CIS253 PHP

CHAPTER 1
"GETTING STARTED WITH PHP"

PHP PROGRAMMING WITH MYSQL
2ND EDITION

Objectives

In this chapter you will:

- Create PHP scripts
- Create PHP code blocks
- Work with variables and constants
- Study data types
- Use expressions and operators

Creating Basic PHP Scripts

- **Embedded language** refers to code that is embedded within a Web page
- PHP code is typed directly into a Web page as a separate section
- A Web page containing PHP code must be saved with an extension of .php to be processed by the scripting engine
- PHP code is never sent to a client's Web browser; only the output of the processing is sent to the browser

Creating Basic PHP Scripts (cont.)

- The Web page generated from the PHP code, and HTML elements found within the PHP file, is returned to the client
- A PHP file that does not contain any PHP code should be saved with an **.html** extension
- **.php** is the default extension that most Web servers use to process PHP scripts

Short Quiz, p. 3

1. Define the term "embedded language" as it applies to PHP.
2. Why should you avoid using the .php extension if the document contains only XHTML code?
3. Explain why you do not see any PHP code when you view the source code of a PHP page in a browser.

Creating PHP Code Blocks

- **Code declaration blocks** are separate sections on a Web page that are interpreted by the scripting engine
- There are four types of code declaration blocks:
  - Standard PHP script delimiters
  - The `<script>` element
  - Short PHP script delimiters
  - ASP-style script delimiters

Standard PHP Script Delimiters

- The standard method of writing PHP code declaration blocks is to use the `<?php` and `?>` script delimiters
- A **delimiter** is a character or sequence of characters used to mark the beginning and end of a code segment
- The individual lines of code that make up a PHP script are called **statements** – end all statements with a semicolon ;

The `<script>` Element

- The **`<script>` element** identifies a script section in a Web page document
- Assign a value of "php" to the **language** attribute of the `<script>` element to identify the code block as PHP
- The `<script>` element's language attribute is deprecated in XHTML.
- Further, the scripting engine ignores `<script>` elements that include the type attribute, which is required for compatibility with both the strict and transitional DTDs.

Short PHP Script Delimiters

- The syntax for the short PHP script delimiters is

  ```
  <? statements; ?>
  ```
- Short delimiters can be disabled in a Web server's php.ini configuration file
- PHP scripts will not work if your Web site ISP does not support short PHP script delimiters
- Short delimiters can be used in XHTML documents, but not in XML documents

ASP-Style Script Delimiters

- The syntax for the ASP-style script delimiters is

  `<% statements; %>`

- ASP-style script delimiters can be used in XHTML documents, but not in XML documents
- ASP-style script delimiters can be enabled or disabled in the php.ini configuration file
- To enable or disable ASP-style script delimiters, assign a value of “On” or “Off ” to the `asp_tags` directive in the php.ini configuration file

Understanding Functions

- A **function** is a subroutine (or individual statements grouped into a logical unit) that performs a specific task
  - To execute a function, you must invoke, or **call**, it from somewhere in the script
- A **function call** is the function name followed by any data that the function needs
- The data (in parentheses following the function name) are called **arguments** or **actual parameters**
- Sending data to a called function is called **passing arguments**

Displaying Script Results

- The `echo` and `print` statements are language constructs (built-in features of a programming language) that create new text on a Web page that is returned as a response to a client
- The `print` statement returns a value of `1` if successful or a value of `0` if not successful, while the `echo` statement does not return a value
- Use the `echo` and `print` statements to return the results of a PHP script within a Web page that is returned to a client

Displaying Script Results (cont.)

- The text passed to the `echo` statement is called a "**literal string**" and must be enclosed in either single or double quotation marks
- To pass multiple arguments to the `echo` statement, separate the statements with commas
- To format the output of text, you can use any XHTML formatting elements you want as part of the text string arguments

Creating Multiple Code Declaration Blocks

- For multiple script sections in a document, include a separate code declaration block for each section

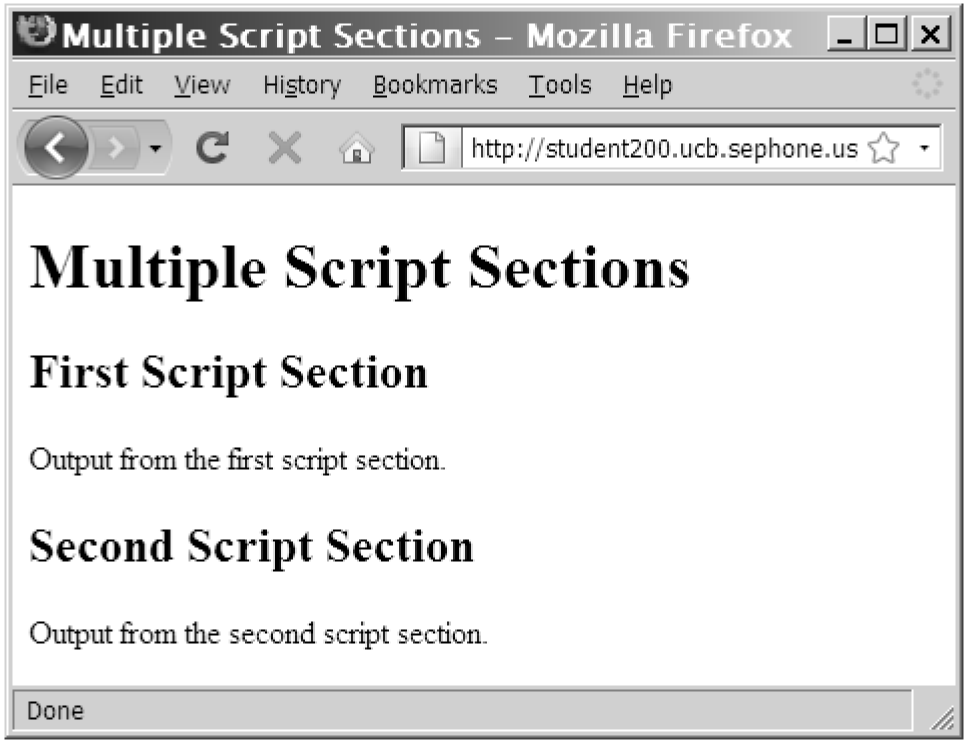
```
...
</head>
<body>
<h1>Multiple Script Sections</h1>
<h2>First Script Section</h2>
<?php echo "<p>Output from the first script section.</p>";
?>
<h2>Second Script Section</h2>
 <?php echo "<p>Output from the second script
 section.</p>";?>
</body>
</html>
```

Creating Multiple Code Declaration Blocks (cont.)

- PHP code declaration blocks execute on a Web server before a Web page is sent to a client

```
...
</head>
<body>
<h1>Multiple Script Sections</h1>
<h2>First Script Section</h2>
<p>Output from the first script section.</p>
<h2>Second Script Section</h2>
<p>Output from the second script section.</p>
</body>
</html>
```

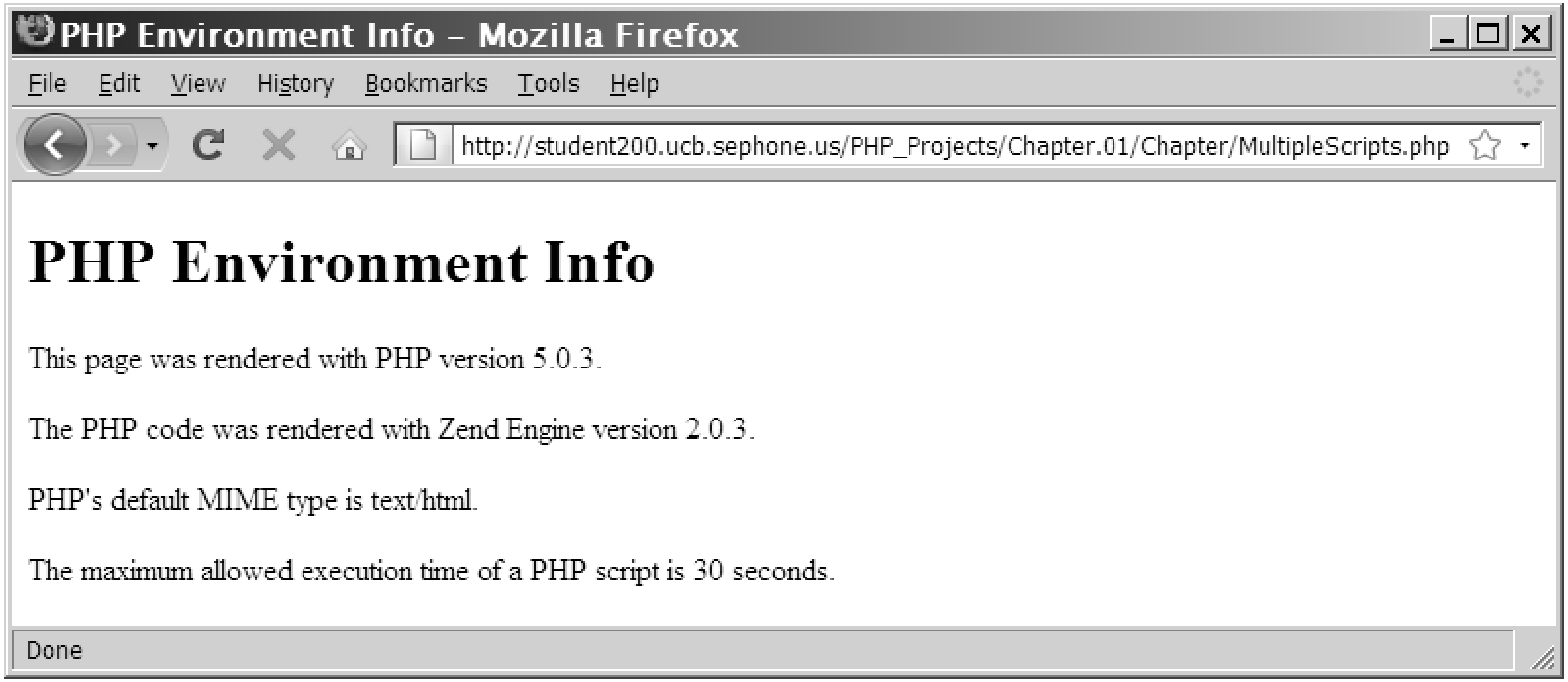
Creating Multiple Code Declaration Blocks (cont.)


