Chapter 3

Fundamentals of PHP

Learning Objectives

After completing this chapter, you will be able to:

- *Understand the basic syntax of PHP*
- *Understand about comments*
- *Understand case sensitivity*
- *Understand echo and print statements*
- *Understand data types*
- *Understand type declaration*

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

INTRODUCTION

PHP stands for "Hypertext Preprocessor". It is a server-side scripting language used for developing dynamic websites. PHP and JavaScript are the languages embedded with HTML and CSS are used to develop dynamic websites. MySQL is a relational database management system and is used for data storage purposes. In this chapter, you will learn basic syntax and functionality of PHP such as comments, case sensitivity, echo and print statements, data types, and type declaration.

PHP FILE

PHP is used to develop dynamic websites. PHP code can be written in any text editor or IDE as discussed in previous chapter. PHP files can contain various other languages including PHP such as HTML, CSS, JavaScript, MySQL, and also plain text. The PHP files are saved with the extension *.php*. If you are working with the XMAPP server, you need to save these files in the *C:\xampp\htdocs* directory.

PHP is embedded with HTML code. Therefore, when a PHP file is executed on the server, it returns the plain HTML result to the browser.

Note

You can save a file having HTML code with .html, .htm, or .php extension. But, to save a file having both HTML and PHP codes, you need to use .php extension only.

SYNTAX OF PHP

PHP has a syntax that governs how a PHP code should be written. PHP script or code can be placed anywhere in the program according to the requirement of website.

The syntax to write a PHP script or code is as follow:

```
<?php

    //PHP Code

?>
```

In the given syntax, **<?php** is the start tag, **//PHP Code** is the one line comment which is not executed as a part of the program, and **?>** is the end/close tag. The code inside these tags will be executed.

PHP code can also start with **<?** tag instead of **<?php** tag; for example, you can write the PHP code inbetween **<?** and **?>** tags where **<?** is the start tag and **?>** is the end or close tag. But, it will not work with XML. Therefore, for the compatibility of PHP with all the scripting languages, you must write the PHP code in between **<?php** and **?>**.

COMMENTS IN PHP

Comments are the statements or lines written in between some special characters or begin with special characters. Comments are used for the reading purpose only. These lines are not

considered as part of the program and ignored at the time of execution. Comments make the program more understandable and are used for the convenience of the programmer. For example, you were working on a PHP project two years ago and now you want to work on that project again, it will become difficult for you to understand the code. But if you have comments in your program, you can easily inspect the functionality of the existing code.

PHP supports C, C++, and Unix shell style comments. There are two types of comments in PHP:

1. One-line comment
2. Multiple line comment

One-Line Comment

One-line comment is a single line comment. The browser ignores the line which is commented. The basic purpose of the comment is to give a name to any part of a code, small description about code, to provide short remarks during programming, or to temporarily remove a line from a program written in PHP wherever needed. There are two ways in which you can write one-line comment in the PHP program:

1. By using pair of forward slashes(//), preceding the line.
2. By using hash(#), preceding the line.

You can use any of them to write a one-line comment.

For example:

```
<?php
   //First one-line comment in PHP.
   #Second one-line comment in PHP.
?>
```

In this example **<?php** is the start tag of PHP, **//First one-line comment in PHP** is a one-line comment, preceding with pair of forward slashes (//). Next, **#Second one-line comment in PHP** is also a one-line comment, preceding with hash(#), and **?>** is the end tag of PHP.

Multiple Line Comment

The Multiple line comment is used to comment more than one line or multiple lines. This comment is used to write a long description, a small documentation about the program, or to temporarily disable a large section of code which you do not want in your program for testing purpose or for some other reasons. Anything written in between /* and */ pair of characters is not executed as a part of the program.

