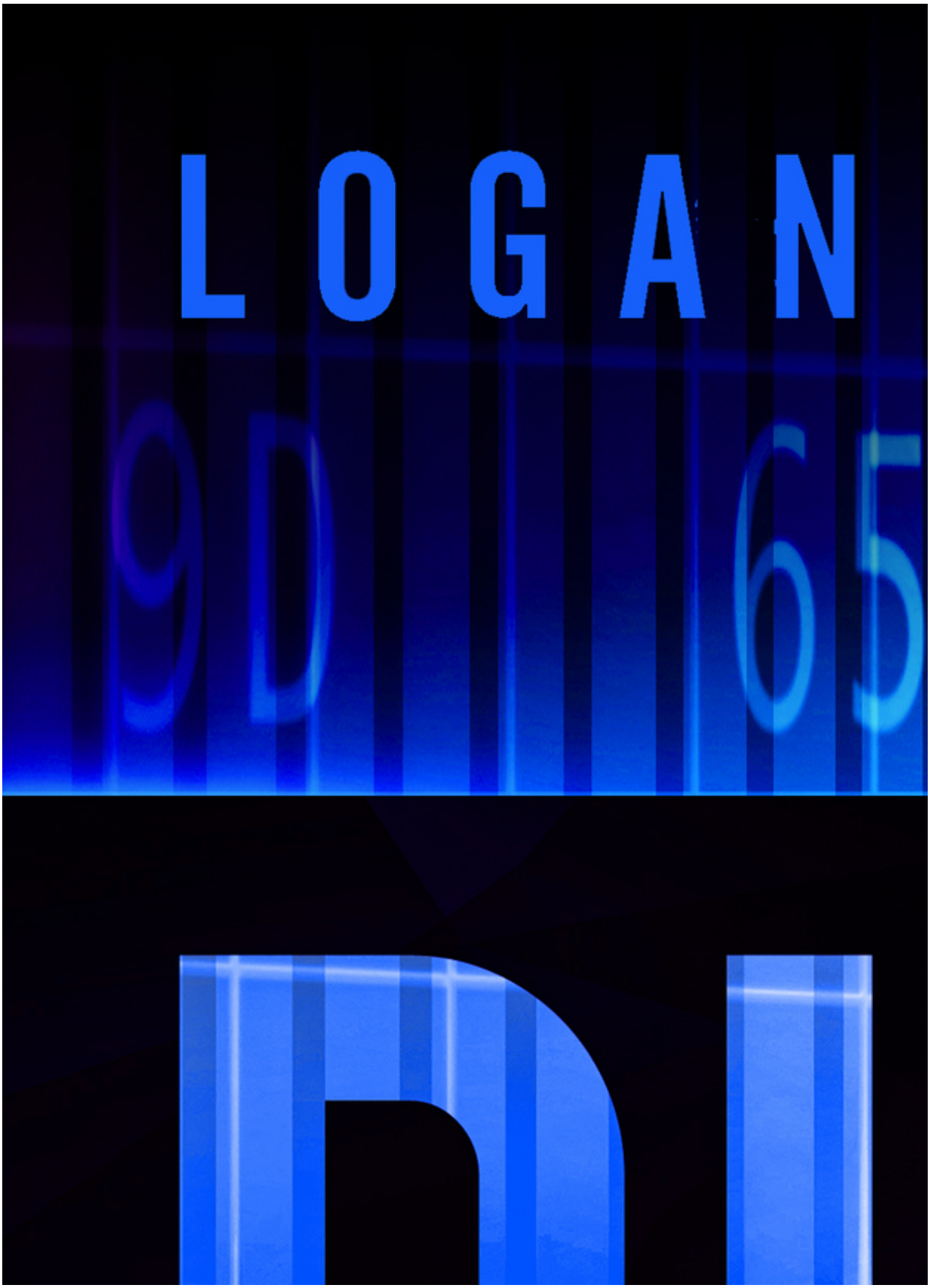
LOGAN PRATT

PHP

Advanced Guide to Learn the Realms of PHP Programming


LOGAN
9D 65

PHP

Advanced Guide to Learn the Realms of PHP Programming

Table of Content

Introduction

This book on PHP programming contains proven steps and strategies on how to code in PHP and build websites. I have provided functional codes so that you can study the details, understand how it works, and start practicing right away. You can load up the codes to directly display on the webpage as each code is already embedded in an HTML page.

The best way to work with the codes is to customize them by introducing the changes you want to make, and then loading them up on the webpage. PHP is an interesting language but it is technical as well. You may see errors popping up everywhere on the editor. Therefore, you should study the codes that are given in the book carefully. Start by pasting as they are written into the editor, and then load them up to display the web page. The most technical part is the section on object-oriented programming. However, with a bit of extra focus, you can master it as well.

Programming is about how much you practice in the text editor. The more you write a code in different styles and with different values, the better is the chance that you can master it.

Chapter One: The First Steps to Learning PHP

There are many reasons to create your programs in PHP. You may be interested in learning PHP because you want to put together a website with some interactive elements.

PHP in the Web World

PHP is used to building website. Instead of a PHP program that runs on a desktop computer for one person’s use, it runs typically on a web server and is accessed by lots of people who are using web browsers on their computers. This section will explain how PHP adjusts itself into the interaction between a web server and a web browser.

When you sit at your computer and also pull up a page with the help of a browser like Firefox or Safari, you may cause a little conversation to break out over the internet between your and anyone else’s computer. Web server reads your programs that you have written in PHP. The programs are like simple instructions that you have communicated to the web server through PHP. The web server figures out what to do. PHP refers to either the interpreter or the programming language.

PHP is referred to as a server-side language because it tends to run on a particular web server. Technologies and languages like Flash and JavaScript are known as client-side because they run on desktop PC or a web client. The instructions that are given in a PHP program cause the interpreter to output as a web page. The instructions trigger the web browser to do something like popping up a new window. Once the web browser sends the generated web page to client, PHP seems to be out of picture. If the page's content contains JavaScript, the JavaScript runs but stays disconnected from the PHP program that has generated the page.

A dynamic page that is generated by PHP is just like a postal letter that you write to a friend across the world. You can add to the page whatever is on your mind and send it off to your friend on the other side of the world. The content of the letter is tailored for a specific person to whom you are sending it. Once the letter is enveloped inside the mailbox, you cannot change it. It will immediately fly off to the destination. You cannot modify it as your friend reads it.