Multiple Script Sections – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephone.us
Done

```
Multiple Script Sections

First Script Section

Output from the first script section.

Second Script Section

Output from the second script section.
```

Figure 1-9 Output of a document with two PHP script sections

Creating Multiple Code Declaration Blocks (cont.)


PHP Environment Info – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephone.us/PHP_Projects/Chapter.01/Chapter/MultipleScripts.php
Done

```
PHP Environment Info

This page was rendered with PHP version 5.0.3.

The PHP code was rendered with Zend Engine version 2.0.3.

PHP's default MIME type is text/html.

The maximum allowed execution time of a PHP script is 30 seconds.
```

Figure 1-10 PHP Environment Information Web page

Case Sensitivity in PHP

- Programming language constructs in PHP are mostly case **insensitive**

```
<?php
echo "<p>Explore <strong>Africa</strong>, <br />";
Echo "<strong>South America</strong>, <br />";
ECHO " and <strong>Australia</strong>!</p>";
?>
```

Adding Comments to a PHP Script

- **Comments** are nonprinting lines placed in code that do not get executed, but provide helpful information, such as:
  - The name of the script
  - Your name and the date you created the program
  - Notes to yourself
  - Instructions to future programmers who might need to modify your work

Adding Comments to a PHP Script (cont.)

- **Line comments** hide a single line of code
  - Add // or # before the text
- **Block comments** hide multiple lines of code
  - Add /* to the first line of code
  - And */ after the last character in the code

Adding Comments to a PHP Script (cont.)

```
<?php
/*
This line is part of the block comment.
This line is also part of the block comment.
*/
echo "<h1>Comments Example</h1>"; // Line comments can
  follow
code statements
// This line comment takes up an entire line.
# This is another way of creating a line comment.
/* This is another way of creating
a block comment. */
?>
```

Short Quiz, p. 22

1. How many code declaration bocks can be inserted in a PHP document?
2. Why does the PHP Group recommend that you use standard PHP script delimiters to write PHP code declaration blocks?
3. What character(s) are used as delimiters to separate multiple arguments (parameters) in a function declaration or function call?
4. Describe the type of information that the phpinfo() function generates.
5. Identify the two types of comments available in PHP and indicate when each would be used?

Using Variables and Constants

- The values stored in computer memory are called **variables**
- The values, or data, contained in variables are classified into categories known as **data types**
- The name you assign to a variable is called an **identifier**

Naming Variables

- The following rules and conventions must be followed when naming a variable:
  - Identifiers must begin with a dollar sign ($)
  - Identifiers may contain uppercase and lowercase letters, numbers, or underscores (_). The first character after the dollar sign must be a letter.
  - Identifiers cannot contain spaces
  - Identifiers are case sensitive

Declaring and Initializing Variables

- Before you can use a variable in your code, you have to create it.
- The process of specifying and creating a variable is called **declaring** the variable.
- The process of assigning a first value to a variable is called **initializing** the variable.
- PHP requires that you declare and initialize in the same statement.

```
$VotingAge = 18;
```

- The value you assign to a variable can be a literal string (enclose in quotes), a numeric value, a Boolean value or another PHP variable.

Displaying Variables

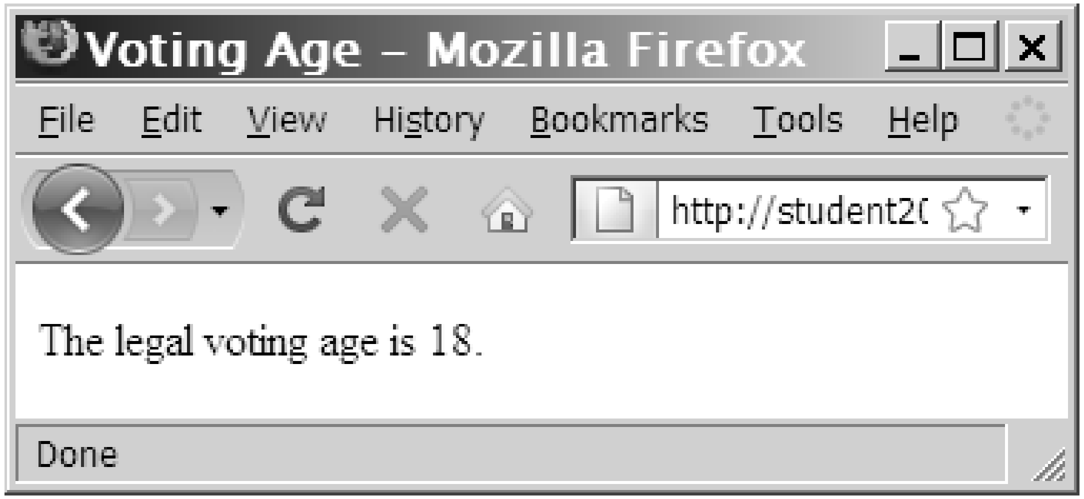
- To display a variable with the `echo` statement, pass the variable name to the `echo` statement without enclosing it in quotation marks:

```
$VotingAge = 18;
echo $VotingAge;
```

- To display both text strings and variables, send them to the `echo` statement as individual arguments, separated by commas:

```
echo "<p>The legal voting age is ",
$VotingAge, ".</p>";
```

Displaying Variables

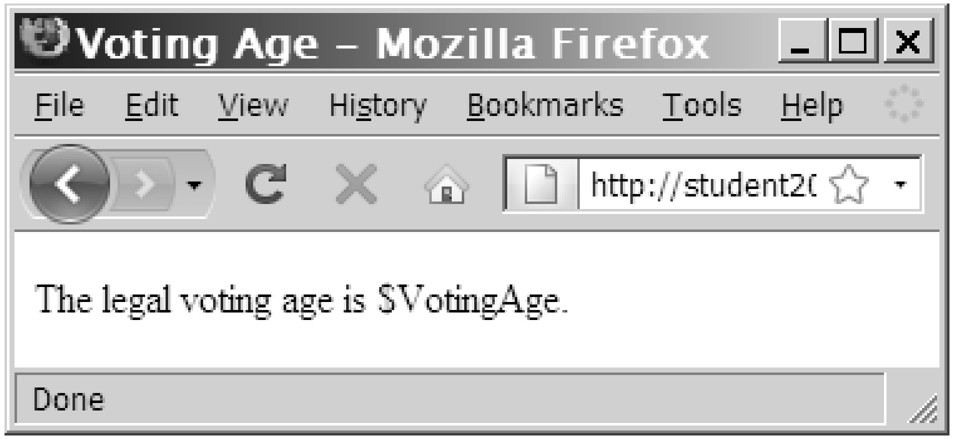

Voting Age – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student2(
Done

```
The legal voting age is 18.
```

Figure 1-11 Output from an echo statement that is passed text and a variable

Displaying Variables (cont.)

- The output of variable names inside a text string depends on whether the string is surrounded by double or single quotation marks


Voting Age – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student2(

```
The legal voting age is $VotingAge.
```

Figure 1-12 Output of an echo statement that includes text and a variable surrounded by single quotation marks

Modifying Variables

- You can modify a variable's value at any point in a script

```
$SalesTotal = 40;
echo "<p>Your sales total is
     $$SalesTotal</p>";
$SalesTotal = 50;
echo "<p>Your new sales total is
$$SalesTotal</p>";
```

Defining Constants

- A **constant** contains information that does not change during the course of program execution
- Constant names do not begin with a dollar sign ($)
- Constant names use all uppercase letters
- Use the **`define()`** function to create a constant

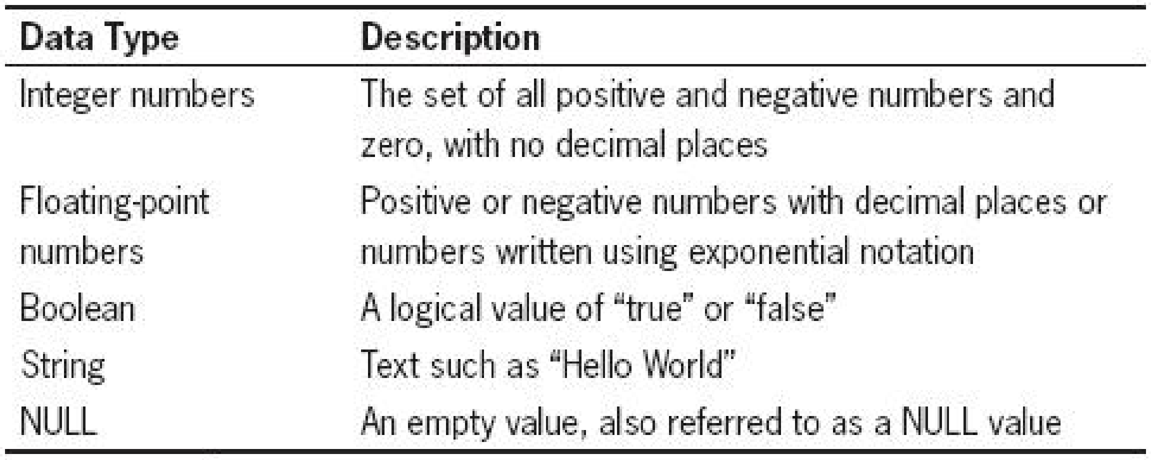
  `define("`*`CONSTANT_NAME`*`", `*`value`*`);`
- The value you pass to the `define()` function can be a text string, number, or Boolean value

Short Quiz, p. 29-30

1. Describe the two-step process of making a variable available for use in the PHP script.
2. Explain the syntax for displaying a variable(s) in the PHP script using the `echo` or `print` statements.
3. How do you make a constant name case insensitive?

Working with Data Types

- A **data type** is the specific category of information that a variable contains
- Data types that can be assigned only a single value are called **primitive types**



Data Type	Description
Integer numbers	The set of all positive and negative numbers and zero, with no decimal places
Floating-point numbers	Positive or negative numbers with decimal places or numbers written using exponential notation
Boolean	A logical value of "true" or "false"
String	Text such as "Hello World"
NULL	An empty value, also referred to as a NULL value

Table 1-1 Primitive PHP data types

Working with Data Types (cont.)

- The PHP language supports:
  - A **resource** data type – a special variable that holds a reference to an external resource such as a database or XML file
  - **Reference** or **composite** data types, which contain multiple values or complex types of information
  - Two reference data types: **arrays** and **objects**

Working with Data Types (cont.)