For example:

```
<?php
   /* This is a multiple line comment which is
      used to comment one or more lines in the PHP code. */
?>
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

In the given example, **<?php** is the start tag of PHP and **?>** is the end tag of PHP. In the next line, /* is used to start a multiple line comment, */ is used to end the multiple line comment.

Nested multiple line comments are not possible in PHP. Nested multiple line comments mean one comment inside another comment. You cannot place opening /* and closing */ characters inside another opening /* and closing */ characters. On doing so, the program will generate an error because comment will end automatically at the first occurrence of closing */ character.

For example:

```
<?php
      /* This is a multiple line comment,
      /* multiple line comment ends here. */
      It will generate an error because it will not
      be able to detect the last closing characters of comment*/
?>
```

In the given example, multiple line comment is used inside another multiple line comment which is not possible in PHP. It will generate parsing error when the code is executed.

ECHO AND PRINT STATEMENTS

In PHP, you can display the output data in the browser by writing any of the two statements in your program:

1. **Echo** statement
2. **Print** statement

The **Echo** and **print** statements are almost same, but there are few differences between them which are discussed next.

Echo Statement

In PHP, the **echo** statement is used to display output data in the browser. This statement can be used with parentheses or without parentheses such as **echo()** or **echo**. This statement does not return any value which makes it faster as compared to the **print** statement. Multiple parameters can be passed to the **echo** statement, although usage of multiple parameters or arguments is rare. The echo statement starts with the keyword **echo**.

For example:

```
<?php
   echo "Welcome to CADCIM Technologies";
?>
```

In the given example, **<?php** is the start tag of PHP, **echo "Welcome to CADCIM Technologies";** is the **echo** statement which will be displayed as output in the browser, and **?>** is the closing tag of PHP.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Note

To work with XAMPP and to run examples given in this book, you need to write programs in any text editor or IDE of your choice. Next, you need to save the program at the location C:\xampp\htdocs with .php extension, such as example.php.

Example 1

The following program will display the output text in the browser by using the **echo** statement. The line numbers on the right are not part of this program and are mentioned for reference only.

```
<!Doctype html>                                                    1
<html>                                                             2
<head>                                                             3
<title>echo</title>                                                4
</head>                                                            5
<body>                                                             6
   <h1>First PHP program</h1>                                      7
   <?php                                                           8
         echo "Cadcim Technologies";                              9
         echo "<h2>PHP echo statement</h2>";                      10
         echo "Echo ", "with ", "multiple parameters.";           11
   ?>                                                              12
</body>                                                            13
</html>                                                            14
```

Explanation

Line 1

<!Doctype html>

In this line, **<!Doctype html>** is the HTML tag used to define the version of HTML, which is HTML 5 in this case. Anything written in between **<>** (angle brackets) in HTML is known as start tag.

Line 2

<html>

In this line, **<html>** is the start tag of HTML document.

Line 3

<head>

In this line, **<head>** is the start tag of HTML. It provides the header information about the document.

Line 4

<title>echo</title>

In this line, **<title>** is the start tag and **</title>** is the end tag. Anything written between these tags is the title given to the document. In this case, **echo** is the title given to the document.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Line 5
</head>
In this line, **</head>** is the end tag of HTML. It indicates that the header information of the document ends here. Anything written between **</>** (angle brackets with a forward slash) in HTML is known as end tag.

Line 6
<body>
In this line, **<body>** is the start tag of HTML body which precedes the visible page content.

Line 7
<h1>First PHP program</h1>
In this line, **<h1>** is the start tag and **</h1>** is the end tag of HTML to define the most important heading in the HTML page. In HTML, there are six tags (from **<h1>** to **<h6>**) that are used to define headings in the HTML page. The **<h1>** tag defines the most important heading and **<h6>** defines the least important heading of the web page. This line will display the following as a heading of the web page in the browser:

First PHP program

Line 8
<?php
In this line, **<?php** is the start tag of PHP which is used to start the PHP code.

Line 9
echo "Cadcim Technologies";
In this line, **echo** is the keyword used to display the output in the browser in PHP. This line will display the following in the browser:

Cadcim Technologies

Line 10
echo "<h2>PHP echo statement</h2>";
In this line, **echo** is the keyword used to display the output in the browser in PHP. **<h2>** and **</h2>** are the start and end tags of HTML used to define the heading of the HTML code. This line will display the following in the browser:

PHP echo statement

Line 11
echo "Echo ", "with ", "multiple parameters.";
In this line, **echo** is the keyword used in PHP to display the output in the browser, but in this line, multiple parameters are passed in the single **echo** statement. This line will display the following in the browser:

Echo with multiple parameters.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Line 12
?>
In this line, **?>** is the end tag of PHP which is used to end the PHP code.

Line 13
</body>
In this line, **</body>** is the end tag of HTML. It indicates that the visible page content ends here.

Line 14
</html>
In this line, **</html>** is the end tag of HTML. It indicates that the HTML document ends here.

The output of Example 1 is displayed in Figure 3-1.

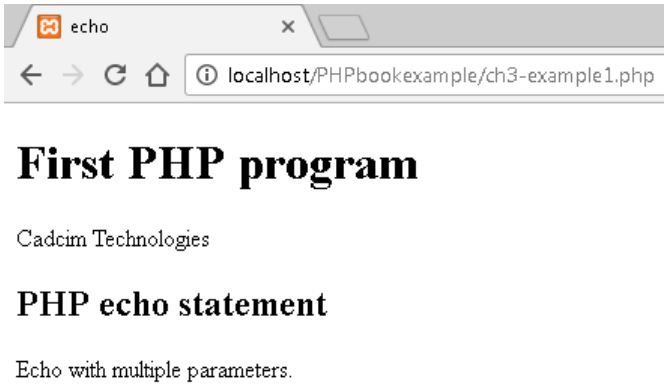

Figure 3-1 *The output of Example 1*

Print Statement

In PHP, the **print** statement can be used in place of the **echo** statement to display output data in the browser. This statement has a return value 1, therefore, it can be used in complex expressions where **echo** statement cannot be used. This statement can be used with parentheses or without parentheses such as **print()** or **print**. You can only pass one argument in **print** statement. This statement does not accept multiple arguments.

For example:

```
<?php
   print "Welcome to CADCIM Technologies";
?>
```

In the given example, **<?php** is the start tag of PHP, **print "Welcome to CADCIM Technologies";** is the **print** statement which will be displayed as output in the browser, and **?>** is the end tag of PHP.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Example 2

The following program will display the output text in the browser by using the **print** statement. The line numbers on the right are not the part of this program and are for reference only.

```
<!Doctype html>                                                  1
<html>                                                           2
<head>                                                           3
<title>print</title>                                             4
</head>                                                          5
<body>                                                           6
   <h1>Second PHP program</h1>                                   7
   <?php                                                         8
         print "Cadcim Technologies";                            9
         print "<h2>PHP print statement</h2>";                   10
   ?>                                                            11
</body>                                                          12
</html>                                                          13
```

Explanation

Line 9

print "Cadcim Technologies";

In this line, **print** is the keyword used to display the output in the browser in PHP. This line will display the following in the browser:

Cadcim Technologies

Line 10

print "<h2>PHP print statement</h2>";

In this line, **print** is the keyword used to display the output in the browser in PHP. The **<h2>** and **</h2>** tags are the start and end tags of HTML used to define the heading of the HTML code. This line will display the following in the browser:

PHP print statement

The output of Example 2 is displayed in Figure 3-2.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

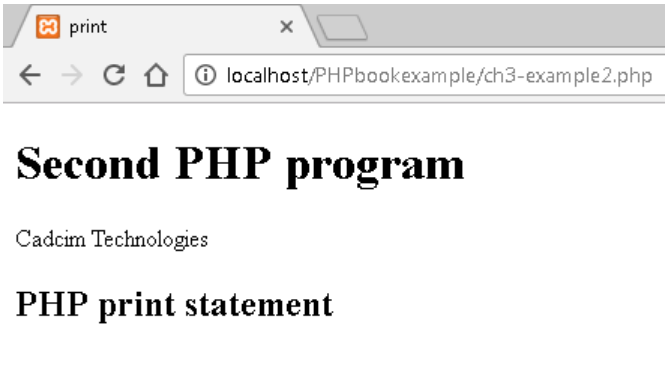

Figure 3-2 *The output of Example 2*

USE OF SPECIAL SYMBOLS

In PHP, some special symbols are used such as semicolon(;) and dollar sign($). They are discussed next.

Use of the ; Symbol

Every PHP command ends with a semicolon(;). It is important to place a semicolon(;) after every PHP command as it indicates that a PHP statement or command ends here. In other words, it terminates the PHP instructions. You do not need to place a semicolon at the last line of a PHP block because the closing tag of PHP block code automatically implies a semicolon.

For example:

```
<?php
   echo "This statement ends with a semicolon";
   echo "Last line of this PHP block does not need a semicolon"
?>
```

In the given example, the last statement does not contain a semicolon; still it would not throw an error as the statement is correct if it is ending with **?>**. But if the end tag of PHP is missing then it will throw an error as the closing tag of PHP automatically implies a semicolon.

For example:

```
<?php
   echo "This statement ends with a semicolon";
   echo "Last line of this PHP block does not ends with a semicolon"
```

Note that this example will throw an error as ?**>** is missing in the code, but this will work properly if the last PHP command ends with a semicolon.

For example:

```
<?php
   echo "This statement ends with a semicolon";
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
echo "Last line of this PHP block needs a semicolon to avoid
error";
```

Tip
It is a good practice to end every PHP statement or instruction with a semicolon and end PHP block with **?>** *tag to avoid errors.*

Use of the $ Symbol

In PHP, the **$** symbol is used for variables. All variables start with a **$** symbol because whenever **$** symbol comes across a variable, it makes the PHP parser faster and defines that any thing written after **$** symbol in PHP is a variable.

Note
A variable is a named location where data is stored. This is a container used to store data and always starts with **$** *symbol. You will learn more about variables in later chapters.*

CASE SENSITIVITY

Case sensitivity means that each letter of a code or text is restricted to follow a proper case of characters whether it is uppercase (capital letter), lowercase (small letter), or both. For example, "Book" is the password set for an account. To open that account it is important to enter password correctly. If user enters "book" instead of "Book" then a message will appear informing that the password is wrong because passwords are case sensitive. In such cases, "book" and "Book" are two different words because "b" is in lowercase in first word and "B" is in uppercase in second word.

In PHP, all the classes, functions, user defined functions, and keywords such as **echo**, **if**, **else**, and so on are not case-sensitive, whereas all the variables and constants are case-sensitive.

Note
You will learn about the classes, functions, user defined functions, variables, constants, and keywords in the later chapters.

Example 3

The following program will display the output text in the browser by using the **echo** statement written in different cases.

```
<!Doctype html>
<html>
<head>
<title>Case sensitivity</title>
</head>
<body>
   <?php
         echo "<h1>echo in lowercase. </h1>";
         ECHO "<h2>ECHO in uppercase. </h2>";
         EcHo "<h3>EcHo in both case. </h3>";
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
    ?>                                                          11
  </body>                                                       12
  </html>                                                       13
```

Explanation

Line 8

echo "<h1>echo in lowercase. </h1>";

In this line, **echo** is the keyword used to display the output in the browser in PHP. **<h1>** and **</h1>** are the start and end tags of HTML used to define the heading of the HTML code. This line will display the following in the browser:

echo in lowercase.

Line 9

ECHO "<h2>ECHO in uppercase. </h2>";

In this line, **ECHO** is the keyword used to display the output in the browser in PHP. **<h2>** and **</h2>** are the start and end tags of HTML used to define the heading of the HTML code. This line will display the following in the browser:

ECHO in uppercase.

Line 10

EcHo "<h3>EcHo in both case. </h3>";

In this line, **EcHo** is the keyword used to display the output in the browser in PHP. **<h3>** and **</h3>** are the start and end tags of HTML used to define the heading of the HTML code. This line will display the following in the browser:

EcHo in both case.

The output of Example 3 is displayed in Figure 3-3.

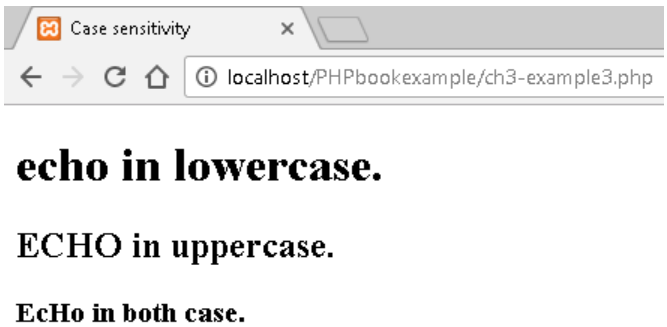

Figure 3-3 *The output of Example 3*

DATA TYPES

While writing a program, you need to store different types of data in variables. Data type describes type of value that can be stored in a variable. Data types are used to construct the variables.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

PHP stores eight types of primitive data types, which are as follows:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

These eight primitive data types are grouped into three different categories:

1. Scalar data types
2. Compound data types
3. Special data types

Scalar Data Types

Scalar data types in PHP are used to represent single value. They are also known as base data types as they provide base for other data types such as array, object, and so on.

In PHP, there are four scalar data types, which are as follows:

- String
- Integer
- Float
- Boolean

String

A string is a sequence of characters where characters are same as bytes. A string can be any text written inside quotes. It can be written inside a single or a double quote. For example, "Welcome to Cadcim".

Before PHP 7, there were some restrictions in length of the string. But now in PHP 7, there are no such restrictions regarding the length of a string on 64-bit builds. On 32-bit builds and in the earlier versions, a string could be as large as up to 2GB (2147483647 bytes maximum).

Example 4

The following program will display the output text in the browser by using string data type.