Why PHP?

PHP is free to use. Whether you run the interpreter on an old PC or a million dollar super computer, you do not have to pay any licensing fees, maintenance fees, upgrade fees, and support fees. Most of the Linux distributions have got PHP preinstalled on them. If your system does not have that, you may be using another operating system like Windows.

As one of the open source projects, PHP allows anyone to inspect it. Even if you lack technical expertise in PHP, you can deploy someone to do the investigation for you.

Installing PHP

In the following section, I will explain different steps for installing PHP on your computer operating system.

Installation of a Web Server

You cannot just run PHP in the browser as you do with HTML. If you want to run PHP, you will need a server installed on your computer system so that the browser can parse PHP scripts and then run several MySQL queries to the database on it to return the results. You also can use a host that supports MySQL and PHP so that they may upload it to the server and allow you to test your website. You can install different servers on your computer system like WAMP, XAMP, and MAMP. I will explain the installation process of XAMP that is suitable for Windows operating systems. It is a completely free open-source platform web server package for Windows, Linux, and Max.

The first step in the process is to download the latest version of XAMP. Once you have downloaded that, you have to locate the file on your local machine and then double-click on that to kick off the installation process. Your antivirus system likely interferes with the process. You might need to switch it off until the server finishes installing it to avoid potential problems.

Text Editor

You will need a text editor to write PHP code. Any text editor will work for you. You can even use Notepad to write the code. However, Notepad++ will help you write cleaner and more efficient code in a faster way. It will

highlight the syntax and it has the auto complete process. The text editors will save you time and pain. Now that you have installed XAMPP, you should start apache web server for the browser to run your scripts. Click open the Control Panel of XAMPP. It will start Apache.

Each PHP script is embedded inside an HTML page because PHP scripts are used to build website pages.

PHP Variables

If you want to build a website with PHP, you will need lots of data and know how to store it. The data may be a name, a date, a picture, or a number. PHP offers you a way to store data with the help of a variable. Each variable in PHP has the $ sign in front of it. There will be no spaces in between the name of the variable and the sign.

```
<html>
<head>
<title>This is a PHP web page</title>
</head>
<body>
<?php
   $txt = "I am learning PHP scripts to build my webpage.";
   echo $txt;
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
<title>This is a PHP web page</title>
</head>
```

```
<body>
I am learning PHP scripts to build my webpage.</body>
</html>
```

The above code has two parts. The first part is the written script that I have created inside a text editor, and the second part starts at $PHP main.php line. This part contains the result displayed on the server side of the webpage when you execute a PHP script. All the codes in the book will have two parts so that you can better understand how the results on the server side look like.

Variables are case sensitive, which means $txt is different from $Txt. Also, they must start with an underscore or a letter. They cannot start with numbers and they may only contain alphanumeric values.

PHP is a typed language so you do not have to define the data types for writing variables. PHP will convert variables to correct data type by examining the value that you have assigned to it.

Some common data types are integers, strings, floats, arrays, Booleans, and objects.

String Concatenation

In this script, I will explain how you can combine two PHP scripts. I will join two strings by using period (.).

```
<html>
<head>
<title>This is a PHP web page</title>
</head>
<body>
<?php
   $txt = "I am learning PHP scripts to build my webpage.";
   echo "Hey Stephen! " .$txt. " You may join me if you have time.";
?>
```

</body>

</html>

$php main.php

<html>

<head>

<title>This is a PHP web page</title>

</head>

<body>

Hey Stephen! I am learning PHP scripts to build my webpage. You may join me if you have time.</body>

</html>

PHP Constants

Constants are like variables except for the fact that you cannot change their values. When you define a constant, you use define () method then allocate it a name and value. The following example shows how you can define a constant.

<html>

<head>

<title>This is a PHP web page</title>

</head>

<body>

<?php

define("msg", "I am learning PHP scripts to build my webpage.");

echo msg;

?>

</body>

</html>

$php main.php

<html>

<head>

<title>This is a PHP web page</title>

</head>

<body>

I am learning PHP scripts to build my webpage.</body>

</html>

You can specify if you want the constant to be insensitive of the case by the addition of one parameter. Otherwise, the case will be sensitive. I will run the above script by changing the case of the echo command.

```
<html>
<head>
<title>This is a PHP web page</title>
</head>
<body>
<?php

   define("msg", "I am learning PHP scripts to build my webpage.");
   echo MSG;
?>
</body>
</html>
```

$php main.php

<html>

<head>

<title>This is a PHP web page</title>

</head>

<body>

MSG</body>

</html>

PHP Notice: Use of undefined constant MSG - assumed 'MSG' in /home/cg/root/7025528/main.php on line 9

Now I will add another parameter to the script that takes a Boolean. The script will work well even if you change the case.

```
<html>
<head>
<title>This is a PHP web page</title>
</head>
<body>
<?php

   define("msg", "I am learning PHP scripts to build my webpage.", true);
   echo MSG;
?>
</body>
</html>
```

$php main.php

<html>

<head>

<title>This is a PHP web page</title>

</head>

<body>

I am learning PHP scripts to build my webpage.</body>

</html>

Constants must start with an underscore and a sign. They should not start with the $sign.

PHP Operators

PHP uses different operators for performing a diverse range of operations of values and variables. We will now look at different operators being used by PHP in the form of groups.

Some operators will perform certain mathematical operations on values and will return calculated results.

- $a + $b is used for deriving the sum of the two variables.
- $a - $b is used for deriving the difference of the two variables.
- $a * $b is used for deriving the multiplication of the two variables.
- $a / $b is used for deriving the division of the two variables.
- $a / $b is used for deriving the remainder of the two variables.
- $a ** $b is used for deriving the exponent the two variables.
- $a = $b is used for getting the value of $b and setting it to $a.

Chapter Two: Expressions and Control

The preceding chapter introduced you to the basics of PHP like the installation framework and PHP variables. This chapter will walk you through how PHP programming works in and how you can control the flow of a program.