- **Strongly typed programming languages** require you to declare the data types of variables
- **Static or strong typing** refers to data types that do not change after they have been declared
- **Loosely typed programming languages** do not require you to declare the data types of variables
- **Dynamic or loose typing** refers to data types that can change after they have been declared

Numeric Data Types

- PHP supports two numeric data types:
  - An **integer** is a positive or negative number and 0 with no decimal places (-250, 2, 100, 10,000)
  - A **floating-point number** is a number that contains decimal places or that is written in exponential notation (-6.16, 3.17, 2.7541)
    - **Exponential notation**, or **scientific notation,** is a shortened format for writing very large numbers or numbers with many decimal places (2.0e11)

Boolean Values

- A **Boolean value** is a value of `TRUE` or `FALSE`
- It decides which part of a program should execute and which part should compare data
- In PHP programming, you can only use `TRUE` or `FALSE` Boolean values
- In other programming languages, you can use integers such as 1 = `TRUE`, 0 = `FALSE`

Arrays

- An **array** contains a set of data represented by a single variable name

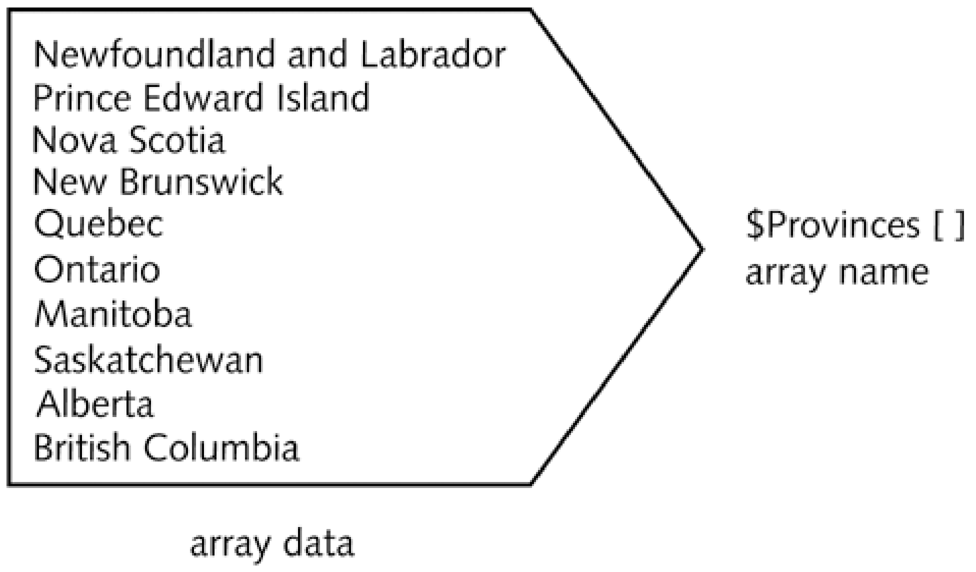


Figure 1-17 Conceptual example of an array

Declaring and Initializing Indexed Arrays

- An **element** refers to each piece of data that is stored within an array
- An **index** is an element's numeric position within the array
  - By default, indexes begin with the number zero (0)
  - An element is referenced by enclosing its index in brackets at the end of the array name:

    ```
    $Provinces[1]
    ```

Declaring and Initializing Indexed Arrays (cont.)

- The `array()` construct syntax is:

  `$array_name = array(values);`

```
$Provinces = array(
     "Newfoundland and Labrador",
     "Prince Edward Island",
     "Nova Scotia",
     "New Brunswick",
     "Quebec",
     "Ontario",
     "Manitoba",
     "Saskatchewan",
     "Alberta",
     "British Columbia"
     );
```

Declaring and Initializing Indexed Arrays (cont.)

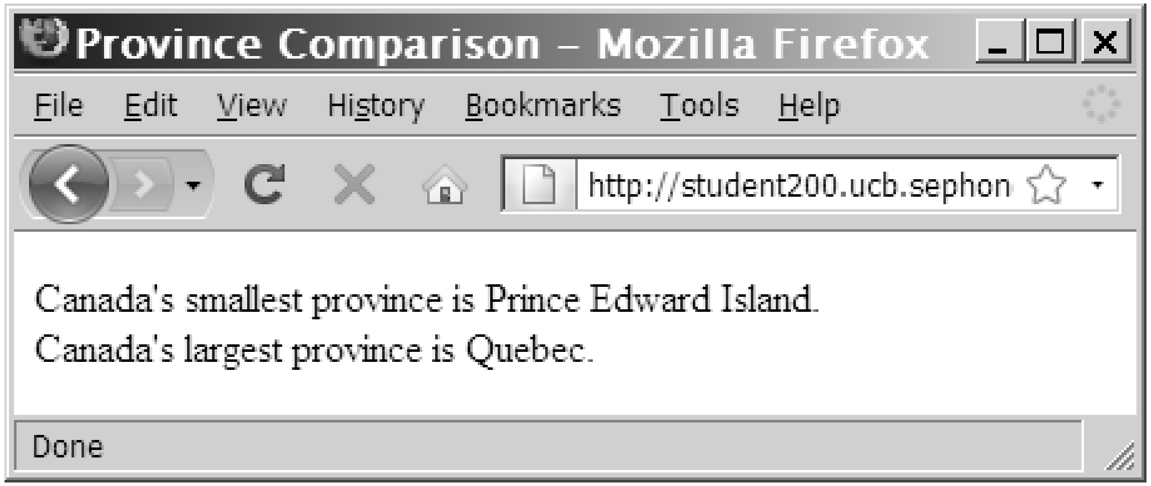
- Array name and brackets syntax is:

  $array_name[ ]

```
$Provinces[] = "Newfoundland and Labrador";
$Provinces[] = "Prince Edward Island";
$Provinces[] = "Nova Scotia";
$Provinces[] = "New Brunswick";
$Provinces[] = "Quebec";
$Provinces[] = "Ontario";
$Provinces[] = "Manitoba";
$Provinces[] = "Saskatchewan";
$Provinces[] = "Alberta";
$Provinces[] = "British Columbia";
```

Accessing Element Information (cont.)

```
echo "<p>Canada's smallest province is
  $Provinces[1].<br />";
echo "Canada's largest province is
  $Provinces[4].</p>";
```


Province Comparison – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephon
Done

```
Canada's smallest province is Prince Edward Island.
Canada's largest province is Quebec.
```

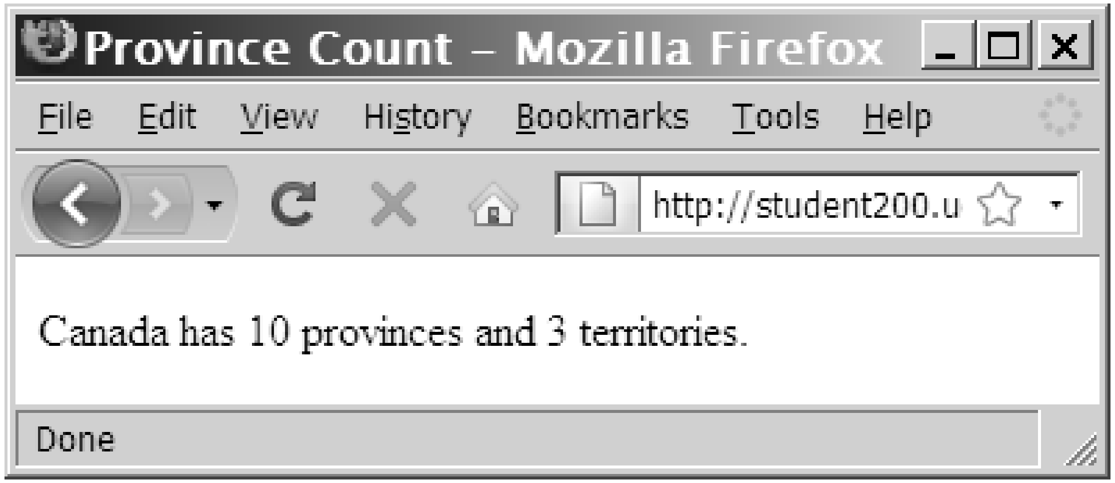
Figure 1-18 Output of elements in the `$Provinces[]` array

Accessing Element Information (cont.)

- Use the **`count()`** function to find the total number of elements in an array

```
$Provinces = array("Newfoundland and Labrador", "Prince
Edward Island", "Nova Scotia", "New Brunswick", "Quebec",
"Ontario", " Manitoba", "Saskatchewan", "Alberta",
"British Columbia");

$Territories = array("Nunavut", "Northwest Territories",
"Yukon Territory");

echo "<p>Canada has ", count($Provinces), " provinces and
", count($Territories), " territories.</p>";
```

Accessing Element Information (cont.)



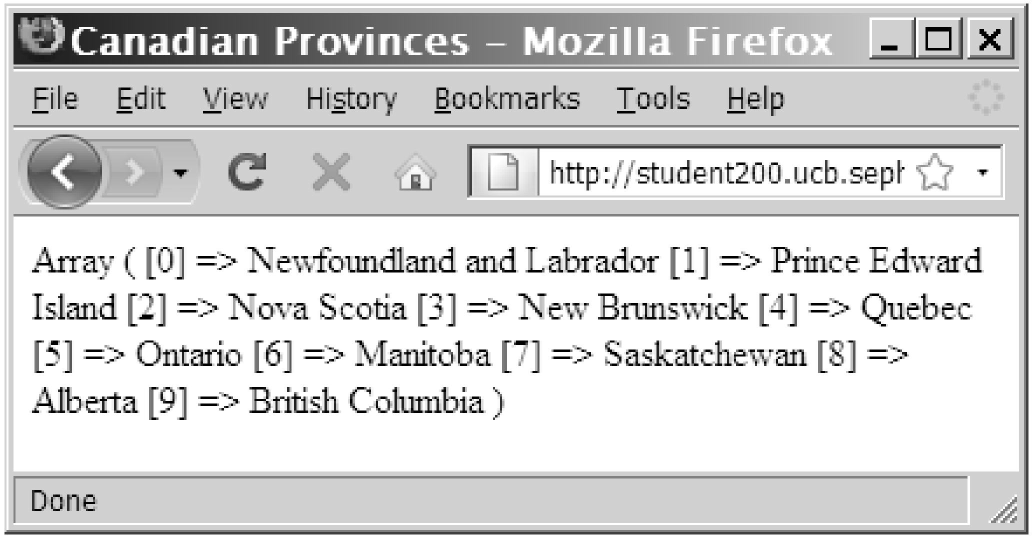
```
Canada has 10 provinces and 3 territories.
```

Figure 1-19 Output of the `count()` function

Accessing Element Information (cont.)

