in the predefined `$_POST` variable. To access individual form values, refer to `$_POST['index']`, replacing *index* with the corresponding form element's *name* value.

These values are assigned to individual local variables here, to make it easier to use them throughout the rest of the script.

Note that each variable is given a descriptive name and is written entirely in lowercase letters.

Script 4.2 *continued*

```

41 // Print out the results:
42 print "<p>You have selected to
    purchase:<br>
43 <span class=\"number\">$quantity
    </span> widget(s) at <br>
44 $<span class=\"number\">$price</span>
    price each plus a <br>
45 $<span class=\"number\">$shipping
    </span> shipping cost and a <br> 46
    <span class=\"number\">$tax</span>
    percent tax rate.<br>
47 After your $<span class=\"number\">
    $discount</span> discount, the total
    cost is
48 $<span class=\"number\">$total
    </span>.<br>
49 Divided over <span class=\"number\">
    $payments</span> monthly payments,
    that would be $<span class=\"number\">
    $monthly</span> each.</p>";
50
51 ?>
52 </body>
53 </html>

```

4. Begin calculating the total cost:

```
$total = $price * $quantity;  
$total = $total + $shipping;  
$total = $total - $discount;
```

The asterisk (*) indicates multiplication in PHP, so the total is first calculated as the number of items purchased (**\$quantity**) multiplied by the price. Then the shipping cost is added to the total value (remember that the shipping cost correlates to the **value** attribute of each shipping drop-down menu's **option** tags), and the discount is subtracted.

Note that it's perfectly acceptable to determine a variable's value in part by using that variable's existing value (as is done in the last two lines).

5. Calculate the tax rate and the new total:

```
$taxrate = $tax/100;  
$taxrate = $taxrate + 1;  
$total = $total * $taxrate;
```

The tax rate should be entered as a percent—for example, 8 or 5.75. This number is then divided by 100 to get the decimal equivalent of the percent (.08 or .0575). Finally, you calculate how much something costs with tax by adding 1 to the percent and then multiplying that new rate by the total. This is the mathematical equivalent of multiplying the decimal tax rate times the total and then adding this result to the total (for example, a 5 percent tax on \$100 is \$5, making the total \$105, which is the same as multiplying \$100 times 1.05).

6. Calculate the monthly payment:

```
$monthly = $total / $payments;
```

As an example of division, assume that the widgets can be paid for over the course of many months. Hence, you divide the total by the number of payments to find the monthly payment.

7. Print the results:

```
print "<p>You have selected to  
→ purchase:<br />  
<span class=\"number\">$quantity  
→ </span> widget(s) at <br />  
$<span class=\"number\">$price  
→ </span> price each plus a <br />  
$<span class=\"number\">$shipping  
→ </span> shipping cost and a <br />  
<span class=\"number\">$tax</span>  
→ percent tax rate.<br />  
After your $<span class=\"number\">  
→ $discount</span> discount, the  
→ total cost is  
$<span class=\"number\">$total  
→ </span>.<br />  
Divided over <span class=\"number\">  
→ $payments</span> monthly  
→ payments, that would be $<span  
→ class=\"number\">$monthly</span>  
→ each.</p>";
```

The **print** statement sends every value to the Web browser along with some text. To make it easier to read, **
** tags are added to format the browser result; in addition, the **print** function operates over multiple lines to make the PHP code cleaner. Each variable's value will be highlighted in the browser by wrapping it within **span** tags that have a **class** attribute of **number** (see Step 1).

8. Close the PHP section and complete the HTML page:

```
?>  
</body>  
</html>
```

9. Save the script as **handle_calc.php** and place it in the proper directory for your PHP-enabled server.

Make sure that **calculator.php** is in the same directory.

continues on next page

10. Test the script in your Web browser by filling out **A** and submitting **B** the form.

Not to belabor the point, but make sure you start by loading the HTML form through a URL (*http://something*) so that when it's submitted, the PHP script is also run through a URL.

You can experiment with these values to see how effectively your calculator works. If you omit any values, the resulting message will just be a little odd but the calculations should still work **C**.

TIP As you'll certainly notice, the calculator comes up with numbers that don't correspond well to real dollar values (see **B** and **C**). In the next section, "Formatting Numbers," you'll learn how to address this issue.