Expressions

I will now start with the fundamental part of PHP. Expressions are combinations of values, operators, variables, and functions that result in a value. You can write an expression by combining Boolean operators such as OR, AND, and XOR. In the next example, I will explain simple expressions in the following blocks of code.

```
<html>
<head>
<title>This is a PHP web page</title>
</head>
<body>
<?php

echo "e: [" . (80 > 9) . "]\n";
echo "f: [" . (10 == 6) . "]\n";
echo "g: [" . (5 == 0) . "]\n";
echo "i: [" . (2 == 2) . "]\n";
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
```

<title>This is a PHP web page</title>
</head>
<body>
e: [1]
f: []
g: []
i: [1]
</body>
</html>

You can get the output of true and false as well by using the following code.

```
<html>
<head>
<title>This is a PHP web page</title>
</head>
<body>
<?php

echo "e: [" . TRUE . "]\n";
echo "f: [" . FALSE . "]\n";
?>
</body>
</html>
```

$php main.php

<html>
<head>
<title>This is a PHP web page</title>
</head>

<body>

e: [1]

f: []

</body>

</html>

In certain languages, FALSE may be defined as -1 or zero therefore in other languages, you should verify the definition of this Boolean expression.

Relational Operators

Relational operators are fed with two operands. It tests them and returns a Boolean result as to whether it is FALSE or TRUE. Three types of relational operators logical, equality, and comparison are as under:

Equality

The equality operator is ==. It is crucial not to confuse it with the single equal sign assignment operator. The first statement assigns a particular value and then the second tests it to check equality in the expression.

<html>

<head>

<title>This is a PHP web page</title>

</head>

<body>

<?php

$month = "June";

if ($month == "June") echo "The summer season has arrived. It is time to go to the beach.";

?>

</body>

</html>

$php main.php

```
<html>
<head>
<title>This is a PHP web page</title>
</head>
<body>
The summer season has arrived. It is time to go to the beach.</body>
</html>
```

By returning TRUE or FALSE, the equality operator enables you to test certain conditions with the if statement's help. However, PHP is considered as a loosely typed language. If the two operands of equality expression belong to different types, PHP will convert them into whatever makes sense. Any strings that are composed of numbers will be converted into numbers if you compare them with numbers.

Comparison Operators

By using comparison operators, you can test more than equality and inequality. PHP gives you the options of greater than, lesser than options to work with while you build your website.

```
<html>
<head>
<title>This is a PHP web page</title>
</head>
<body>
<?php
  $x = 20; $y = 35;
if ($x > $y) echo "$x is greater than $y\n";
if ($x < $y) echo "$x is less than $y\n";
if ($x >= $y) echo "$x is greater than or equal to $y\n";
if ($x <= $y) echo "$x is less than or equal to $y\n";
```

?>

</body>

</html>

$php main.php

<html>

<head>

<title>This is a PHP web page</title>

</head>

<body>

20 is less than 35

20 is less than or equal to 35

</body>

</html>

Logical Operator

Logical operators produce either false or true results therefore they are also known as Boolean operators. There are four of these operators.

- AND
- OR
- XOR
- NOT

<html>

<head>

<title>This is a PHP web page</title>

</head>

<body>

<?php

```
$x = 1; $y = 0 ;
 echo ($x AND $y) . "\n";
echo ($x or $y) . "<\n";
echo ($x XOR $y) . "\n";
echo !$x . "\n";

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

$php main.php

```
<html>
<head>
<title>This is a PHP web page</title>
</head>
<body>

1<
1

</body>
</html>
```

PHP Conditionals

PHP conditionals alter the flow of the program. They will enable you to ask certain questions about different things and then respond to answers in multiple ways. Conditionals have a center stage on dynamic website pages because they make it easy to get different output each time a viewer visits your webpage. There are three types of conditionals; if statement, the ? operator and the switch statement.

The if Statement

One way to control the flow of your programs is using the if statement. The contents of an if statement may be a valid PHP expression including comparison, equality, and the values that functions return. The actions you want PHP to take when the if condition is TRUE need to be placed inside curly braces {}. You may ignore the braces if you only have one statement to execute. However, if you always use curly braces, you may avoid hunting down hard bugs.

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
<?php
if ($my_account_balance < 1000)
{
    $my_money = 10000;
$my_account_balance += $my_money;
}

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

The else Statement

When a conditional statement is not TRUE, you may need to continue to the main code but you might wish to do something else as well. This is the point where the else statement is put to work. With the help of an if-else statement, the first conditional statement stands executed if the condition is TRUE. If it is FALSE, the second one stands executed. One of the two

choices ought to be executed. Under the current circumstances, the two of them are executed. The following example shows the if-else structure.

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
<?php
$greetings_time = date("H");

if ($greetings_time < "12") {
  echo "I wish you a very good day!";
} else {
  echo "I wish you have an excellent night!";
}

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

$php main.php

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
I wish you a very good day!</body>
</html>
```

The if….elseif….else Statement

This is longer than the first two and executes multiple codes for more than two conditions. It works well to test multiple possibilities. See how the code works in the following example.

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
<?php
$greetings_time = date("H");

if ($greetings_time < "12") {
  echo "I wish you a very good morning!";
} elseif($greetings_time < "20") {
    echo "I wish you have a very good noon!";
}

else {
  echo "I wish you have an excellent night!";
}

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

$php main.php

```
<html>
<head>
```

```
<title>This is a PHP Script</title>
</head>
<body>
I wish you a very good morning!</body>
</html>
```

I will now change the time to test whether other code blocks work. I will fill in the code with 10 for the clock.