- Use the `print_r()`, `var_dump()` or `var_export()` functions to display or return information about variables
  - The `print_r()` function displays the index and value of each element in an array
  - The `var_dump()` function displays the index, value, data type and number of characters in the value
  - The `var_export()` function is similar to `var_dump()` function except it returns valid PHP code

Accessing Element Information (cont.)


Canadian Provinces – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.seph
Done

```
Array ( [0] => Newfoundland and Labrador [1] => Prince Edward
Island [2] => Nova Scotia [3] => New Brunswick [4] => Quebec
[5] => Ontario [6] => Manitoba [7] => Saskatchewan [8] =>
Alberta [9] => British Columbia )
```

Figure 1-21 Output of the `$Provinces[ ]` array with the `print_r()` function

Modifying Elements

- To modify an array element. include the index for an individual element of the array:

```
$HospitalDepts = array(
    "Anesthesia",              // first element(0)
    "Molecular Biology",       // second element (1)
    "Neurology");              // third element (2)
```

To change the first array element in the `$HospitalDepts[]` array from “Anesthesia” to “Anesthesiology” use:

```
$HospitalDepts[0] = "Anesthesiology";
```

Avoiding Assignment Notation Pitfalls

- Assigns the string "Hello" to a variable named $list

```
$list = "Hello";
```

- Assigns the string "Hello" to a new element appended to the end of the $list array

```
$list[] = "Hello";
```

- Replaces the value stored in the first element (index 0) of the $list array with the string "Hello"

```
$list[0] = "Hello";
```

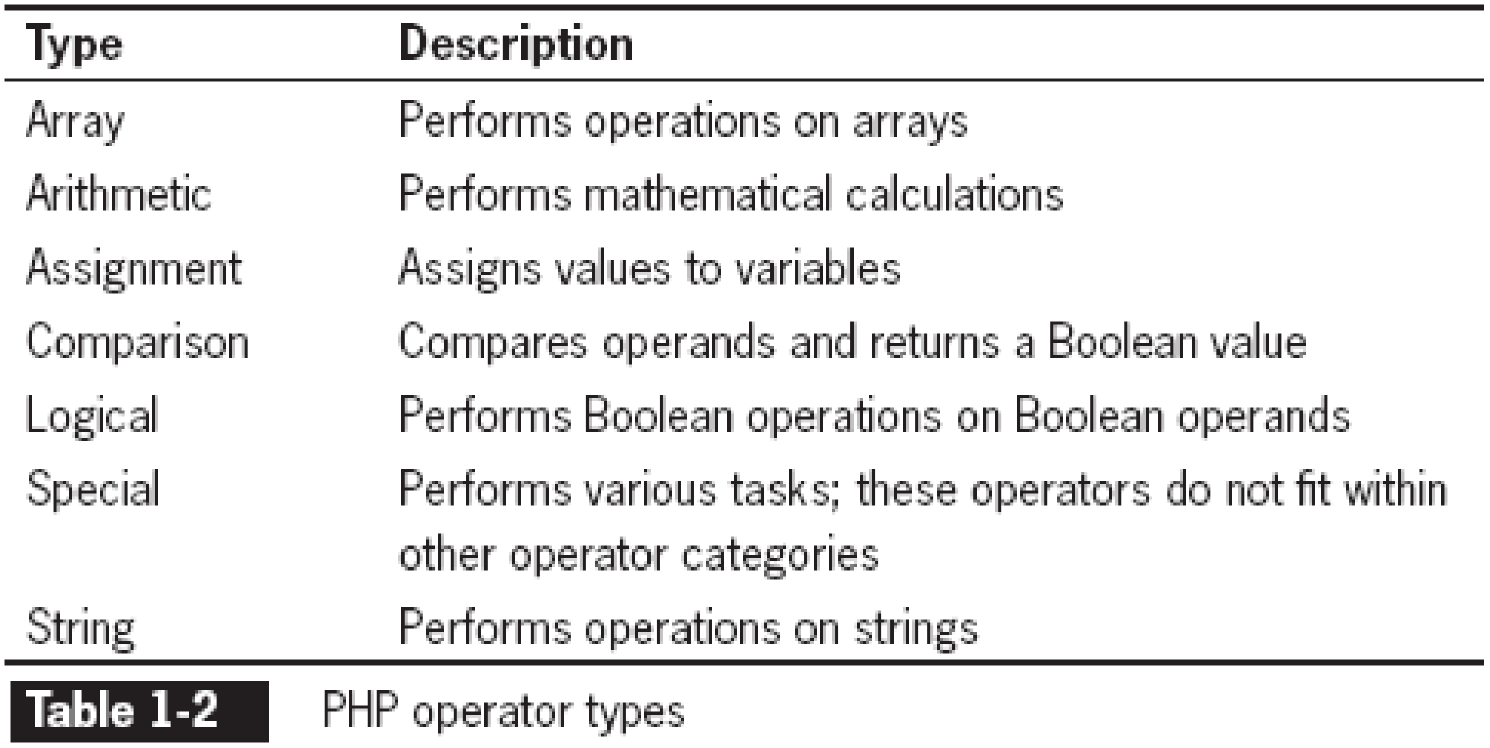
Short Quiz, p. 40

1. Explain why you do not need to assign a specific data type to a variable when it is declared.
2. Positive and negative numbers and 0 with no decimal places belong to which data type?
3. Explain how you access the value of the second element in an array named `$signs`.
4. What function can be used to determine the total number of elements in an array?
5. Illustrate the value of using the `print_r()` function to return information about an array variable.

Building Expressions

- An **expression** is a literal value or variable that can be evaluated by the PHP scripting engine to produce a result
- **Operands** are variables and literals contained in an expression
- A **literal** is a static value such as a literal string or a number
- **Operators** are symbols (+) (*) that are used in expressions to manipulate operands

Building Expressions (cont.)



Type	Description
Array	Performs operations on arrays
Arithmetic	Performs mathematical calculations
Assignment	Assigns values to variables
Comparison	Compares operands and returns a Boolean value
Logical	Performs Boolean operations on Boolean operands
Special	Performs various tasks; these operators do not fit within other operator categories
String	Performs operations on strings

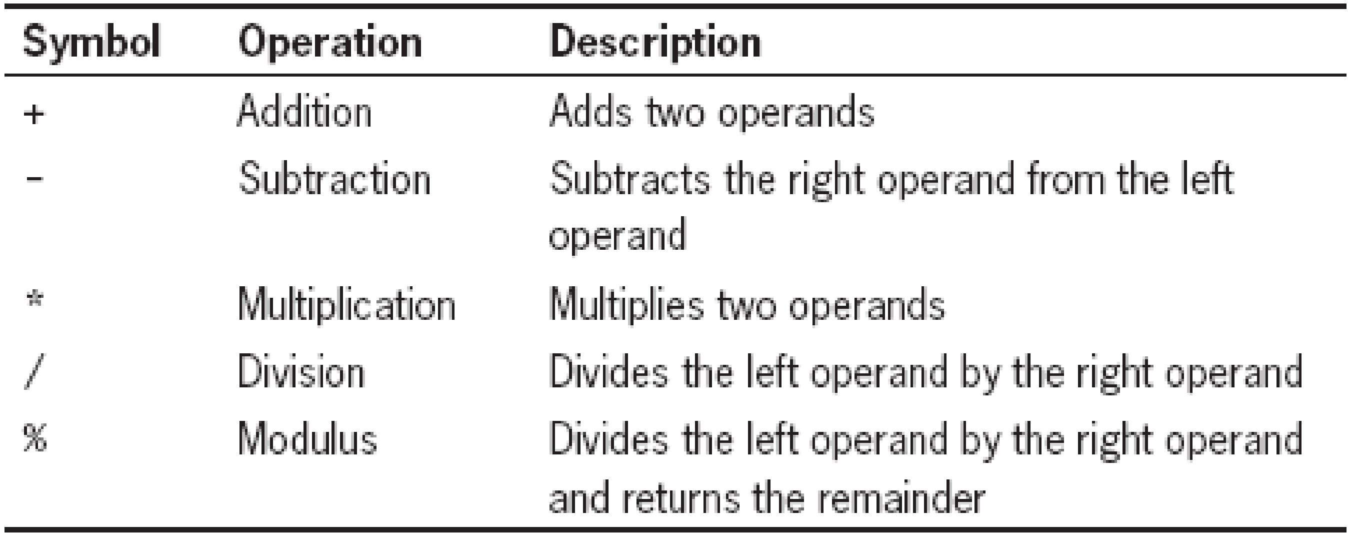
Table 1-2 PHP operator types

Building Expressions (cont.)

- A **binary operator** requires an operand before and after the operator
  - `$MyNumber = 100;`
- A **unary operator** requires a single operand either before or after the operator
  - `$MyNumber++;`

Arithmetic Operators

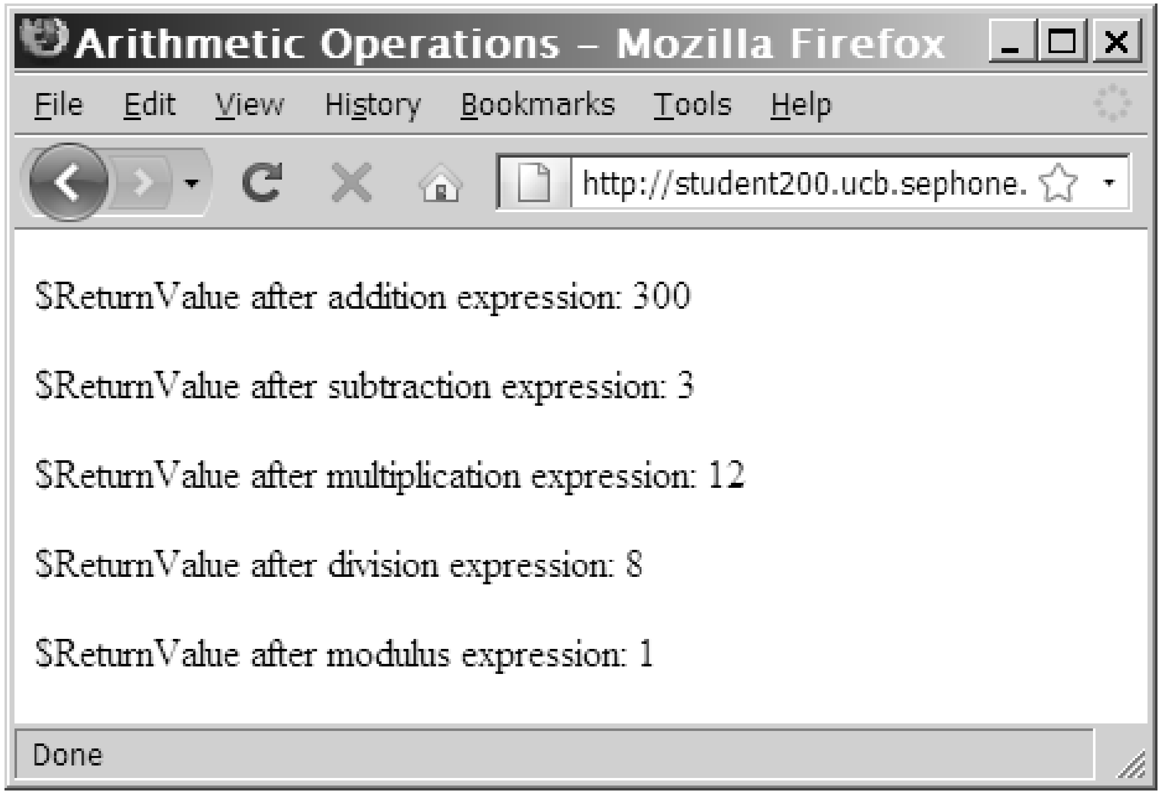
- **Arithmetic operators** are used in PHP to perform mathematical calculations (+ – x ÷)



