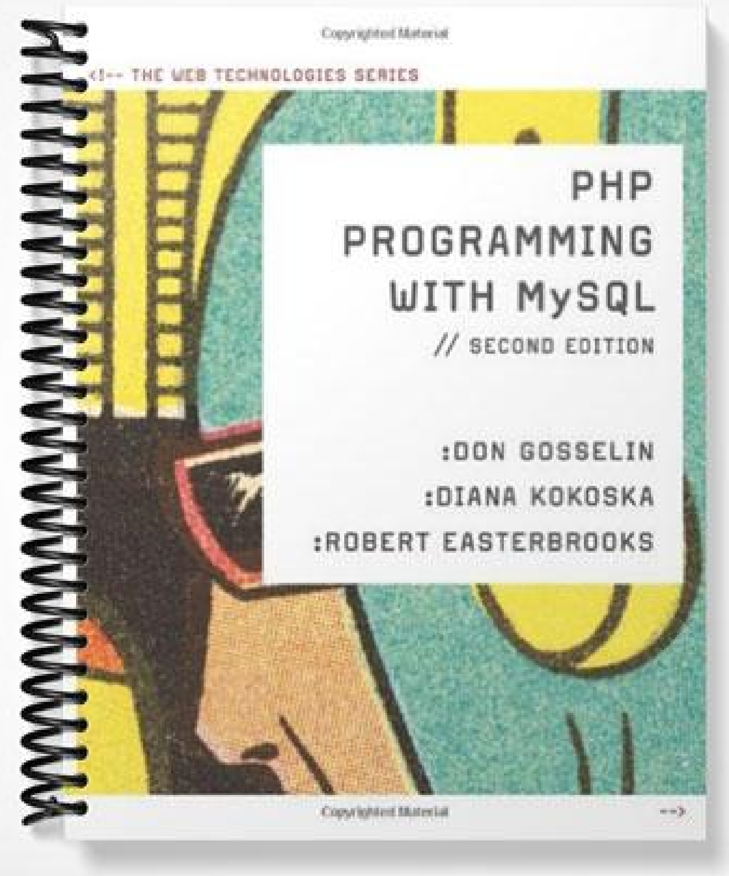
SOLUTIONS MANUAL



Chapter 2

Functions and Control Structures

At a Glance

Table of Contents

Lecture Notes

Chapter Overview

In this chapter, students will study how to use functions to organize their PHP code. They will also learn about variable scope, nested control structures, and conditional coding: `if`, `if`…`else`, and `switch` statements and `while`, `do`…`while`, `for`, and `foreach` looping statements. Students will also be introduced to the `include` and `require` statements.

Chapter Objectives

In this chapter, students will:

- Study how to use functions to organize your PHP code
- Learn about variable scope
- Make decisions using `if`, `if`…`else`, and `switch` statements
- Repeatedly execute code using `while`, `do`…`while`, `for`, and `foreach` statements
- Learn about `include` and `require` statements

Instructor Notes

In PHP, groups of statements that you can execute as a single unit are called *functions*. Functions are often made up of decision-making and looping statements. These are two of the most fundamental statements that execute in a program.

Teaching Tip	Mention to your students that functions are like paragraphs in writing language. Remember a paragraph is a group of related sentences that make up one idea. A function is a group of related statements that make up a single task.

Working with Functions

PHP has two types of functions: *built–in* (internal) and *user-defined* (custom) functions. PHP has over 1000 built-in library functions that can be used without declaring them. You can also your own user-defined functions to perform specific tasks.

Defining Functions

To begin writing a function, you first need to define it. The syntax for the function definition is:

```
function name of _function(parameters) {
statements;
}
```

Parameters are placed within the parentheses that follow the function name. The parameter receives its value when the function is called. When you name a variable within parentheses, you do not need to explicitly declare and initialize the parameter as you do with a regular variable.

Following the parentheses that contain the function parameters is a set of curly braces, called *function braces*, which contain the function statements. *Function statements* are the statements that do the actual work of the function and must be contained within the function braces. For example:

```
function displayCompanyName($Company1) {

 echo "<p>$Company1</p>";
 }
```

Teaching Tip	Remind your students that functions, like all PHP code, must be contained with `<?php...?>` tags. Stress that the function name should be descriptive of the task that the function will perform. Ask students to add this to their list of PHP “Best Coding Practices.” Illustrate to students that in a function: • There is no space between the function name and the opening parentheses • The opening curly brace is on the same line as the function name • The function statements are indented five spaces • The closing curly brace is on a separate line Ask students to add this to their list of PHP “Best Coding Practices.”