TIP If you want to print the value of the total before tax or before the discount (or both), you can do so in two ways. You can insert the appropriate print statements immediately after the proper value has been determined but before the `$total` variable has been changed again. Or, you can use new variables to represent the values of the subsequent calculations (for example, `$total_with_tax` and `$total_less_discount`).

TIP Attempting to print a dollar sign followed by the value of a variable, such as `$10` (where 10 comes from a variable), has to be handled carefully. You can't use the syntax `$$variable`, because the combination of two dollar signs creates a type of variable that's too complex to discuss in this book. One solution is to put something—a space or an HTML tag, as in this example—between the dollar sign and the variable name. Another option is to escape the first dollar sign:

```
print "The total is \$$total";
```

A third option is to use concatenation, which is introduced in the next chapter.

Fill out this form to calculate the total cost:

Price:

Quantity:

Discount:

Tax: (%)

Shipping method:

Number of payments to make:

A The HTML form...

You have selected to purchase:
6 widget(s) at
\$19.95 price each plus a
\$5.00 shipping cost and a
6 percent tax rate.
After your \$10.00 discount, the total cost is \$121.582.
Divided over 12 monthly payments, that would be \$10.1318333333 each.

B ...and the resulting calculations.

You have selected to purchase:
6 widget(s) at
\$19.95 price each plus a
\$5.00 shipping cost and a
percent tax rate.
After your \$ discount, the total cost is \$124.7.
Divided over 12 monthly payments, that would be \$10.3916666667 each.

C You can omit or change any value and rerun the calculator. Here the tax and discount values have been omitted.

TIP This script performs differently, depending on whether the various fields are submitted. The only truly problematic field is the number of monthly payments: If this is omitted, you'll see a division-by-zero warning. Chapter 6, "Control Structures," will cover validating form data before it's used.

TIP HTML 5 is expected to have one or more inputs that restrict the user to entering numeric values.

php.net
Search for String
functions -->
number_format
Math --> math
functions --> round

Formatting Numbers

Although the calculator is on its way to being practical, it still has one legitimate problem: You can't ask someone to make a monthly payment of \$10.13183333! To create more usable numbers, you need to format them.

There are two appropriate functions for this purpose. The first, **round()**, rounds a value to a specified number of decimal places. The function's first argument is the number to be rounded. This can be either a number or a variable that has a numeric value. The second argument is optional; it represents the number of decimal places to which to round. If omitted, the number will be rounded to the nearest integer. For example:

```
round (4.30); // 4
round (4.289, 2); // 4.29
$num = 236.26985;
round ($num); // 236
```

The other function you can use in this situation is **number_format()**. It works like **round()** in that it takes a number (or a variable with a numeric value) and an optional decimal specifier. This function has the added benefit of formatting the number with commas, the way it would commonly be written:

```
number_format (428.4959, 2); // 428.50
number_format (428, 2); // 428.00
number_format (123456789);
→ // 123,456,789
```

Let's rewrite the PHP script to format the numbers appropriately.

To format numbers:

1. Open `handle_calc.php` in your text editor or IDE, if it is not already open (Script 4.2).
2. After all the calculations but before the **print** statement, add the following (Script 4.3):

```
$total = number_format ($total, 2);  
$monthly = number_format  
→ ($monthly, 2);
```

To format these two numbers, apply this function after every calculation has been made but before they're sent to the Web browser. The second argument (the 2) indicates that the resulting number should have exactly two decimal places; this setting rounds the numbers and adds zeros at the end, as necessary.