Symbol	Operation	Description
+	Addition	Adds two operands
-	Subtraction	Subtracts the right operand from the left operand
*	Multiplication	Multiplies two operands
/	Division	Divides the left operand by the right operand
%	Modulus	Divides the left operand by the right operand and returns the remainder

Table 1-3 PHP arithmetic binary operators

Arithmetic Operators (cont.)

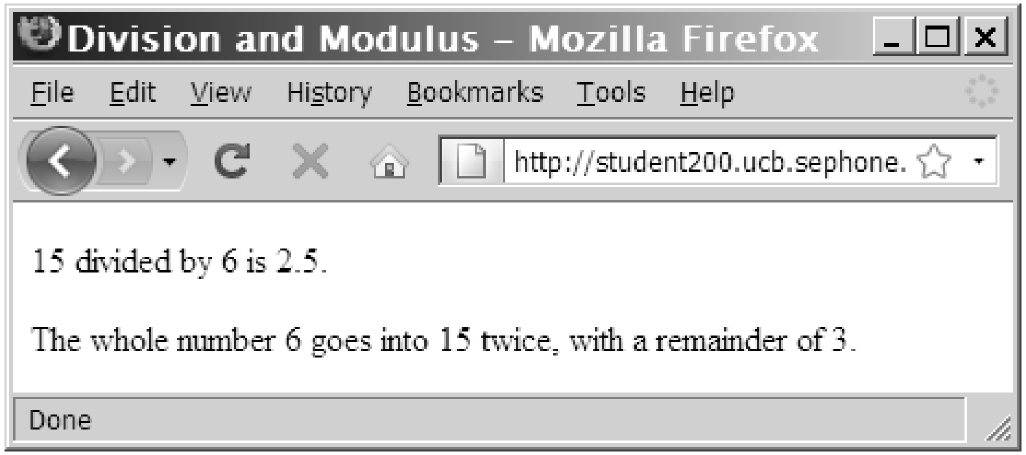

Arithmetic Operations - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephone.
Done

```
$ReturnValue after addition expression: 300

$ReturnValue after subtraction expression: 3

$ReturnValue after multiplication expression: 12

$ReturnValue after division expression: 8

$ReturnValue after modulus expression: 1
```

Figure 1-22 Results of arithmetic expressions

Arithmetic Operators (cont.)

```
$DivisionResult = 15 / 6;
$ModulusResult = 15 % 6;
echo "<p>15 divided by 6 is
       $DivisionResult.</p>"; // prints '2.5'
echo "The whole number 6 goes into 15 twice, with a
       remainder of $ModulusResult.</p>"; // prints '3'
```

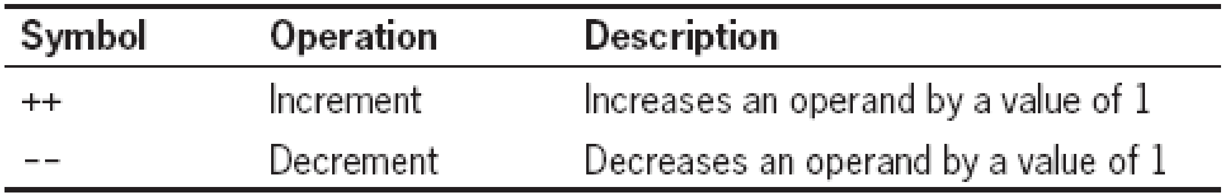

Division and Modulus – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.sephone.
Done

```
15 divided by 6 is 2.5.

The whole number 6 goes into 15 twice, with a remainder of 3.
```

Figure 1-23 Division and modulus expressions

Arithmetic Unary Operators

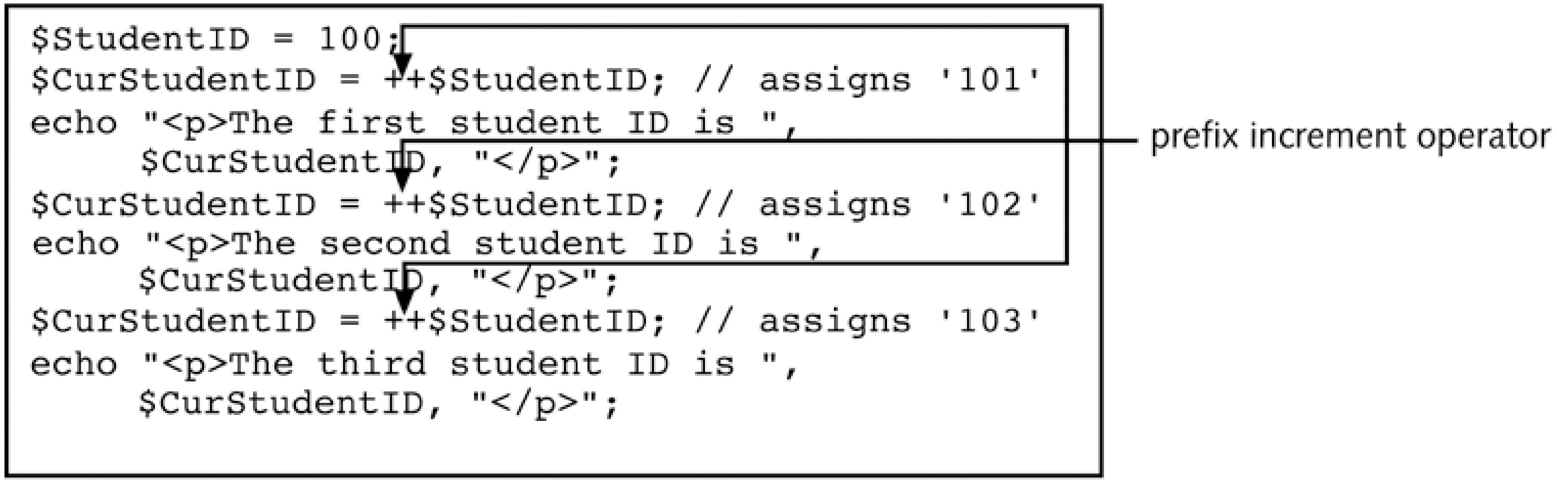
- The increment (++) and decrement (--) unary operators can be used as prefix or postfix operators
- A **prefix operator** is placed before a variable
- A **postfix operator** is placed after a variable



Symbol	Operation	Description
++	Increment	Increases an operand by a value of 1
--	Decrement	Decreases an operand by a value of 1

Table 1-4 PHP arithmetic unary operators

Arithmetic Unary Operators (cont.)

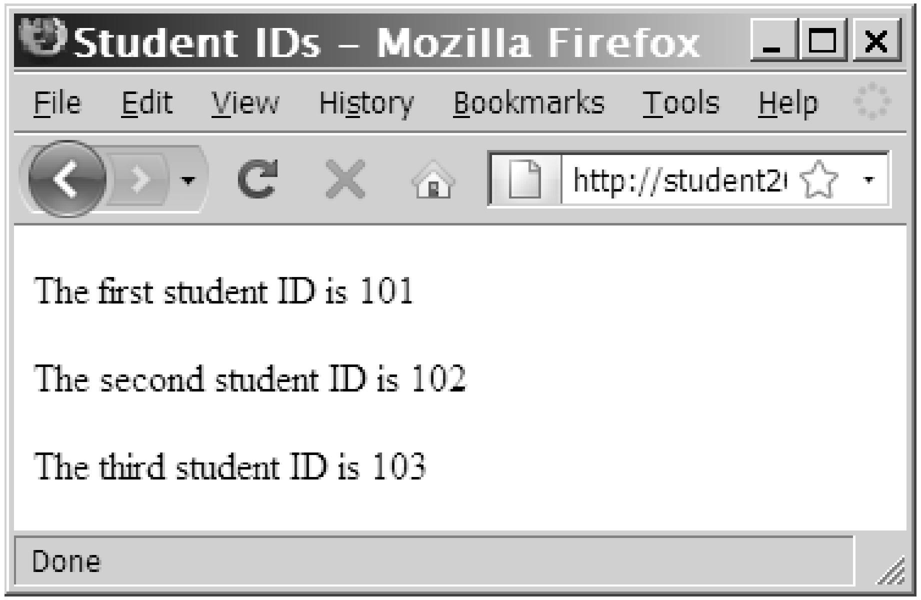


```
$StudentID = 100;
$CurStudentID = ++$StudentID; // assigns '101'
echo "<p>The first student ID is ",
     $CurStudentID, "</p>";
$CurStudentID = ++$StudentID; // assigns '102'
echo "<p>The second student ID is ",
     $CurStudentID, "</p>";
$CurStudentID = ++$StudentID; // assigns '103'
echo "<p>The third student ID is ",
     $CurStudentID, "</p>";
```