```
<!Doctype html>
<html>
<head>
<title>String Data type</title>
</head>
<body>
   <?php
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

```
            $x = "Hello User!";                                  8
            $y = 'It is a string.';                              9
            echo $x;                                             10
            echo "<br>";                                         11
            echo $y;                                             12
    ?>                                                           13
</body>                                                          14
</html>                                                          15
```

Explanation

Line 8

$x = "Hello User!";

In this line, **$x** is the variable which is holding a string type data **Hello User!**. This string type data is written inside the double quote.

Line 9

$y = 'It is a string.';

In this line, **$y** is the variable which holds a string type data **It is a string**. This string type data is written inside the single quote.

Line 10

echo $x;

In this line, **echo** will display the value of the **$x** variable which was assigned to it in Line 8. This line will display the following in the browser:

Hello User!

Line 11

echo "<br>";

In this line, **
** is the break tag of HTML used to break the line**.** This line does not display anything in the browser as it is used to break the line only.

Line 12

echo $y;

In this line, **echo** will display the value of the **$y** variable which was assigned to it in Line 9. This line will display the following in the browser:

It is a string.

The output of Example 4 is displayed in Figure 3-4.

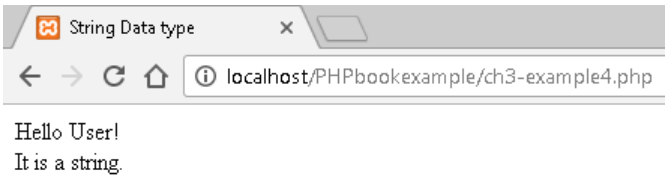

Figure 3-4 *The output of Example 4*

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Integer

The integer data type is used only for those numbers that do not contain any fractional part or decimal point. In other words, this data type is used only for signed whole numbers, either negative or positive.

Rules for integer data type are as follows:

- Data should have at least one digit.
- There cannot be a decimal point in data.
- This can be either positive or negative.
- This can be specified in three formats: decimal (10-based), hexadecimal (16-based -prefixed with 0x), or octal (8-based -prefixed with 0).

For example:

```
42 // decimal (10-based)
0755 // octal (8-based -prefixed with 0)
0xC4E // hexadecimal (16-based -prefixed with 0x)
```

In the given example, **42**, **0755**, and **0xC4e** are integer type values written in different formats as mentioned in the comments represented by double forward slash(//).

Example 5

The following program will display the output text in the browser by using integer data type.

```
<!Doctype html>                                          1
<html>                                                   2
<head>                                                   3
<title>Integer Data type</title>                         4
</head>                                                  5
<body>                                                   6
   <?php                                                 7
         $x = 1233;                                      8
         echo $x;                                        9
         echo "<br>";                                    10
         echo "It is an integer.";                       11
   ?>                                                    12
</body>                                                  13
</html>                                                  14
```

Explanation

Line 8

$x = 1233;

In this line, **1233** is the integer type value assigned to the **$x** variable.

Line 9
echo $x;
In this line, **echo** will display the values of the **$x** variable assigned to it in Line 8. This line will display the following in the browser:

1233

The output of Example 5 is displayed in Figure 3-5.

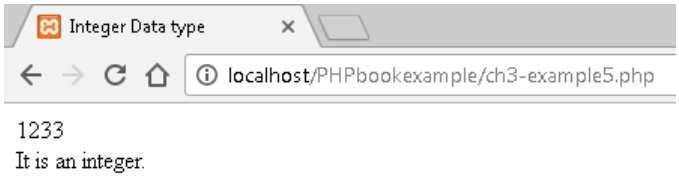

Figure 3-5 *The output of Example 5*

Float

The float data type is used only for those numbers that contain a decimal point or that have a fractional part. All the float, double, or real numbers are the part of float data type in PHP.

For example:

```
$x = 4.20;
```

In the given example, **$x** is the variable and 4.20 is the float type value assigned to it.