Script 4.3 The `number_format()` function is applied to the values of two number variables, so they are more appropriate.

```
1  <!DOCTYPE html">  
2  <html lang="en">  
3  <head>  
4  <meta charset=utf-8">  
5      <title>Product Cost Calculator</title>  
6      <style type="text/css" media="screen">  
7          .number { font-weight: bold;}  
8      </style>  
9  </head>  
10 <body>  
11 <?php // Script 4.3 - handle_calc.php #2  
12 /* This script takes values from  
13    calculator.php and performs  
14    total cost and monthly payment  
15    calculations. */  
16 // Address error handling, if you want.  
17  
18 // Get the values from the $_POST array:  
19 $price = $_POST['price'];  
20 $quantity = $_POST['quantity'];  
21 $discount = $_POST['discount'];  
22 $tax = $_POST['tax'];  
23 $shipping = $_POST['shipping'];  
24 $payments = $_POST['payments'];  
25  
26 // Calculate the total:  
27 $total = $price * $quantity;  
28 $total = $total + $shipping;  
29 $total = $total - $discount;  
30  
31 // Determine the tax rate:  
32 $taxrate = $tax/100;  
33 $taxrate = $taxrate + 1;  
34  
35 // Factor in the tax rate:  
36 $total = $total * $taxrate;  
37  
38 // Calculate the monthly payments:  
39 $monthly = $total / $payments;  
40  
41 // Apply the proper formatting:  
42 $total = number_format ($total, 2);  
43 $monthly = number_format ($monthly, 2);  
44
```

code continues on next page

Script 4.3 continued

```
45 // Print out the results:
46 print "<p>You have selected to
    purchase:<br>
47 <span class='number'>$quantity</span>
    widget(s) at <br>
48 $<span class='number'>$price</span>
    price each plus a <br>
49 $<span class='number'>$shipping</span>
    shipping cost and a <br>
50 <span class='number'>$tax</span> percent
    tax rate.<br />
51 After your $<span class='number'>
    $discount</span> discount, the total
    cost is
52 $<span class='number'>$total</span>.<br>
53 Divided over <span class='number'>
    $payments</span> monthly payments, that
    would be $<span class='number'>$monthly
    </span> each.</p>";
54
55 ?>
56 </body>
57 </html>
```

Fill out this form to calculate the total cost:

Price:

Quantity:

Discount:

Tax: (%)

Shipping method:

Number of payments to make:

A Another form entry.

You have selected to purchase:
4 widget(s) at
\$99.00 price each plus a
\$8.95 shipping cost and a
5.5 percent tax rate.
After your \$25.00 discount, the total cost is \$400.85.
Divided over 24 monthly payments, that would be \$16.70 each.

B The updated version of the script returns more appropriate number values thanks to the `number_format()` function.

3. Save the file, place it in the same directory as `calculator.html`, and test it in your browser **A** and **B**.

TIP Another, much more complex way to format numbers is to use the `printf()` and `sprintf()` functions. Because of their tricky syntax, they're not discussed in this book; see the PHP manual for more information.

TIP Non-Windows versions of PHP also have a `money_format()` function, which can be used in lieu of `number_format()`.

TIP For complicated reasons, the `round()` function rounds exact halves (.5, .05, .005, and so on) down half the time and up half the time.

TIP In PHP, function calls can have spaces between the function name and its parentheses or not. Both of these are fine:

```
round($num);
round($num);
```

TIP The `number_format()` function takes two other optional arguments that let you specify what characters to use to indicate a decimal point and break up thousands. This is useful, for example, for cultures that write 1,000.89 as 1.000,89. See the PHP manual for the correct syntax, if you want to use this option.

`sprintf()` returns a formatted string

Understanding Precedence

Inevitably, after a discussion of the various sorts of mathematical operators comes the discussion of precedence. **Precedence refers to the order in which a series of calculations are executed.** For example, what is the value of the following variable?

\$number = 10 - 4 / 2;

Is **\$number** worth 3 (10 minus 4 equals 6, divided by 2 equals 3) or 8 (4 divided by 2 equals 2, subtracted from 10 equals 8)? The answer here is 8, because division takes precedence over subtraction.

Appendix B, “Resources and Next Steps,” shows the complete list of operator precedence for PHP (including operators that haven’t been covered yet). However, instead of attempting to memorize a large table of peculiar characters, you can bypass the whole concept by using parentheses. **Parentheses always take precedence over any other operator.** Thus:

\$number = (10 - 4) / 2; // 3

\$number = 10 - (4 / 2); // 8

Using parentheses in your calculations ensures that you never see peculiar results due to precedence issues. Parentheses can also be used to rewrite complex calculations in fewer lines of code. Let’s rewrite the **handle_calc.php** script, combining multiple lines into one by using parentheses, while maintaining accuracy.

Script 4.4 By using parentheses, calculations made over multiple lines (see Script 4.3) can be condensed without affecting the script’s mathematical accuracy.