Calling Functions

A function definition does not execute automatically. Creating a function definition only names the function, specifies its parameters (if any), and organizes the statements it will execute.

Teaching Tip	Refer to the textbook example on page 77, which defines a function and calls it passing a literal value to the function argument.

Returning Values

Some functions return a value, which may be displayed or passed as an argument to another function. A calculator function is a good example of a return value function. A *return statement* is a statement that returns a value to the statement that called the function.

Teaching Tip	Use the textbook example on page 78 to illustrate how a calling statement calls a function and sends the multiple values as arguments of the function. The function then performs a calculation and returns the value to the calling statement.

Passing Parameters by Reference

Usually, the *value* of a variable is passed as the parameter of a function, which means that a local copy of the variable is created to be used by the function. When the value is returned to the calling statement, any changes are lost. If you want the function to change the value of the parameter, you must pass the value by *reference*, so the function works on the actual value instead of a copy. Any changes made by the function statements remain after the function ends.

Teaching Tip	Use the textbook example on pages 80 and 81 to illustrate the difference of passing a parameter by value or by reference. Explain to students that a function definition should be placed above any calling statements. As of PHP4, this is not required, but is considered a good programming practice. Ask students to add this to their list of PHP “Best Coding Practices.”

Understanding Variable Scope

When you use a variable in a PHP program, you need to be aware of the variable's *scope*—that is, you need to think about where in your program a declared variable can be used. A variable's scope can be either global or local. *Global variables* are declared outside a function and are available to all parts of the programs. *Local variables* are declared inside a function and are only available within the function in which they are declared.

The `global` Keyword

Unlike many programming languages that make global variables automatically available to all parts of your program, in PHP, you must declare a global variable with the `global` keyword inside a function for the variable to be available within the scope of that function. When you declare a global variable with the `global` keyword, you do not need to assign a value, as you do when you declare a standard variable. Instead, within the declaration statement you only need to include the *`global`* keyword along with the name of the variable, as in:

```
global $variable_name;
```

Teaching Tip	Demonstrate the syntax of declaring a global variable within a function using the textbook example on page 83.

Quick Quiz 1

1. In PHP, groups of statements that you can execute as a single unit are called ___________________________.
ANSWER: functions

2. A ___________________________ is a statement that returns a value to the statement that called the function.
ANSWER: return statement

3. ___________________________ are declared inside a function and are only available within the function in which they are declared.
ANSWER: Local variables

4. A ___________________________is a variable that is declared outside a function and is available to all parts of your program.
ANSWER: global variable

Making Decisions

In any programming language, the process of determining the order in which statements execute in the program is called *decision making* or *flow control*. The special types of PHP statements used for making decision are called decision-making statements or control structures.

`if` Statements

The `if` statement is used to execute specific programming code if the evaluation of a conditional expression returns a value of `TRUE`. The syntax for a simple `if` statement is as follows:

```
     if (conditional expression)// condition evaluates to 'TRUE'
          statement;
```

Teaching Tip	You should insert a space after the conditional keyword `if` before the opening parenthesis of the conditional expression. This will help you see a visual difference between a structure and a function. Using a line break and indentation to enter the statements to execute makes it easier for the programmer to follow the flow of the code. Ask students to add the space and indentation to their list of PHP "Best Coding Practices." Explain to students that if there is only one statement, they do not need to use curly braces. If there are multiple statements, the statements should be enclosed within beginning and ending curly braces. `({...})`. Ask students to add the use of curly braces to their list of PHP "Best Coding Practices."

You can use a command block to construct a decision-making structure using multiple `if` statements. A *command block* is a group of statements contained within a set of braces, similar to the way function statements are contained within a set of braces. When an `if` statement evaluates to `TRUE`, the statements in the command block execute.