Example 6

The following program will display the output text in the browser by using the float data type. The line numbers on the right are not the part of the program and are for reference only.

```
<!Doctype html>                                          1
<html>                                                   2
<head>                                                   3
<title>Float Data type</title>                           4
</head>                                                  5
<body>                                                   6
   <?php                                                 7
         $x = 12.33;                                     8
         echo $x;                                        9
         echo "<br>";                                    10
         echo "It is a float type.";                     11
   ?>                                                    12
</body>                                                  13
</html>                                                  14
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Explanation

Line 8

$x = 12.33;

In this line, **12.33** is the float type value assigned to the **$x** variable.

Line 9

echo $x;

In this line, **echo** will display the value of the **$x** variable which was assigned to it in Line 8. This line will display the following in the browser:

12.33

The output of Example 6 is displayed in Figure 3-6.

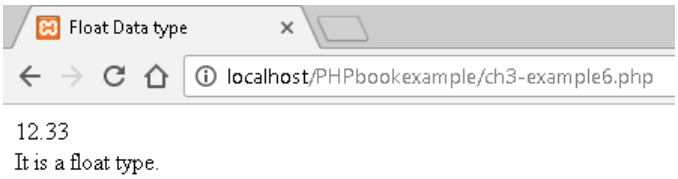

Figure 3-6 *The output of Example 6*

Boolean

A boolean returns a true or false value. Alternatively, you can use zero to represent false and any nonzero value to represent true. Booleans are often used in conditional testing, so to check any condition, a user can use boolean.

Note

You will learn about conditional testing in later chapter.

For example:

```
$x = true;
$y = false;
```

In the above example, **$x** is the variable and **true** is the boolean type value assigned. Similarly, **$y** is the variable and **false** is the boolean type value assigned.

Compound Data Types

Compound data types in PHP are used for multiple items of the same type to be aggregated under a single representative entity. It is a collection of basic data types.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

In PHP, there are two compound data types that are as follows:

- Array
- Object

Array

In PHP, an array is the collection of data elements of the same or different types. These elements are referred by a common name and are stored in contiguous memory locations. You can refer to a particular data element in an array by using the index. The index starts from the lowest address (0) and ends at the highest address (n-1, here n specifies the total number of elements).

For example:

```
<?php
$mobile = array("Samsung", "Nexus", "iPhone");
echo "I like". $mobile[0]. "," . $mobile[1]. "and" . $mobile[2].
   ".";
?>
```

In the given example, **$mobile** is a single variable which holds multiple value or elements, **array()** is the array function used to define the elements of array inside the parentheses. Here, **Samsung**, **Nexus**, and **iPhone** are the elements of the array referred by a common name **$mobile**. Next, **$mobile[0]**, **$mobile[1]**, and **$mobile[2]** are the array index which will print the array element according to the index.

Object

An object is a data type in PHP that stores data and gives information how to process the data. An object is an instance of a class. Class serves as templates for objects and on the basis of these templates, objects are created using the **new** keyword.

An object must be explicitly declared. The declaration of an object's characteristics and behavior takes place within a class. Every object has properties and methods corresponding to those of its parent class. Objects can be manipulated independent of other objects of the same class as every object instance is completely independent.

For example:

```
$obj = new Car;
$temp= new Car('name' , 'model');
```

In the above example, **$obj** and **$temp** are declared as the objects of class **Car** by using **new** keyword. Here, **$obj** has no arguments and **$temp** has two values **name** and **model** which are passed as arguments.

Note

You will learn more about array and object in the later chapters.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Special Data Types

Special data types in PHP are of two types:

- NULL
- Resource

These types of data types are used in special cases in PHP and are discussed next.

NULL

NULL is a special data type which has only one value **NULL**. If a variable is created without a value, then the value **NULL** will be automatically assigned to the variable. You can also empty a variable by setting its value **NULL**.

For example:

```
$x;
$y = NULL;
```

In the above example, both **$x** and **$y** variables are NULL type. The **$x** variable has no value assigned to it. So by default, the **NULL** value is automatically assigned to the **$x** variable.

Example 7

The following program will display the output text in the browser by using NULL type.