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
<?php
$greetings_time = date("10");

if ($greetings_time < "12") {
  echo "I wish you a very good morning!";
} elseif($greetings_time < "20") {
    echo "I wish you have a very good noon!";
}
else {
  echo "I wish you have an excellent night!";
}

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

$php main.php

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
I wish you a very good morning!</body>
</html>
```

Now I will add 15 for the clock.

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
<?php
$greetings_time = date("15");

if ($greetings_time < "12") {
  echo "I wish you a very good morning!";
} elseif($greetings_time < "20") {
    echo "I wish you have a very good noon!";
}
else {
  echo "I wish you have an excellent night!";
}

?>
```

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

$php main.php

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
I wish you have a very good noon!</body>
</html>
```

In the last example, I will add more than 23 to the clock timer.

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
<?php
$greetings_time = date("23");

if ($greetings_time < "12") {
  echo "I wish you a very good morning!";
} elseif($greetings_time < "20") {
    echo "I wish you have a very good noon!";
}
else {
  echo "I wish you have an excellent night!";
```

```
}

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

$php main.php

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
I wish you have an excellent night!</body>
</html>
```

The Switch Statement

The switch statement comes handy when you are dealing in cases in which a variable and the result of a single expression may have multiple values, which should trigger a different function. Take the example of a code in which a user has to select different cities from a list. If you write by using the if statements, it will look like the following:

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
<?php
$states = "Texas";

switch ($states)
```

```
{
case "Texas":
echo "I have visited Texas in the United States";
break;
case "California":
echo "I have visited California in the United States";
break;
case "Los Angeles":
echo "I have visited Los Angeles in the United States";
break;
case "Florida":
echo "I have visited Florida in the United States";
break;
case "New York":
echo "I have visited New York in the United States";
break;
case "Las Vegas":
echo "I have visited Las Vegas in the United States";
case "Virginia":
echo "I have visited Virginia in the United States";
break;
default:
    echo "I have not visited any state in the United States which means I have not yet visited the United States.";
}

?>
```

</body>

</html>

$php main.php

<html>

<head>

<title>This is a PHP Script</title>

</head>

<body>

I have visited Texas in the United States</body>

</html>

In the next example, I will change the name of the state and see how the code works.

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
<?php
$states = "Las Vegas";

switch ($states)
{
case "Texas":
echo "I have visited Texas in the United States";
break;
case "California":
```

```
echo "I have visited California in the United States";
break;
case "Los Angeles":
echo "I have visited Los Angeles in the United States";
break;
case "Florida":
echo "I have visited Florida in the United States";
break;
case "New York":
echo "I have visited New York in the United States";
break;
case "Las Vegas":
echo "I have visited Las Vegas in the United States";
break;
case "Virginia":
echo "I have visited Virginia in the United States";
break;
default:
    echo "I have not visited any state in the United States which means I have not yet visited the United States.";
}

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

$php main.php

```
<html>
```

```
<head>
<title>This is a PHP Script</title>
</head>
<body>
I have visited Las Vegas in the United States</body>
</html>
```

Now I will fill in the code with no name.

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
<?php
$states = "";

switch ($states)
{
case "Texas":
echo "I have visited Texas in the United States";
break;
case "California":
echo "I have visited California in the United States";
break;
case "Los Angeles":
echo "I have visited Los Angeles in the United States";
break;
```

```
case "Florida":
echo "I have visited Florida in the United States";
break;
case "New York":
echo "I have visited New York in the United States";
break;
case "Las Vegas":
echo "I have visited Las Vegas in the United States";
break;
case "Virginia":
echo "I have visited Virginia in the United States";
break;
default:
    echo "I have not visited any state in the United States which means I have not yet visited the United States.";
}

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

$php main.php

```
<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
```

```
I have not visited any state in the United States which means I have not yet visited the United States.</body>

</html>
```

I mentioned $states just once at the start of the statement. After that, the case command checked for the matches in the code block. When one happens, the matching conditional statement was executed. With the help of the switch statements, you never use curly braces in the case commands. Instead, they start with a colon and conclude with the break statement. The complete list of the cases inside the switch statement is packed up within curly braces.

If you desire to break out of the switch statement, you can use the break command. The commands ask PHP to break out of the statement and jump over to the next statement.

A default action must be added to a switch statement so that if none of the case conditions is fulfilled, the code falls back on that. You can see that in the latest example that I have included above.

Alternative Syntax

There is an alternative syntax that you can use to get the same results. You can replace the curly braces with a single colon and endswitch command. It will work just as fine as it did before.

```
<html>

<head>

<title>This is a PHP Script</title>

</head>

<body>

<?php

$states = "New York";


switch ($states):
```

```
case "Texas":
echo "I have visited Texas in the United States";
break;
case "California":
echo "I have visited California in the United States";
break;
case "Los Angeles":
echo "I have visited Los Angeles in the United States";
break;
case "Florida":
echo "I have visited Florida in the United States";
break;
case "New York":
echo "I have visited New York in the United States";
break;
case "Las Vegas":
echo "I have visited Las Vegas in the United States";
break;
case "Virginia":
echo "I have visited Virginia in the United States";
break;
default:
    echo "I have not visited any state in the United States which means I have not yet visited the United States.";
endswitch;

?>
```

</body>
</html>

$php main.php

<html>
<head>
<title>This is a PHP Script</title>
</head>
<body>
I have visited New York in the United States</body>
</html>

Chapter Three: PHP Looping

One of the best things about computers is that they tend to repeat some calculating tasks tirelessly and fast. You might want a software to repeat some code until you deduce something out of that. PHP has different types of looping structures that provide the perfect way to doing that. Once you enter a code, you can only get out of it, if the conditions are right.

While Loops

The while loop runs a block of code multiple times as long as a certain condition stays true. The following code will does just the same. It will run as long as the condition I will set up remains true.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
    $a = 1;
while ($a <= 25) {
echo "The requisite value of the variable a should be: " . $a . "\n";
$a++;
}
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
```

</head>
<body>
The requisite value of the variable a should be: 1
The requisite value of the variable a should be: 2
The requisite value of the variable a should be: 3
The requisite value of the variable a should be: 4
The requisite value of the variable a should be: 5
The requisite value of the variable a should be: 6
The requisite value of the variable a should be: 7
The requisite value of the variable a should be: 8
The requisite value of the variable a should be: 9
The requisite value of the variable a should be: 10
The requisite value of the variable a should be: 11
The requisite value of the variable a should be: 12
The requisite value of the variable a should be: 13
The requisite value of the variable a should be: 14
The requisite value of the variable a should be: 15
The requisite value of the variable a should be: 16
The requisite value of the variable a should be: 17
The requisite value of the variable a should be: 18
The requisite value of the variable a should be: 19
The requisite value of the variable a should be: 20
The requisite value of the variable a should be: 21
The requisite value of the variable a should be: 22
The requisite value of the variable a should be: 23
The requisite value of the variable a should be: 24

```
The requisite value of the variable a should be: 25
</body>
</html>
```