`if...else` Statements

Should you want to execute one set of statements when the condition evaluates to `FALSE`, and another set of statements when the condition evaluates to `TRUE`, you need to add an `else` clause to the `if` statement.

```
if (conditional expression) {
 statements; //condition evaluates to 'TRUE'
}
else {
 statements; //condition evaluates to 'FALSE'
}
```

Nested `if` and `if...else` Statements

When one decision-making statement is contained within another decision-making statement, they are referred to as nested decision-making structures. An `if` statement contained within an `if` statement or within an `if...else` statement is called a nested `if` statement. Similarly, an `if...else` statement contained within an `if` or `if…else` statement is called a nested `if...else` statement. You use nested `if` and `if...else` statements to perform conditional evaluation that must be executed after the original conditional evaluation.

`switch` Statements

The `switch` statement controls program flow by executing a specific set of statements, depending on the value of the expression. The `switch` statement compares the value of an expression to a value contained within a special statement called a `case` label. A `case` label represents a specific value and contains one or more statements that execute if the value of the case label matches the value of the switch statement's expression. The syntax for the `switch` statement is a follows:

```
switch (expression) {
 case label:
     statement(s);
     break;
 case label:
     statement(s);
     break;
 ...
 default:
     statement(s);
     break;
}
```

Another type of label used within `switch` statements is the `default` label. The `default` label contains statements that execute when the value returned by the `switch` statement does not match a `case` label. A `default` label consists of the keyword `default` followed by a colon. In a switch statement, execution does not automatically end when a matching label is found. A `break` statement is used to exit a control structures before it reaches the closing brace (}). A `break` statement is also used to exit `while`, `do...while`, and `for` looping statements.

Teaching Tip	The `break` statement after the `default` case is optional; however, it is good programming practice to include it. Ask the students to add this to their list of PHP “Best Coding Practices.”

Quick Quiz 2

1. The process of determining the order in which statements execute in a program is called __________________________.
   ANSWER: decision-making or flow control

2. When one decision-making statement is contained within another decision-making statement, they are referred to as __________________________.
   ANSWER: nested decision-making structures

3. The __________________________ statement is used to execute specific programming code if the evaluation of a conditional expression returns a value of `TRUE`.
   ANSWER: `if`

4. The __________________________ statement controls program flow by executing a specific set of statements, depending on the value of the expression.
   ANSWER: `switch`

Repeating Code

Conditional statements allow you select only a single branch of code to execute and then continue to the statement that follows. If you need to perform the same statement more than once, however, you need a use a *loop statement*, a control structure that repeatedly executes a statement or a series of statements while a specific condition is `TRUE` or until a specific condition becomes `TRUE`. In an infinite loop, a loop statement never ends because its conditional expression is never `FALSE`.

`while` Statements

The `while` statement is a simple loop statement that repeats a statement or series of statements as long as a given conditional expression evaluates to `TRUE`. The syntax for the while statement is as follows:

```
while (conditional expression) {
     statement(s);
}
```

Each repetition of a looping statement is called an *iteration*. The loop ends and the next statement, following the `while` statement executes only when the conditional statement evaluates to `FALSE`. You normally track the progress of the `while` statement evaluation, or any other loop, with a counter. A *counter* is a variable that increments or decrements with each iteration of a loop statement.

Teaching Tip	Programmers often name counter variables `$Count`, `$Counter`, or something descriptive of its purpose. The letters i, j, k, l, x, y, z are also commonly used as counter names. Using a variable name such as count or the letter i (for iteration) helps you remember (and informs other programmers) that the variable is being used as a counter.

`do...while` Statements

The `do...while` statement executes a statement or statements once, then repeats the execution as long as a given conditional expression evaluates to `TRUE` The syntax for the `do...while` statement is as follows:

```
do {
     statements(s);
} while (conditional expression);
```

`for` Statements

The `for` statement is used for repeating a statement or series of statements as long as a given conditional expression evaluates to `TRUE`. The syntax of the `for` statement is as follows:

```
for (counter declaration and initialization; condition;
     update statement) {
     statement(s);
 }
```

`foreach` Statements

The `foreach` statement is used to iterate or loop through the elements in an array. With each loop, a `foreach` statement moves to the next element in an array. The basic syntax of the `foreach` statement is as follows:

```
foreach ($array_name as $variable_name) {
      statement(s);
 }
```

Teaching Tip

Most of the conditional statements (such as the `if`, `if...else`, `while`, `do...while`, and `for`) are probably familiar to students, but the `foreach` statement, which iterates through elements of an array may not be as familiar.