Figure 1-24 Script that uses the prefix increment operator

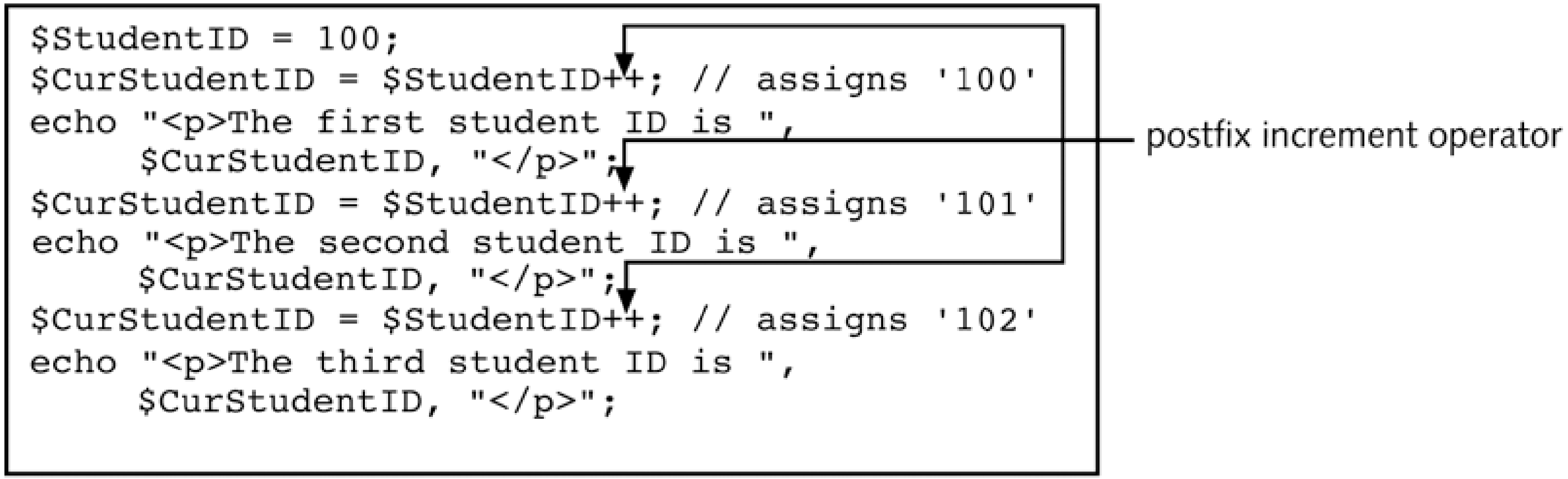
Arithmetic Unary Operators (cont.)

Figure 1-25 Output of the prefix version of the student ID script


Student IDs – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student2
Done

```
The first student ID is 101

The second student ID is 102

The third student ID is 103
```

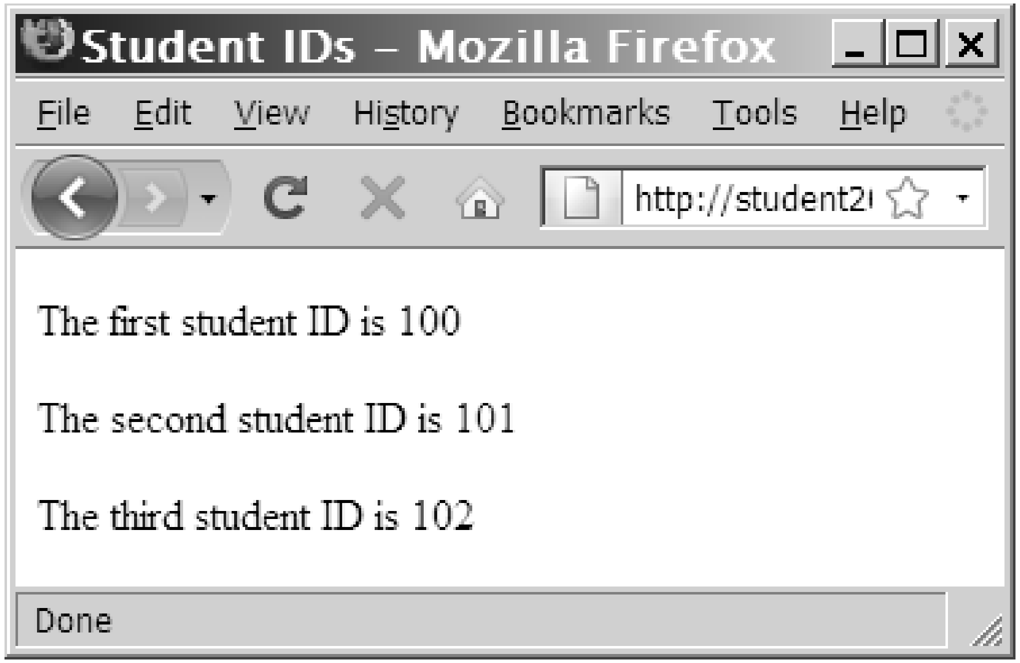
Arithmetic Unary Operators (cont.)



```
$StudentID = 100;
$CurStudentID = $StudentID++; // assigns '100'
echo "<p>The first student ID is ",
     $CurStudentID, "</p>";
$CurStudentID = $StudentID++; // assigns '101'
echo "<p>The second student ID is ",
     $CurStudentID, "</p>";
$CurStudentID = $StudentID++; // assigns '102'
echo "<p>The third student ID is ",
     $CurStudentID, "</p>";
```

Figure 1-26 Script that uses the postfix increment operator

Arithmetic Unary Operators (cont.)


Student IDs – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student2
Done

```
The first student ID is 100

The second student ID is 101

The third student ID is 102
```

Figure 1-27 Output of the postfix version of the student ID script

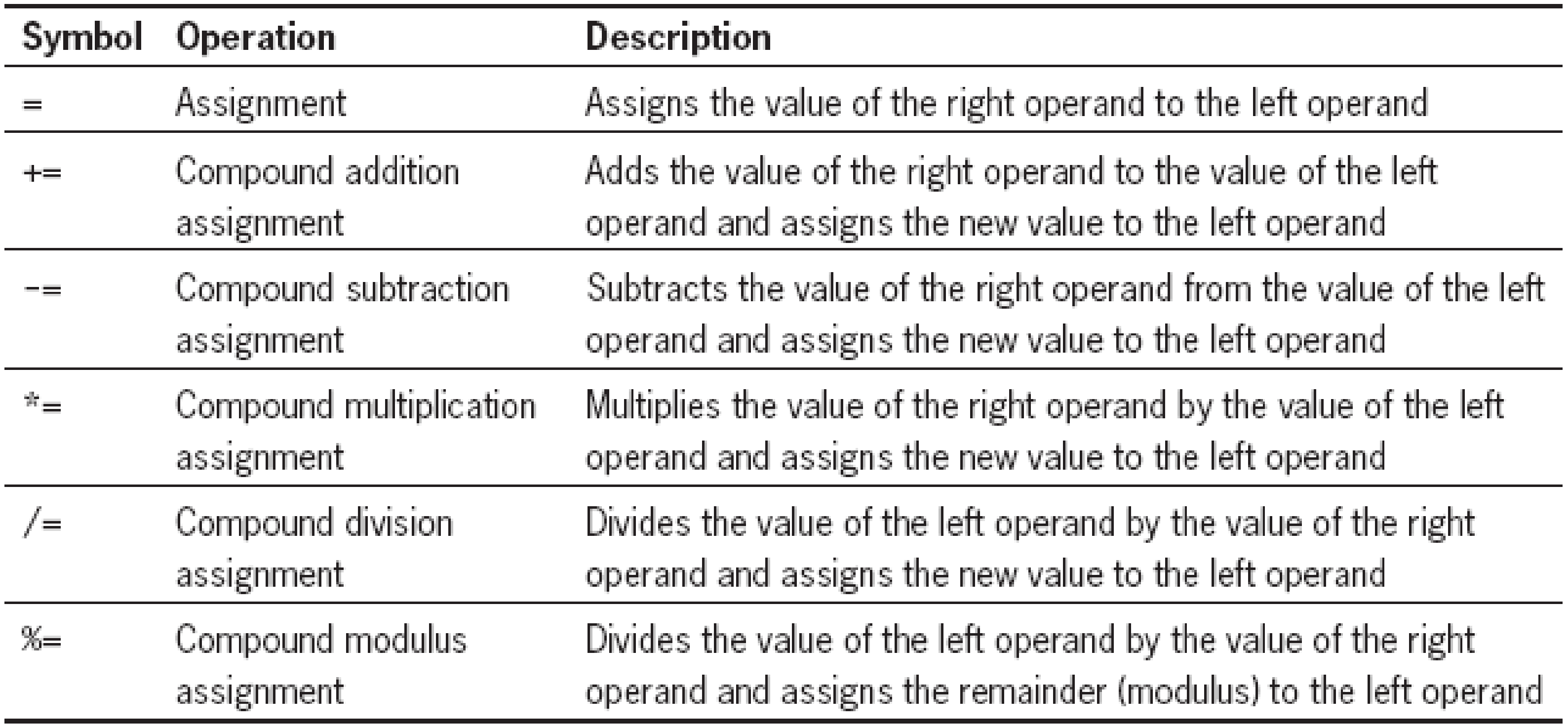
Assignment Operators

- **Assignment operators** are used for assigning a value to a variable:

```
    $MyFavoriteSuperHero = "Superman";
    $MyFavoriteSuperHero = "Batman";
```

- **Compound assignment operators** perform mathematical calculations on variables and literal values in an expression, and then assign a new value to the left operand

Assignment Operators (cont.)