You can see that the loop only stopped when the condition stood false. We can change the while loop by slicing ++$a off the last line and putting it beside the while keyword in the parenthesis. That is how it becomes a part of the conditional expression of the while loop. PHP checks the $a variable at the start of all while loop iterations after observing that it first increments the variable and compares it to its value. The only difference will be that the loop will kick off from two. To adjust it to one, you will have to make adjustments in the variable's initial value.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
    $a = 0;
while ($a++ <= 25) {
echo "The requisite value of the variable a should be: " . $a . "\n";

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

$php main.php

```
<html>
<head>
```

<title>This is my PHP Script</title>
</head>
<body>
The requisite value of the variable a should be: 1
The requisite value of the variable a should be: 2
The requisite value of the variable a should be: 3
The requisite value of the variable a should be: 4
The requisite value of the variable a should be: 5
The requisite value of the variable a should be: 6
The requisite value of the variable a should be: 7
The requisite value of the variable a should be: 8
The requisite value of the variable a should be: 9
The requisite value of the variable a should be: 10
The requisite value of the variable a should be: 11
The requisite value of the variable a should be: 12
The requisite value of the variable a should be: 13
The requisite value of the variable a should be: 14
The requisite value of the variable a should be: 15
The requisite value of the variable a should be: 16
The requisite value of the variable a should be: 17
The requisite value of the variable a should be: 18
The requisite value of the variable a should be: 19
The requisite value of the variable a should be: 20
The requisite value of the variable a should be: 21
The requisite value of the variable a should be: 22
The requisite value of the variable a should be: 23

The requisite value of the variable a should be: 24

The requisite value of the variable a should be: 25

The requisite value of the variable a should be: 26

</body>

</html>

do…while Loops

This is a slight variation to the while loop. It is used when you need a block of code to be executed one time and then pushed into a certain condition. See the following code sample.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
    $a = 0;
    do
echo "The requisite value of the variable a should be: " . $a . "\n";
while ($a++ <= 25)

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
```

</head>
<body>
The requisite value of the variable a should be: 0
The requisite value of the variable a should be: 1
The requisite value of the variable a should be: 2
The requisite value of the variable a should be: 3
The requisite value of the variable a should be: 4
The requisite value of the variable a should be: 5
The requisite value of the variable a should be: 6
The requisite value of the variable a should be: 7
The requisite value of the variable a should be: 8
The requisite value of the variable a should be: 9
The requisite value of the variable a should be: 10
The requisite value of the variable a should be: 11
The requisite value of the variable a should be: 12
The requisite value of the variable a should be: 13
The requisite value of the variable a should be: 14
The requisite value of the variable a should be: 15
The requisite value of the variable a should be: 16
The requisite value of the variable a should be: 17
The requisite value of the variable a should be: 18
The requisite value of the variable a should be: 19
The requisite value of the variable a should be: 20
The requisite value of the variable a should be: 21
The requisite value of the variable a should be: 22
The requisite value of the variable a should be: 23

The requisite value of the variable a should be: 24

The requisite value of the variable a should be: 25

The requisite value of the variable a should be: 26

</body>

</html>

The only thing you should notice is that the loop starts from 0 instead of 1. This is because PHP editor executes the code immediately without the opportunity to increment the variable. The code, except that, looks quite similar. You will have to use curly braces if you have multiple statements to fill into the code block.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
    $a = 0;
    do {
echo "The requisite value of the variable a should be: " . $a . "\n";
echo "The do...while loop will go on.\n";
}
while ($a++ <= 25)

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

$php main.php

<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
The requisite value of the variable a should be: 0
The do...while loop will go on.
The requisite value of the variable a should be: 1
The do...while loop will go on.
The requisite value of the variable a should be: 2
The do...while loop will go on.
The requisite value of the variable a should be: 3
The do...while loop will go on.
The requisite value of the variable a should be: 4
The do...while loop will go on.
The requisite value of the variable a should be: 5
The do...while loop will go on.
The requisite value of the variable a should be: 6
The do...while loop will go on.
The requisite value of the variable a should be: 7
The do...while loop will go on.
The requisite value of the variable a should be: 8
The do...while loop will go on.
The requisite value of the variable a should be: 9
The do...while loop will go on.
The requisite value of the variable a should be: 10

The do...while loop will go on.
The requisite value of the variable a should be: 11
The do...while loop will go on.
The requisite value of the variable a should be: 12
The do...while loop will go on.
The requisite value of the variable a should be: 13
The do...while loop will go on.
The requisite value of the variable a should be: 14
The do...while loop will go on.
The requisite value of the variable a should be: 15
The do...while loop will go on.
The requisite value of the variable a should be: 16
The do...while loop will go on.
The requisite value of the variable a should be: 17
The do...while loop will go on.
The requisite value of the variable a should be: 18
The do...while loop will go on.
The requisite value of the variable a should be: 19
The do...while loop will go on.
The requisite value of the variable a should be: 20
The do...while loop will go on.
The requisite value of the variable a should be: 21
The do...while loop will go on.
The requisite value of the variable a should be: 22
The do...while loop will go on.
The requisite value of the variable a should be: 23

The do...while loop will go on.

The requisite value of the variable a should be: 24

The do...while loop will go on.

The requisite value of the variable a should be: 25

The do...while loop will go on.

The requisite value of the variable a should be: 26

The do...while loop will go on.

</body>

</html>

for Loops

There is another loop namely the for loop in PHP coding. It is the most powerful loop because it allows you to set up variables inside the loop, test for certain conditions while you are iterating, and modify the variables for different iterations. The first part of the for loop is initialization. This is where you can declare a variable to initiate the loop counter. The second part of the for loop is the condition where we may check if the loop counter matches a certain condition or not. When it returns false value, the loop stops the execution of code block. When the loop reaches its end, it starts repeating itself. This is the point at which you can conduct operations to loop counter. See the following code example.