Illustrate the syntax of the `foreach` statement using the basic and advanced forms shown on pages 105 – 107 of the textbook.

Quick Quiz 3

1. Each repetition of a looping statement is called a(n) ____________________________.
   ANSWER: iteration

2. A ____________________________ is a variable that increments or decrements with each iteration of a loop statement.
   ANSWER: counter

3. The `foreach` statement is used to iterate or loop through the elements in a(n) ____________________________.
   ANSWER: array

Discussion Questions

- When is it better to use a global variable in programming? When is it better to use a local variable?

- Why are conditional statements important in programming?

- Explain the difference between a `while` statement and a `do`…`while` statement.

Key Terms

- **break statement:** Used to exit control structures.
- **case label:** In a switch statement, represents a specific value and contains one or more statements that execute if the value of the case label matches the value of the switch statement's expression.
- **command block:** A group of statements contained within a set of braces, similar to the way function statements are contained within a set of braces.
- **counter:** A variable that increments or decrements with each iteration of a loop statement.
- **decision making** (or **flow control**)**:** The process of determining the order in which statements execute in a program.
- **default label:** Contains statements that execute when the value returned by the switch statement expression does not match a case label.
- **do…while statement:** Executes a statement or statements once, then repeats the execution as long as a given conditional expression evaluates to TRUE.
- **for statement:** Used for repeating a statement or series of statements as long as a given conditional **expression** evaluates to TRUE.
- **function definition:** Line of code that make up a function.
- **functions:** Groups of statements that execute as a single unit.
- **global variable:** A variable declared outside a function and available to all parts of the program.
- **if statement:** Used to execute specific programming code if the evaluation of a conditional expression returns a value of TRUE.
- **if…else statement:** if statement with an else clause that is implemented when the condition returns a value of FALSE.
- **infinite loop:** A loop statement never ends because its exit condition is never met.
- **iteration:** The repetition of a looping statement.
- **local variable:** A variable declared inside a function and only available with the function in which it is declared.
- **loop statement:** A control structure that repeatedly executes a statement or a series of statements while a specific condition is TRUE or until a specific condition becomes TRUE.
- **nested decision-making structures:** One decision-making statement contained within another decision-making statement.
- **parameter:** A variable that is passed to a function from the calling statement.
- **return statement:** A statement that returns a value to the statement that called the function.
- **switch statement:** Controls program flow by executing a specific set of statements, depending on the value of an expression.
- **variable scope:** The context in which a variable is accessible (such as local or global).
- **while statement:** Repeats a statement or a series of statements as long as a given conditional expression evaluates to TRUE.

Best Coding Practices – Chapter 2

Topic	Best Practice
Formatting code	Format code consistently within a program Indent code five spaces for readability
Writing PHP code blocks	Use the Standard Script Delimiters `<?php … ?>`
Using the echo statement	Use the `echo` construct exclusively Use lowercase characters for the `echo` keyword
Coding the echo statement	Do not use parentheses with the `echo` statement
Adding comments to PHP script	Add both line and block comments to your PHP code
Adding a line comment	Use the two forward slashes `(//)` to begin a line comment
Displaying array elements with built-in functions	Enclose the `print_r()`, `var_export()`, and `var_dump()` functions in beginning and ending `<pre>` tags
Writing a user-defined function	The name of a user-defined function should be descriptive of the task it will perform
Writing a function definition	Do NOT space between the function name and the opening parenthesis Key the opening curly brace on the same line as the function name Indent the function statements five spaces Key the closing curly brace on a separate line
Calling a function	Place the function definition above the calling statement
Writing a control structure	To differentiate a control structure from a function, space once after the conditional keyword before the parenthesis i.e. `if (...)` or `else (...)` Indent the statements to make them easier for the programmer to follow the flow Use curly braces around the statements to execute in a control structure if there is more than one statement.
Coding a switch statement	Include the optional `break` statement after the default `case` label in a `switch` statement
Naming a counter variable	If appropriate, name your counter variable `$Count` or `$Counter` or use the letters (i, j, k, l, x, y , z)