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/
    xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/
    xhtml" xml:lang="en" lang="en">
4  <head>
5      <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8"/>
6      <title>Product Cost Calculator</title>
7      <style type="text/css" media="screen">
8          .number { font-weight: bold;}
9      </style>
10 </head>
11 <body>
12 <?php // Script 4.4 - handle_calc.php #3
13 /* This script takes values from
    calculator.html and performs
14 total cost and monthly payment
    calculations. */
15
16 // Address error handling, if you want.
17
18 // Get the values from the $_POST array:
19 $price = $_POST['price'];
20 $quantity = $_POST['quantity'];
21 $discount = $_POST['discount'];
22 $tax = $_POST['tax'];
23 $shipping = $_POST['shipping'];
24 $payments = $_POST['payments'];
25
26 // Calculate the total:
27 $total = (($price * $quantity) +
    $shipping) - $discount;
28
29 // Determine the tax rate:
30 $taxrate = ($tax/100) + 1;
31
32 // Factor in the tax rate:
33 $total = $total * $taxrate;
34
35 // Calculate the monthly payments:
36 $monthly = $total / $payments;
37
38 // Apply the proper formatting:
39 $total = number_format ($total, 2);
40 $monthly = number_format ($monthly, 2);
41
```

code continues on next page

Script 4.4 continued

```
42 // Print out the results:
43 print "<p>You have selected to
44 purchase:<br>
45 <span class='number'>$quantity</span>
46 widget(s) at <br>
47 <span class='number'>$price</span>
48 price each plus a <br>
49 <span class='number'>$shipping</span>
50 shipping cost and a <br>
51 <span class='number'>$tax</span> percent
52 tax rate.<br>
53 After your <span class='number'>$discount
54 </span> discount, the total cost is
55 <span class='number'>$total</span>.<br>
56 Divided over <span class='number'>
57 $payments</span> monthly payments, that
58 would be <span class='number'>$monthly
59 </span> each.</p>";
60
61 ?>
62 </body>
63 </html>
```

Fill out this form to calculate the total cost:

Price:

Quantity:

Discount:

Tax: (%)

Shipping method:

Number of payments to make:

A Testing the form one more time.

You have selected to purchase:
250 widget(s) at
\$1.50 price each plus a
\$19.36 shipping cost and a
6 percent tax rate.
After your \$0 discount, the total cost is \$418.02.
Divided over 2 monthly payments, that would be \$209.01 each.

B Even though the calculations have been condensed, the math works out the same. If you see different results or get an error message, double-check your parentheses for balance (an equal number of opening and closing parentheses).

To manage precedence:

1. Open **handle_calc.php** in your text editor or IDE, if it is not already open (Script 4.3).
2. Replace the three lines that initially calculate the order total with the following (Script 4.4):

```
$total = (($price * $quantity) +  
→ $shipping) - $discount;
```

There's no reason not to make all the calculations in one step, as long as you use parentheses to ensure that the math works properly. The other option is to memorize PHP's rules of precedence for multiple operators, but using parentheses is a lot easier.

3. Change the two lines that calculate and add in the tax to this:

```
$taxrate = ($tax/100) + 1;
```

Again, the tax calculations can be made in one line instead of two separate ones.

4. Save the script, place it in the same directory as **calculator.html**, and test it in your browser **A** and **B**.

TIP Be sure that you match your parentheses consistently as you create your formulas (every opening parenthesis requires a closing parenthesis). Failure to do so will cause parse errors.

TIP Granted, using the methods applied here, you could combine all the total calculations into just one line of code (instead of three)—but there is such a thing as oversimplifying.

Incrementing and Decrementing a Number

PHP, like Perl and most other programming languages, includes some shortcuts that let you avoid ugly constructs such as

```
$tax = $tax + 1;
```

When you need to increase the value of a variable by 1 (called an *incremental* adjustment) or decrease the value of a variable by 1 (a *decremental* adjustment), you can use **++** and **--**, respectively:

```
$var = 20; // 20
```

```
$var++; // 21
```

```
$var++; // 22
```

```
$var--; // 21
```

Solely for the sake of testing this concept, you'll rewrite the **handle_calc.php** script one last time.

To increment the value of a variable:

1. Open **handle_calc.php** in your text editor or IDE, if it is not already open (Script 4.4).
2. Change the tax rate calculation from Script 4.3 to read as follows (Script 4.5):