```
<html>

<head>

<title>This is my PHP Script</title>

</head>

<body>

<?php

    for ($a = 1; $a <= 25; $a++) {

echo "The requisite value of the variable a should be: " . $a . "\n";
```

```
echo "The do...while loop will go on.\n";
}

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
The requisite value of the variable a should be: 1
The do...while loop will go on.
The requisite value of the variable a should be: 2
The do...while loop will go on.
The requisite value of the variable a should be: 3
The do...while loop will go on.
The requisite value of the variable a should be: 4
The do...while loop will go on.
The requisite value of the variable a should be: 5
The do...while loop will go on.
The requisite value of the variable a should be: 6
The do...while loop will go on.
The requisite value of the variable a should be: 7
The do...while loop will go on.
The requisite value of the variable a should be: 8
```

The do...while loop will go on.
The requisite value of the variable a should be: 9
The do...while loop will go on.
The requisite value of the variable a should be: 10
The do...while loop will go on.
The requisite value of the variable a should be: 11
The do...while loop will go on.
The requisite value of the variable a should be: 12
The do...while loop will go on.
The requisite value of the variable a should be: 13
The do...while loop will go on.
The requisite value of the variable a should be: 14
The do...while loop will go on.
The requisite value of the variable a should be: 15
The do...while loop will go on.
The requisite value of the variable a should be: 16
The do...while loop will go on.
The requisite value of the variable a should be: 17
The do...while loop will go on.
The requisite value of the variable a should be: 18
The do...while loop will go on.
The requisite value of the variable a should be: 19
The do...while loop will go on.
The requisite value of the variable a should be: 20
The do...while loop will go on.
The requisite value of the variable a should be: 21

The do...while loop will go on.

The requisite value of the variable a should be: 22

The do...while loop will go on.

The requisite value of the variable a should be: 23

The do...while loop will go on.

The requisite value of the variable a should be: 24

The do...while loop will go on.

The requisite value of the variable a should be: 25

The do...while loop will go on.

</body>

</html>

A semi colon separates all the expressions in the for loop. At the start of the maiden iteration of the for loop, the initialization expression stands executed. The count starts from 0. Each time the condition is tested, the loop goes on after finding the condition to be true. Finally, at the end of iterations, the modification expression stands executed. The for loop is designed for a single value that will keep changing regularly.

for each Loop

A loop that iterates through each element that is found inside an array is known as foreach loop.

<html>

<head>

<title>This is my PHP Script</title>

</head>

<body>

<?php

$states = array("New York", "Los Angles", "Florida", "California", "Texas", "Virginia", "South Dakota", "North Dakota");

```
foreach ($states as $states) {
echo "I have visited " . $states. " during my stay in the United States.\n";
}

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
I have visited New York during my stay in the United States.
I have visited Los Angles during my stay in the United States.
I have visited Florida during my stay in the United States.
I have visited California during my stay in the United States.
I have visited Texas during my stay in the United States.
I have visited Virginia during my stay in the United States.
I have visited South Dakota during my stay in the United States.
I have visited North Dakota during my stay in the United States.
</body>
</html>
```

The Break Statement

If you are already into programming, you might have heard about the break statement. I have also used it in the switch statement in earlier chapters. A break statement is the best to jump out of PHP loops.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
    for ($a = 1; $a <= 25; $a++) {
    if ($a == 20) {
         break;
    }
echo "The requisite value of the variable a should be: " . $a . "\n";
echo "The do...while loop will go on.\n";
}

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
The requisite value of the variable a should be: 1
The do...while loop will go on.
The requisite value of the variable a should be: 2
The do...while loop will go on.
```

The requisite value of the variable a should be: 3
The do...while loop will go on.
The requisite value of the variable a should be: 4
The do...while loop will go on.
The requisite value of the variable a should be: 5
The do...while loop will go on.
The requisite value of the variable a should be: 6
The do...while loop will go on.
The requisite value of the variable a should be: 7
The do...while loop will go on.
The requisite value of the variable a should be: 8
The do...while loop will go on.
The requisite value of the variable a should be: 9
The do...while loop will go on.
The requisite value of the variable a should be: 10
The do...while loop will go on.
The requisite value of the variable a should be: 11
The do...while loop will go on.
The requisite value of the variable a should be: 12
The do...while loop will go on.
The requisite value of the variable a should be: 13
The do...while loop will go on.
The requisite value of the variable a should be: 14
The do...while loop will go on.
The requisite value of the variable a should be: 15
The do...while loop will go on.

The requisite value of the variable a should be: 16

The do...while loop will go on.

The requisite value of the variable a should be: 17

The do...while loop will go on.

The requisite value of the variable a should be: 18

The do...while loop will go on.

The requisite value of the variable a should be: 19

The do...while loop will go on.

</body>

</html>

I can make it stop at an earlier point. See the following example.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?ph p
    for ($a = 1; $a <= 25; $a++) {
    if ($a == 10) {
         break;
    }
echo "The requisite value of the variable a should be: " . $a . "\n";
echo "The do...while loop will go on.\n";
}

?>
</body>
```

</html>

$php main.php

<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
The requisite value of the variable a should be: 1
The do...while loop will go on.
The requisite value of the variable a should be: 2
The do...while loop will go on.
The requisite value of the variable a should be: 3
The do...while loop will go on.
The requisite value of the variable a should be: 4
The do...while loop will go on.
The requisite value of the variable a should be: 5
The do...while loop will go on.
The requisite value of the variable a should be: 6
The do...while loop will go on.
The requisite value of the variable a should be: 7
The do...while loop will go on.
The requisite value of the variable a should be: 8
The do...while loop will go on.
The requisite value of the variable a should be: 9
The do...while loop will go on.
</body>

```
</html>
```

There is one important point to note down. There is no restriction on the upper limit of the break point so if you enter a higher number than the upper limit, you may end up with a loop that runs its complete course.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
   for ($a = 1; $a <= 25; $a++) {
   if ($a == 30) {
        break;
   }
echo "The requisite value of the variable a should be: " . $a . "\n";
echo "The do...while loop will go on.\n";
}

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
```

The requisite value of the variable a should be: 1
The do...while loop will go on.
The requisite value of the variable a should be: 2
The do...while loop will go on.
The requisite value of the variable a should be: 3
The do...while loop will go on.
The requisite value of the variable a should be: 4
The do...while loop will go on.
The requisite value of the variable a should be: 5
The do...while loop will go on.
The requisite value of the variable a should be: 6
The do...while loop will go on.
The requisite value of the variable a should be: 7
The do...while loop will go on.
The requisite value of the variable a should be: 8
The do...while loop will go on.
The requisite value of the variable a should be: 9
The do...while loop will go on.
The requisite value of the variable a should be: 10
The do...while loop will go on.
The requisite value of the variable a should be: 11
The do...while loop will go on.
The requisite value of the variable a should be: 12
The do...while loop will go on.
The requisite value of the variable a should be: 13
The do...while loop will go on.