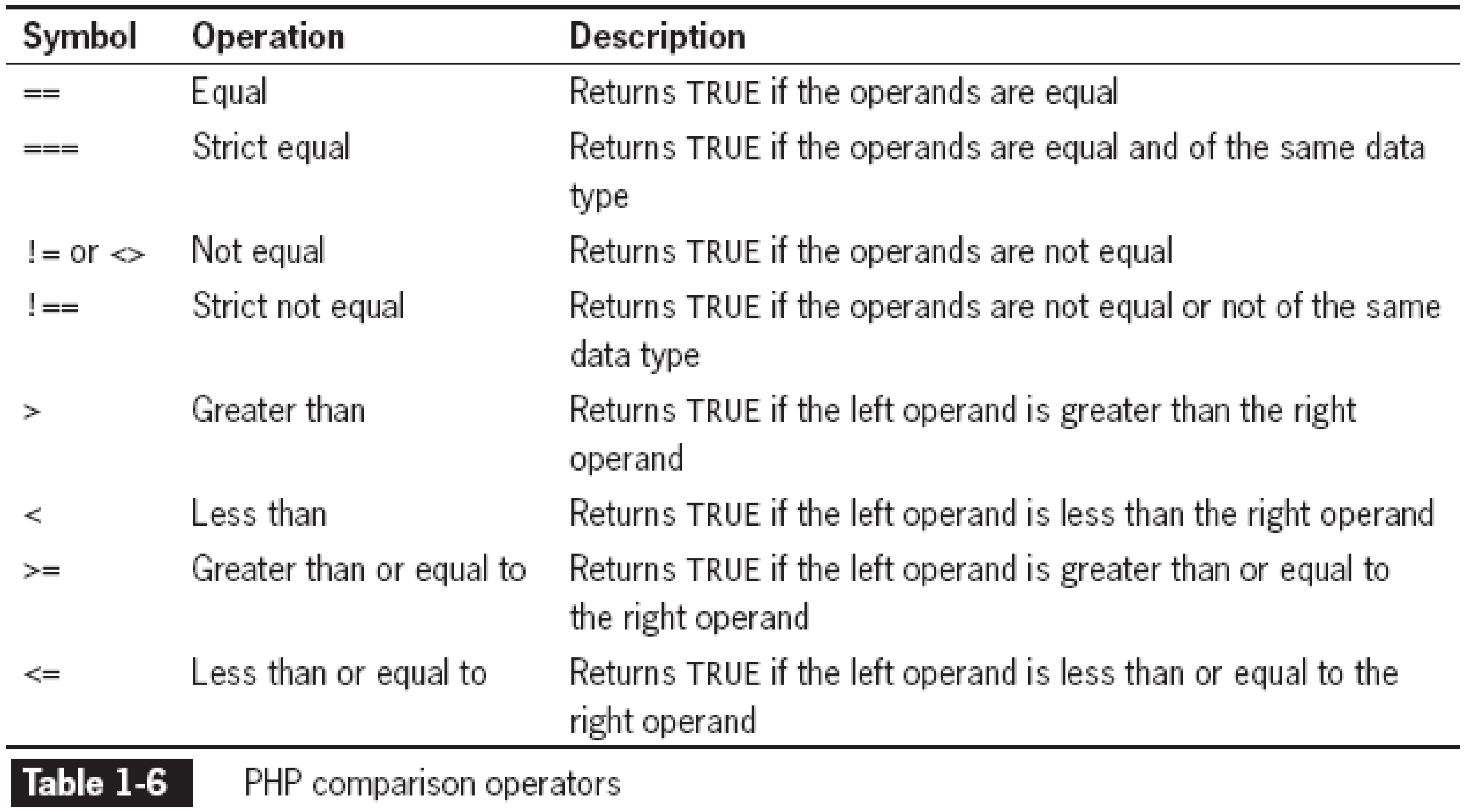
Symbol	Operation	Description
=	Assignment	Assigns the value of the right operand to the left operand
+=	Compound addition assignment	Adds the value of the right operand to the value of the left operand and assigns the new value to the left operand
-=	Compound subtraction assignment	Subtracts the value of the right operand from the value of the left operand and assigns the new value to the left operand
*=	Compound multiplication assignment	Multiplies the value of the right operand by the value of the left operand and assigns the new value to the left operand
/=	Compound division assignment	Divides the value of the left operand by the value of the right operand and assigns the new value to the left operand
%=	Compound modulus assignment	Divides the value of the left operand by the value of the right operand and assigns the remainder (modulus) to the left operand

Table 1-5 Common PHP assignment operators

Comparison and Conditional Operators

- **Comparison operators** are used to compare two operands and determine how one operand compares to another
- A Boolean value of `TRUE` or `FALSE` is returned after two operands are compared
- The comparison operator *compares* values, whereas the assignment operator *assigns* values
- Comparison operators are used with **conditional statements** and **looping statements**

Comparison and Conditional Operators (cont.)



Symbol	Operation	Description
==	Equal	Returns TRUE if the operands are equal
===	Strict equal	Returns TRUE if the operands are equal and of the same data type
!= or <>	Not equal	Returns TRUE if the operands are not equal
!==	Strict not equal	Returns TRUE if the operands are not equal or not of the same data type
>	Greater than	Returns TRUE if the left operand is greater than the right operand
<	Less than	Returns TRUE if the left operand is less than the right operand
>=	Greater than or equal to	Returns TRUE if the left operand is greater than or equal to the right operand
<=	Less than or equal to	Returns TRUE if the left operand is less than or equal to the right operand

Table 1-6 PHP comparison operators

Comparison and Conditional Operators (cont.)

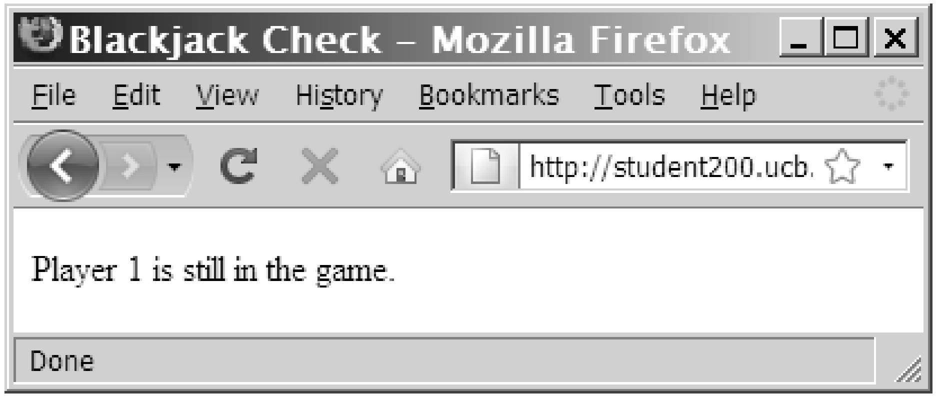
- The **conditional operator** executes one of two expressions, based on the results of a conditional expression
- The syntax for the conditional operator is:

```
conditional expression ? expression1 : expression2;
```

- If the conditional expression evaluates to `TRUE`, *`expression1`* executes
- If the conditional expression evaluates to `FALSE`, *`expression2`* executes

Comparison and Conditional Operators (cont.)

```
$BlackjackPlayer1 = 20;
 ($BlackjackPlayer1 <= 21) ? $Result =
      "Player 1 is still in the game." : $Result =
      "Player 1 is out of the action.";
 echo "<p>", $Result, "</p>";
```

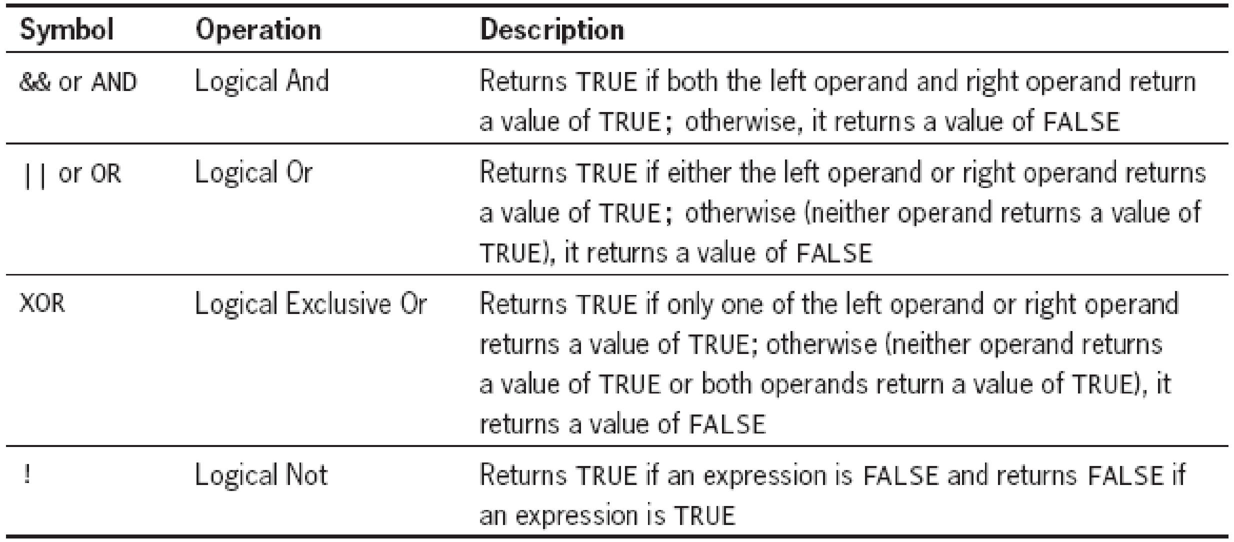

Blackjack Check – Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://student200.ucb.
Done

Player 1 is still in the game.