```
$taxrate = $tax/100;
```

```
$taxrate++;
```

The first line calculates the tax rate as the **\$tax** value divided by 100. The second line increments this value by 1 so that it can be multiplied by the total to determine the total with tax.

Script 4.5 Incrementing or decrementing a number is a common operation using **++** or **--**, respectively.

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/
        xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/
        xhtml" xml:lang="en" lang="en">
4  <head>
5      <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8"/>
6      <title>Product Cost Calculator
        </title>
7      <style type="text/css" media="screen">
8          .number { font-weight: bold;}
9      </style>
10 </head>
11 <body>
12 <?php // Script 4.3 - handle_calc.php #4
13 /* This script takes values from
        calculator.html and performs
14 total cost and monthly payment
        calculations. */
15
16 // Address error handling, if you want.
17
18 // Get the values from the $_POST array:
19 $price = $_POST['price'];
20 $quantity = $_POST['quantity'];
21 $discount = $_POST['discount'];
22 $tax = $_POST['tax'];
23 $shipping = $_POST['shipping'];
24 $payments = $_POST['payments'];
25
26 // Calculate the total:
27 $total = (($price * $quantity) +
        $shipping) - $discount;
28
29 // Determine the tax rate:
30 $taxrate = $tax/100;
31 $taxrate++;
32
33 // Factor in the tax rate:
34 $total = $total * $taxrate;
35
36 // Calculate the monthly payments:
37 $monthly = $total / $payments;
38
39 // Apply the proper formatting:
40 $total = number_format ($total, 2);
41 $monthly = number_format ($monthly, 2);
42
```

code continues on next page

Script 4.5 *continued*

```
43 // Print out the results:
44 print "<p>You have selected to
    purchase:<br />
45 <span class='number'>$quantity</span>
    widget(s) at <br />
46 <span class='number'>$price</span>
    price each plus a <br />
47 <span class='number'>$shipping</span>
    shipping cost and a <br />
48 <span class='number'>$tax</span>
    percent tax rate.<br />
49 After your <span class='number'>
    $discount</span> discount, the total
    cost is
50 <span class='number'>$total</span>.<br />
51 Divided over <span class='number'>
    $payments</span> monthly payments, that
    would be <span class='number'>
    $monthly</span> each.</p>";
52
53 ?>
54 </body>
55 </html>
```

Fill out this form to calculate the total cost:

Price:

Quantity:

Discount:

Tax: (%)

Shipping method:

Number of payments to make:

A The last execution of the form.

You have selected to purchase:
100 widget(s) at
\$5.00 price each plus a
\$5.00 shipping cost and a
7.5 percent tax rate.
After your **\$10.00** discount, the total cost is **\$532.13**.
Divided over **10** monthly payments, that would be **\$53.21** each.

B It won't affect your calculations if you use the long or short version of incrementing a variable (compare Scripts 4.4 and 4.5).

3. Save the script, place it in the same directory as **calculator.html**, and test it in your browser **A** and **B**.

TIP Although functionally it doesn't matter whether you code `$taxrate = $taxrate + 1;` or the abbreviated `$taxrate++`, the latter method (using the increment operator) is more professional and common.

TIP In Chapter 6, "Control Structures," you'll see how the increment operator is commonly used in conjunction with loops.

Arithmetic Assignment Operators

PHP also supports a combination of mathematical and assignment operators. These are `+=`, `-=`, `*=`, and `/=`. Each will assign a value to a variable by performing a calculation on it. For example, these next two lines both add 5 to a variable:

```
$num = $num + 5;
$num += 5;
```

This means the **handle_calc.php** script could determine the tax rate using this:

```
$tax = $_POST['tax']; // Say, 5
$tax /= 100; // Now $tax is .05
$tax += 1; // 1.05
```

You'll frequently see these shorthand ways of performing arithmetic.

Creating Random Numbers

The last function you'll learn about in this chapter is **rand()**, a random-number generator. All it does is output a random number:

```
$n = rand(); // 31
```

```
$n = rand(); // 87
```

The **rand()** function can also take minimum and maximum parameters, if you prefer to limit the generated number to a specific range:

```
$n = rand (0, 10);
```

These values are inclusive, so in this case 0 and 10 are feasible returned values.