The requisite value of the variable a should be: 14
The do...while loop will go on.
The requisite value of the variable a should be: 15
The do...while loop will go on.
The requisite value of the variable a should be: 16
The do...while loop will go on.
The requisite value of the variable a should be: 17
The do...while loop will go on.
The requisite value of the variable a should be: 18
The do...while loop will go on.
The requisite value of the variable a should be: 19
The do...while loop will go on.
The requisite value of the variable a should be: 20
The do...while loop will go on.
The requisite value of the variable a should be: 21
The do...while loop will go on.
The requisite value of the variable a should be: 22
The do...while loop will go on.
The requisite value of the variable a should be: 23
The do...while loop will go on.
The requisite value of the variable a should be: 24
The do...while loop will go on.
The requisite value of the variable a should be: 25
The do...while loop will go on.
</body>

```
</html>
```

The Continue Statement

The continue statement works like the break statement in a way that it breaks iterations. If the condition is true, it breaks the loop but continues to execute the remaining code instead of jumping out of it. It is like pausing on a journey for a while and then getting on.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
    for ($a = 1; $a <= 15; $a++) {
    if ($a == 10) {
        continue;
    }
echo "The requisite value of the variable a should be: " . $a . "\n";
echo "The do...while loop will go on.\n";
}

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
```

</head>
<body>
The requisite value of the variable a should be: 1
The do...while loop will go on.
The requisite value of the variable a should be: 2
The do...while loop will go on.
The requisite value of the variable a should be: 3
The do...while loop will go on.
The requisite value of the variable a should be: 4
The do...while loop will go on.
The requisite value of the variable a should be: 5
The do...while loop will go on.
The requisite value of the variable a should be: 6
The do...while loop will go on.
The requisite value of the variable a should be: 7
The do...while loop will go on.
The requisite value of the variable a should be: 8
The do...while loop will go on.
The requisite value of the variable a should be: 9
The do...while loop will go on.
The requisite value of the variable a should be: 11
The do...while loop will go on.
The requisite value of the variable a should be: 12
The do...while loop will go on.
The requisite value of the variable a should be: 13
The do...while loop will go on.

The requisite value of the variable a should be: 14

The do...while loop will go on.

The requisite value of the variable a should be: 15

The do...while loop will go on.

</body>

</html>

You can see that the continue statement broke the code at number 10 and continued the execution with the rest of the block.

Break and Continue Statements in the while Loop

In the above code, you saw the usage of break and continue statements in the for loop. Now I will explain how you can integrate both statements in the while loop.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
$a = 0;

while($a < 20) {
  if ($a == 10) {
    break;
  }
echo "The requisite value of the variable a should be: " . $a . "\n";
echo "The do...while loop will go on.\n";
  $a++;
```

```
}


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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
The requisite value of the variable a should be: 0
The do...while loop will go on.
The requisite value of the variable a should be: 1
The do...while loop will go on.
The requisite value of the variable a should be: 2
The do...while loop will go on.
The requisite value of the variable a should be: 3
The do...while loop will go on.
The requisite value of the variable a should be: 4
The do...while loop will go on.
The requisite value of the variable a should be: 5
The do...while loop will go on.
The requisite value of the variable a should be: 6
The do...while loop will go on.
The requisite value of the variable a should be: 7
```

The do...while loop will go on.

The requisite value of the variable a should be: 8

The do...while loop will go on.

The requisite value of the variable a should be: 9

The do...while loop will go on.

</body>

</html>

Now I will add the continue statement to the same block of code as above. It will skip 10 and jump to 11 to continue the code.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
$a = 0;

while($a < 20) {
  if ($a == 10) {
    $a++;
    continue;
  }
echo "The requisite value of the variable a should be: " . $a . "\n";
echo "The do...while loop will go on.\n";
  $a++;
}
```

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
The requisite value of the variable a should be: 0
The do...while loop will go on.
The requisite value of the variable a should be: 1
The do...while loop will go on.
The requisite value of the variable a should be: 2
The do...while loop will go on.
The requisite value of the variable a should be: 3
The do...while loop will go on.
The requisite value of the variable a should be: 4
The do...while loop will go on.
The requisite value of the variable a should be: 5
The do...while loop will go on.
The requisite value of the variable a should be: 6
The do...while loop will go on.
The requisite value of the variable a should be: 7
The do...while loop will go on.
The requisite value of the variable a should be: 8
```

The do...while loop will go on.
The requisite value of the variable a should be: 9
The do...while loop will go on.
The requisite value of the variable a should be: 11
The do...while loop will go on.
The requisite value of the variable a should be: 12
The do...while loop will go on.
The requisite value of the variable a should be: 13
The do...while loop will go on.
The requisite value of the variable a should be: 14
The do...while loop will go on.
The requisite value of the variable a should be: 15
The do...while loop will go on.
The requisite value of the variable a should be: 16
The do...while loop will go on.
The requisite value of the variable a should be: 17
The do...while loop will go on.
The requisite value of the variable a should be: 18
The do...while loop will go on.
The requisite value of the variable a should be: 19
The do...while loop will go on.
</body>
</html>

Chapter Four: PHP Functions

One of the foundations of any programming languages includes a place to store data, which is the means to direct the program's flow. When you are writing a lengthy program, you need something to facilitate and ease off the process. That is where the role of functions comes in. A function is a bunch of statements that perform a specific job and then return a value. You may pull out a specific section of code that you have used multiple times and place it into a function. Then you can call the same function by its name when you need the code to execute.

Functions have a number of advantages over inline code. There will be less typing, reduced programming errors, a reduction in the loading time, a reduction in the execution time, and exceptional speed. You only have to compile it once and then you can call it by the name and the entire block of code it contains will execute.

This chapter will walk you through the process of defining and calling functions to pass arguments forward and backward. PHP functions come with many ready-made and built-in functions, which make PHP quite a rich language. If you want to use a function, you may call it by the name. The parenthesis inside a function tells PHP that you are referring to a function. The print function is the pseudo function, which is also known as construct. The only difference is that you might want to omit the parenthesis sometimes.