Figure 1-31 Output of a script with a conditional operator

Logical Operators

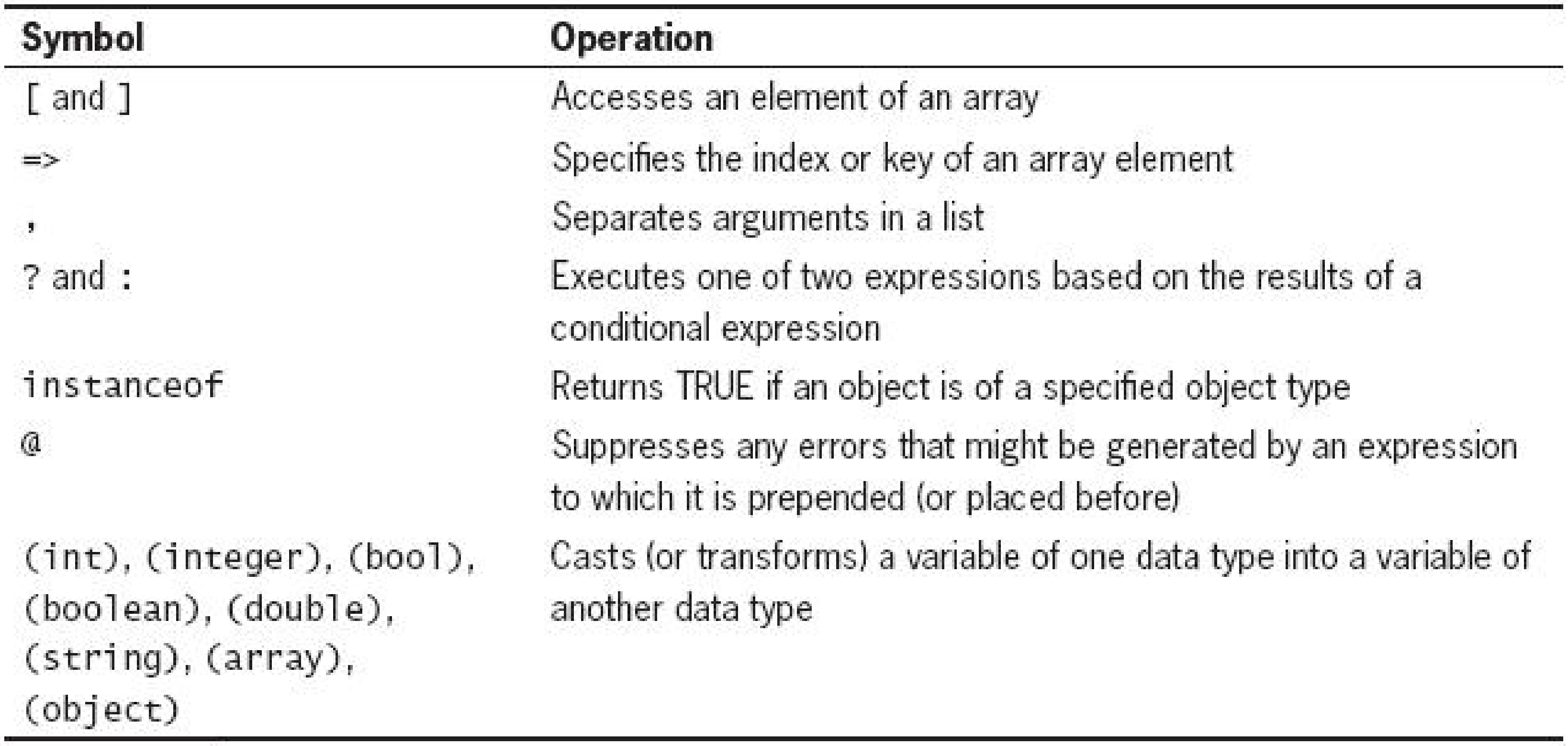
- **Logical operators** are used for comparing two Boolean operands for equality
- A Boolean value of `TRUE` or `FALSE` is returned after two operands are compared



Symbol	Operation	Description
&& or AND	Logical And	Returns TRUE if both the left operand and right operand return a value of TRUE; otherwise, it returns a value of FALSE
|| or OR	Logical Or	Returns TRUE if either the left operand or right operand returns a value of TRUE; otherwise (neither operand returns a value of TRUE), it returns a value of FALSE
XOR	Logical Exclusive Or	Returns TRUE if only one of the left operand or right operand returns a value of TRUE; otherwise (neither operand returns a value of TRUE or both operands return a value of TRUE), it returns a value of FALSE
!	Logical Not	Returns TRUE if an expression is FALSE and returns FALSE if an expression is TRUE

Table 1-7 PHP logical operators

Special Operators



Symbol	Operation
[and]	Accesses an element of an array
=>	Specifies the index or key of an array element
,	Separates arguments in a list
? and :	Executes one of two expressions based on the results of a conditional expression
instanceof	Returns TRUE if an object is of a specified object type
@	Suppresses any errors that might be generated by an expression to which it is prepended (or placed before)
(int), (integer), (bool), (boolean), (double), (string), (array), (object)	Casts (or transforms) a variable of one data type into a variable of another data type

Table 1-8 PHP special operators

Type Casting

- **Casting** or **type casting** copies the value contained in a variable of one data type into a variable of another data type
- The PHP syntax for casting variables is:
  `$NewVariable = (new_type) $OldVariable;`
- `(new_type)` refers to the type-casting operator representing the type to which you want to cast the variable

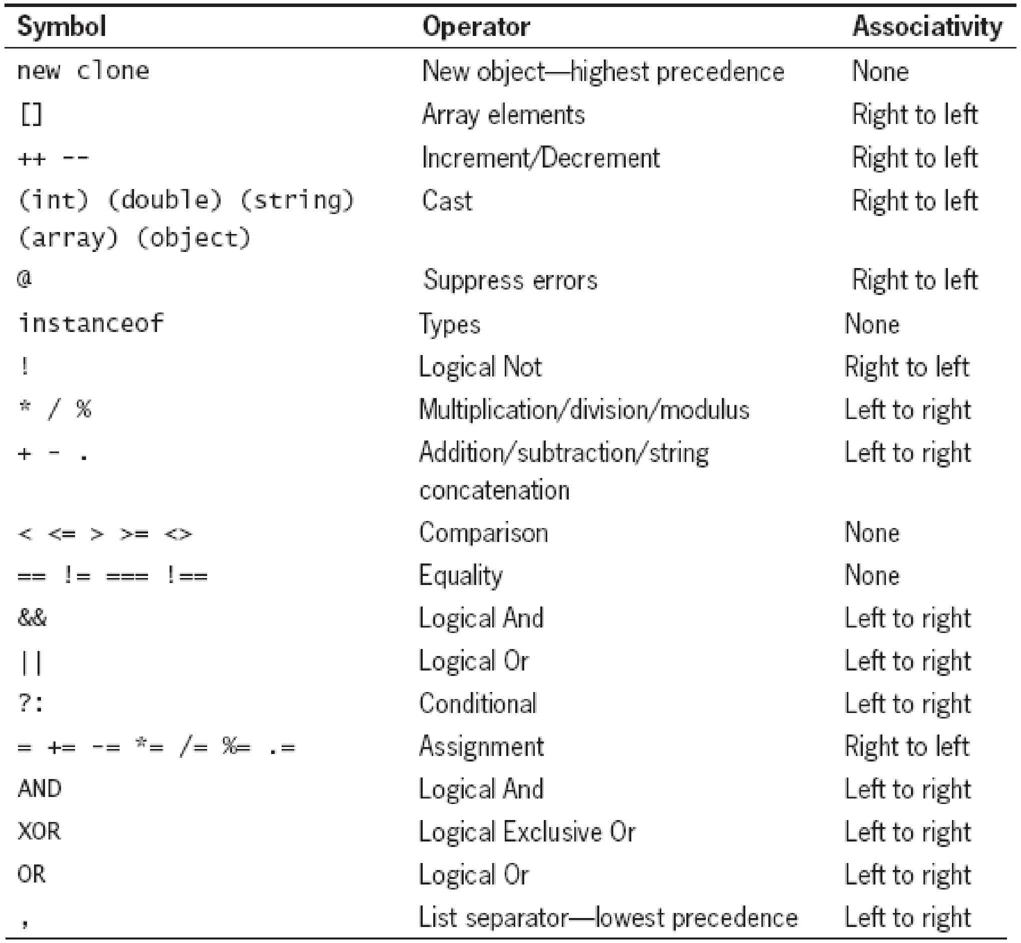
Type Casting (cont.)

- `gettype()` function - returns one of the following strings, depending on the data type:
  - Boolean
  - Integer
  - Double
  - String
  - Array
  - Object
  - Resource
  - NULL
  - Unknown type

Understanding Operator Precedence

- **Operator precedence** refers to the order in which operations in an expression are evaluated
- **Associativity** is the order in which operators of equal precedence execute
- Associativity is evaluated on a left-to-right or a right-to-left basis

Understanding Operator Precedence (cont.)



Symbol	Operator	Associativity
`new clone`	New object—highest precedence	None
`[]`	Array elements	Right to left
`++ --`	Increment/Decrement	Right to left
`(int) (double) (string) (array) (object)`	Cast	Right to left
`@`	Suppress errors	Right to left
`instanceof`	Types	None
`!`	Logical Not	Right to left
`* / %`	Multiplication/division/modulus	Left to right
`+ - .`	Addition/subtraction/string concatenation	Left to right
`< <= > >= <>`	Comparison	None
`== != === !==`	Equality	None
`&&`	Logical And	Left to right
`||`	Logical Or	Left to right
`?:`	Conditional	Left to right
`= += -= *= /= %= .=`	Assignment	Right to left
AND	Logical And	Left to right
XOR	Logical Exclusive Or	Left to right
OR	Logical Or	Left to right
`,`	List separator—lowest precedence	Left to right

Table 1-9 Operator precedence in PHP

Short Quiz, p. 63

1. What symbol is used to divide the left operand by the right operand and return the remainder?
2. Explain the difference between an assignment operator and a compound assignment operator.
3. Explain the difference between a prefix and a postfix operator.
4. Define the term "associativity" as it applies to the order of precedence.

Summary

- JavaScript and PHP are both referred to as embedded languages because code for both languages is embedded within a Web page (either an HTML or XHTML document)
- You write PHP scripts within code declaration blocks, which are separate sections within a Web page that are interpreted by the scripting engine
- The individual lines of code that make up a PHP script are called statements

Summary (cont.)

- The term, **function**, refers to a procedure (or individual statements grouped into a logical unit) that performs a specific task
- Comments are lines that you place in code to contain various types of remarks, including the name of the script, your name and the date you created the program, notes to yourself, or instructions to future programmers who might need to modify your work
  - Comments do not display in the browser

Summary (cont.)

- The values a program stores in computer memory are commonly called **variables**
- The name you assign to a variable is called an **identifier**
- A **constant** contains information that cannot change during the course of program execution
- A **data type** is the specific category of information that a variable contains
- PHP is a *loosely-typed* programming language

Summary (cont.)

- An **integer** is a positive or negative number or zero, with no decimal places
- A floating-point number contains decimal places or is written in exponential notation
- A **Boolean** value is a logical value of `TRUE` or `FALSE`
- An **array** contains a set of data represented by a single variable name

Summary (cont.)

- An **expression** is a single literal value or variable or a combination of literal values, variables, operators, and other expressions that can be evaluated by the PHP scripting engine to produce a result
- **Operands** are variables and literals contained in an expression. A literal is a value such as a string or a number.

Summary (cont.)

- **Operators** are symbols used in expressions to manipulate operands, such as the addition operator (+) and multiplication operator (*)
- A **binary operator** requires an operand before and after the operator
- A **unary operator** requires a single operand either before or after the operator

Summary (cont.)

- **Arithmetic operators** are used in the PHP scripting engine to perform mathematical calculations, such as addition, subtraction, multiplication, and division
- **Assignment operators** are used for assigning a value to a variable
- **Comparison operators** are used to determine how one operand compares with another

Summary (cont.)

- The **conditional operator** executes one of two expressions, based on the results of a conditional expression
- **Logical operators** are used to perform operations on Boolean operands
- **Casting** or type casting creates an equivalent value in a specific data type for a given value
- **Operator precedence** is the order in which operations in an expression are evaluated