```
<!Doctype html>                                          1
<html>                                                   2
<head>                                                   3
<title>NULL type</title>                                 4
</head>                                                  5
<body>                                                   6
   <?php                                                 7
         $x = "Hello User";                              8
         $x = null;                                      9
         echo $x;                                        10
   ?>                                                    11
</body>                                                  12
</html>                                                  13
```

Explanation

Line 8
$x = "Hello User";
In this line, **Hello User** is a string type value assigned to the **$x** variable.

Line 9
$x = NULL;
In this line, **NULL** is the NULL type value assigned to the **$x** variable.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Line 10
echo $x;
In this line, **echo** will display value of the **$x** variable assigned in Line 9.

There will be no output of Example 7 in the browser as the value of **$x** variable has become null (empty) in Line 9.

Resource

A resource type is not an actual data type. It is a special variable which holds a reference to an external resource and therefore referred as special type. These resources are created and used by special functions. They are used to store the reference to access the resources external to PHP file. A common example of using the resource data type is a database call or database connectivity.

Note
You will learn more about resource in later chapter.

TYPE DECLARATION

Type declaration means specifying the expected type of data which will be stored in the variable. It is also known as type hinting. The data type of expected data can be declared during function declaration while passing the parameters to the function.

In PHP versions prior to PHP 5 version, type declaration was not possible. PHP used to set the data type of variables automatically according to the values assigned. The type declaration was introduced in PHP 5 version that allows you to set the data type of the variable. This is used to check that the value passed in the function calling is related to the required data type or not.

PHP 7 version introduces two new features for type declaration that are as follows:

1. Scalar type declaration
2. Return type declaration

Scalar Type Declaration

Scalar type declaration is introduced in PHP 7. Scalar type declaration specifies the data type of input parameters of functions used in PHP program as integer, float, string, or boolean. Keywords used for scalar type declarations are: **int** (for integers), **float** (for floating point numbers), **string** (for strings), and **bool** (for booleans). Scalar type declaration can also specify the type of array and name of an interface. Scalar type declaration helps in finding out mistakes in program by comparing the data type of values passed in the parameters with the expected data type. Scalar type declaration ensures the input consistency and easily eliminate the error in the complex programs.

Scalar type declaration can be done with any of the two modes:

1. Coercive mode(default)
2. Strict mode

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Coercive Mode

Coercive mode is the default mode. This mode is selected by default in a PHP program if the mode of scalar type declaration is not specified.

For example:

```
<?php
function Cmode(int $a){
echo "integer  = " .$a;
}
Cmode(12); //function call
//output: integer = 12
```

In the above example, mode of scalar type declaration is not specified. So this mode is coercive mode which is the default mode. Here, **function** is the keyword used for function in PHP, **Cmode** is the function name, and **(int $a)** is the input parameter. Inside the input parameter, **int** is the keyword used to specify that the function will accept integer type value and **$a** is the variable. When a value is passed to the function, the value will be matched with the data type passed in the input parameters of function and the output will be displayed. In this case, **12** is the integer type value passed to the function **Cmode**. So without any type conversion, the program will display the exact value **12** as output.

If the data type of value passed to the function does not match the data type in the input parameter list of function, the program will not generate an error. Due to coercive mode, the program will convert the value passed to the function into the expected data type specified in the input parameter list of function and display the output.

For example:

```
<?php
function Cmode(int $a){
echo "integer = " .$a;
echo "</br>";
}
Cmode(12.1); //function call
Cmode(True); //function call
/*output: integer = 12
          integer = 1 */
```

In the given example, **(int $a)** is the input parameter where **int** is the keyword used to specify that the function will accept integer type values. Here, **12.1** and **True** are the float and boolean type values passed to the function **Cmode**, respectively. These two values do not match with the **int** data type passed in input parameter of function. Therefore, due to the coercive mode, these values will get converted into integer type and the output will be displayed as 1 for True and 12 for 12.1.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Note

*If the input parameter(s) of a function consists of **int** or **float** data type and the value passed to the function call is of **string** data type then an error will occur because the **string** data type cannot get converted into integer or float type. But, if the input parameter(s) of the function consists of **string** data type and the value passed to the function call is of integer type then it will accept integer as string.*

Strict Mode

When a developer specifies the mode of scalar type then that mode is known as strict mode. It gets enabled when developer places a single declare directive at the top of the PHP program file. This directive ensures that any function call made in the program will have to strictly follow the specified data type.

The single declare directive to be placed at the top of the program file is:

```
declare(strict_types=1);
```

For example:

```
<?php
declare(strict_types=1);
function Smode(int $a){
echo $a;
}
Smode(True); //Function call
 /* Output: Fatal error: Uncaught TypeError: Argument 1 passed to
   Smode() must be of the type integer, boolean given ..*/
```

In the above example, **declare(strict_types=1);** is the single declare directive which makes the program in strict mode. Its working is same as it is in coercive mode with only difference that it does not perform type conversion as coercive does. If the data type in input parameter of function does not match with the value passed in the function, it will throw a fatal error instead of converting it in that type. In the given example, it will throw fatal error because value passed in function **Smode** is **True** which is of boolean type and it does not match with data type **int** (keyword for integer data type) which is passed in the input parameter of the function.

Return Type Declaration

Return type declaration is also introduced in PHP 7. It specifies the type of value that a function should return. It is available in most of the popular programming languages such as C, C++, Java, and so on, but it was not available before PHP 7. Return type declaration ensures output consistency.

It works with coercive and strict mode. In coercive mode, if the value gets matched with the data type passed for return type then it will return the exact value; otherwise it will convert it into the given type and display the output. In strict mode, it will display the output only if it returns the value with the correct data type, otherwise it will throw a fatal error, specifically TypeError. To specify the return type, colon is added in front of the function input parameter and the return data type is specified next to it.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

For example:

```
function returnIntValue(int $value): int {
         //block of function.
      return $value;
   }
```

In the above example, **function** is the keyword used for function, **returnIntValue** is the function name, **(int $value)** is the function input parameter where **int** is the keyword used to declare the scalar type and **$value** is the variable, **:** (colon) is used before specifying the return data type, and **int** is the keyword used to return integer data type. In the block of function, **return** is the keyword used to return a value.

Self-Evaluation Test

Answer the following questions and then compare them to those given at the end of this chapter:

1. In PHP, _____ and _____ statements are used to display output data in the browser.
2. The _____ statement cannot take multiple arguments.
3. The _____ type in PHP is used to represent single value.
4. The _____ type is a number with a decimal point.
5. _____ and _____ type declarations are introduced in PHP 7.

Review Questions

Answer the following questions:

1. Which of the following options is the correct basic syntax of PHP?

   (a)
   ```
   <?php
            //PHP Code
      ?>
   ```

   (b)
   ```
   <php
            //PHP Code
      ?>
   ```

   (c)
   ```
   <!--php
            //PHP Code
      -->
   ```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

(d)
```
<php>
        //PHP Code
</php>
```

2. Which of the following symbols is used to end the PHP command?

(a) **$** (b) *****
(c) **?** (d) **;**

3. Which of the following options is the correct example of integer data type?

(a) 45.70 (b) TRUE
(c) "45" (d) 15

4. Which of the following options is the single declare directive which is placed at the top of the program file to declare the strict mode?

(a) `declare(strict_type=1);` (b) `declare(strict_types=1);`
(c) `declare(strict_mode=1);` (d) `declare(strict_types=on);`

5. **Find errors in the following blocks of source code:**

(a)
```
<?php
echo "<h1>Find Error</h1>;
echo "This","is","string","made,"with","multiple parameters.";
?>
```

(b)
```
<?php
/* find an error in the multi line comment, this is line 1
      /* this is line 2, of PHP multi line comment */
            and this is line 3 where the comment ends */
?>
```

(c)
```
<?php
      $x = "Hello User!";
      $y = It is a string.;
      echo $x;
      echo "<br>";
      echo $y;
```

(d)
```
<?php
      $x = "Hello User!";
      $y = 'you found the error.';
      echo $x;
      echo "<br>"
      echo $y;
?>
```

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

(e)
```
<!doctype html>
<html>
<head>
<title> Program
<body>
      <?php
            print "Hello User";
            print "How are you";
      ?>
</body>
</html>
```

EXERCISES

Exercise 1

Write a program to display the string value "This is the String type scalar declaration" in the browser using strict mode of scalar type declaration.

Exercise 2

Write a program to display any float and integer numbers in the browser using the **echo** statement in coercive mode of scalar type declaration where input parameter function must be of float type.

Evaluation Copy. Do not reproduce. For information visit www.cadcim.com

Answers to Self-Evaluation Test

1. print, **echo**, **2. Print**, **3.** Scalar, **4.** Float, **5.** Scalar, return