As an example of generating random numbers, let's create a simple "Lucky Numbers" script.

To generate random numbers:

1. Begin a new document in your text editor or IDE, to be named **random.php** (Script 4.6):

```
<!DOCTYPE html>  
  
<html lang="en">  
<head>  
  
    <meta charset=utf-8"/>  
    <title>Lucky Numbers</  
title>  
  
</head>  
<body>
```

2. Include the PHP tags and address error management, if you need to:

```
<?php // Script 4.6 - random.php
```

Script 4.6 The **rand()** function generates a random number.

```
1  <!DOCTYPE html>  
2  <html lang="en">  
3  <head>  
4  <meta charset=utf-8">  
5  
6      <title>Lucky Numbers</title>  
7  </head>  
8  <body>  
9  <?php // Script 4.6 - random.php  
10 /* This script generates 3 random  
    numbers. */  
11  
12 // Address error handling, if you want.  
13  
14 // Create three random numbers:  
15 $n1 = rand (1, 99);  
16 $n2 = rand (1, 99);  
17 $n3 = rand (1, 99);  
18  
19 // Print out the numbers:  
20 print "<p>Your lucky numbers are:<br>  
21     $n1<br>  
22     $n2<br>  
23     $n3</p>";  
24  
25 ?>  
26 </body>  
27 </html>
```



```
Your lucky numbers are:  
32  
68  
71
```

A The three random numbers created by invoking the **rand()** function.

```
Your lucky numbers are:  
23  
81  
2
```

B Running the script again produces different results.

Other Mathematical Functions

PHP has a number of built-in functions for manipulating mathematical data. This chapter introduced **round()**, **number_format()**, and **rand()**.

PHP has broken **round()** into two other functions. The first, **ceil()**, rounds every number to the next highest integer. The second, **floor()**, rounds every number to the next lowest integer.

Another function the calculator page could make good use of is **abs()**, which returns the absolute value of a number. In case you don't remember your absolute values, the function works like this:

```
$number = abs(-23); // 23  
$number = abs(23); // 23
```

In layman's terms, the absolute value of a number is always a positive number.

Beyond these functions, PHP supports all the trigonometry, exponent, base conversion, and logarithm functions you'll ever need. See the PHP manual for more information.

3. Create three random numbers:

```
$n1 = rand (1, 99);  
$n2 = rand (1, 99);  
$n3 = rand (1, 99);
```

This script prints out a person's lucky numbers, like those found on the back of a fortune cookie. These numbers are generated by calling the **rand()** function three separate times and assigning each result to a different variable.

4. Print out the numbers:

```
print "<p>Your lucky  
numbers are:<br>  
$n1<br>  
$n2<br>  
$n3</p>";
```

The **print** statement is fairly simple. The numbers are printed, each on its own line, by using the HTML break tag.

5. Close the PHP code and the HTML page:

```
?>  
</body>  
</html>
```

6. Save the file as **random.php**, place it in the proper directory for your PHP-enabled server, and test it in your Web browser **A**. Refresh the page to see different numbers **B**.

TIP The **getrandmax()** function returns the largest possible random number that can be created using **rand()**. This value differs by operating system.

TIP PHP has another function that generates random numbers: **mt_rand()**. It works similarly to (but, arguably, better than) **rand()** and is the smarter choice for sensitive situations like cryptography. Also see the PHP manual's page for the **mt_rand()** function for more discussion of generating random numbers as a whole.

Review and Pursue

If you have any problems with the review questions or the pursue prompts, turn to the book's supporting forum (www.LarryUllman.com/forum/).

Review

- What are the four primary arithmetic operators?
- Why will the following code not work:

```
print "The total is $$total";
```


What must be done instead?
- Why must an HTML page that contains a form that's being submitted to a PHP script be loaded through a URL?
- What functions can be used to format numerical values? How do you format numbers to a specific number of decimals?
- What is the importance of operator precedence?
- What are the incremental and decremental operators?
- What are the arithmetic assignment operators?

Pursue

- Look up the PHP manual page for one of the new functions mentioned in this chapter. Use the links on that page to investigate a couple of other number-related functions PHP has.
- Create another HTML form for taking numeric values. Then create the PHP script that receives the form data, performs some calculations, formats the values, and prints the results.