You can fill in a function with any number of arguments. You can even fill them in with a zero. The phpinfo() function is very handy for retrieving information about the present PHP installation. However, it is also equally useful for white hat hackers. A function call should not be left in a web-ready mode to avoid an attack. It is estimated that there are about 1000 built-in functions. On top of that, you can create your own custom functions as well. Here are some of the key features of PHP functions.

- A function is a pack of statements, which you can use multiple times in a program.
- A function does not load when a web page loads.
- A function only executes when you make a function call.

Defining a Function

A function definition starts with the keyword function. After the name, you must put in a letter or an underscore. After that, you can put in different letters, underscores, or numbers. You also have to fill the function with parenthesis. The names of functions are case sensitive which means print and PRINT are two separate functions. The function statements may contain multiple return statements, which force your function to cease its execution and return it to calling code. If you attack a value to the return statement, the calling code may retrieve it.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
function greetings() {
   echo "I wish you a good morning on this beautiful shiny day!";
}

greetings(); // I am calling the function

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
```

<body>

I wish you a good morning on this beautiful shiny day!</body>

</html>

PHP Function Arguments

You can pass on information to functions through arguments. An argument in a function is like a variable. You can specify arguments inside parenthesis, and add as many of them as you need. You should separate them by a comma. I will create a function with one argument. I will take it from there and pass information to the function through the same argument.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
function greetings($name) {
  echo "$name, I wish you a good morning on this beautiful shiny day!\n";
}

greetings('James'); // I am calling the function
greetings('Jasmine');
greetings('Martha');
greetings('Tim');
greetings('David');
greetings('Chelsea');

?>
</body>
```

</html>

$php main.php

<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
James, I wish you a good morning on this beautiful shiny day!
Jasmine, I wish you a good morning on this beautiful shiny day!
Martha, I wish you a good morning on this beautiful shiny day!
Tim, I wish you a good morning on this beautiful shiny day!
David, I wish you a good morning on this beautiful shiny day!
Chelsea, I wish you a good morning on this beautiful shiny day!
</body>
</html>

Multiple Arguments

In the following example, I will give an example of multiple arguments.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
function greetings($name, $day) {
  echo "$name, I wish you a good morning on this beautiful shiny day!\n";
```

```
  echo "Anyway, we shall start working on $day.\n";
}

greetings('James', 'Monday'); // I am calling the function
greetings('Jasmine', 'Tuesday');
greetings('Martha', 'Wednesday');
greetings('Tim', 'Thursday');
greetings('David', 'Friday');
greetings('Chelsea', 'Saturday');

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
James, I wish you a good morning on this beautiful shiny day!
Anyway, we shall start working on Monday.
Jasmine, I wish you a good morning on this beautiful shiny day!
Anyway, we shall start working on Tuesday.
Martha, I wish you a good morning on this beautiful shiny day!
Anyway, we shall start working on Wednesday.
Tim, I wish you a good morning on this beautiful shiny day!
Anyway, we shall start working on Thursday.
```

David, I wish you a good morning on this beautiful shiny day!

Anyway, we shall start working on Friday.

Chelsea, I wish you a good morning on this beautiful shiny day!

Anyway, we shall start working on Saturday.

</body>

</html>

As PHP is a loosely typed language, we do not have to tell PHP which type of variable it is working on. You can see that it dealt with integers in the same way as it did with strings. There is no need to convert an integer into a string, as we have to do in Python language.

Python associates a data type automatically to a variable based on the value of the variable. Since the data types are not set up in the strictest senses, you can do many things like the addition of strings to integers without triggering an error.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
function addingtheNumbers(int $x, int $y) {
  return $x + $y;
}
echo addingtheNumbers(10, "10 days");
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
20</body>
</html>
```

PHP Function Default Values

Just like other programming languages, you can set up default argument values for PHP. I will explain in the following example how to set up the default parameter. If we make a function call, without arguments, PHP will pick up the default arguments and run the code.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
function greetings($name = 'Mark', $day = 'Friday') {
  echo "$name, I wish you a good morning on this beautiful shiny day!\n";
  echo "Anyway, we shall start working on $day.\n";
}

greetings('James', 'Monday'); // I am calling the function
greetings('Jasmine', 'Tuesday');
greetings('Martha', 'Wednesday');
greetings();
```

```
greetings('Tim', 'Thursday');
greetings('David', 'Friday');
greetings();
greetings('Chelsea', 'Saturday');

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
James, I wish you a good morning on this beautiful shiny day!
Anyway, we shall start working on Monday.
Jasmine, I wish you a good morning on this beautiful shiny day!
Anyway, we shall start working on Tuesday.
Martha, I wish you a good morning on this beautiful shiny day!
Anyway, we shall start working on Wednesday.
Mark, I wish you a good morning on this beautiful shiny day!
Anyway, we shall start working on Friday.
Tim, I wish you a good morning on this beautiful shiny day!
Anyway, we shall start working on Thursday.
David, I wish you a good morning on this beautiful shiny day!
Anyway, we shall start working on Friday.
Mark, I wish you a good morning on this beautiful shiny day!
```

Anyway, we shall start working on Friday.

Chelsea, I wish you a good morning on this beautiful shiny day!

Anyway, we shall start working on Saturday.

</body>

</html>

Returning Values

You can let PHP functions to return values by using the return statement.

```
<?php declare(strict_types=1); // strict requirement ?>
<!DOCTYPE html>
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
function summing(int $a, int $b) {
  $c = $a + $b;
  return $c;
}

echo "15 + 10 = " . summing(5, 10) . "\n";
echo "7 + 33 = " . summing(7, 13) . "\n";
echo "2 + 9 = " . summing(2, 4) . "\n";
echo "2 + 10 = " . summing(2, 4) . "\n";
echo "29 + 14 = " . summing(2, 4) . "\n";
echo "82 + 94 = " . summing(2, 4) . "\n";
```

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

$php main.php

```
<!DOCTYPE html>
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
15 + 10 = 15
7 + 33 = 20
2 + 9 = 6
2 + 10 = 6
29 + 14 = 6
82 + 94 = 6
</body>
</html>
```

PHP supports Type Declarations for return statement. As with the type declaration for different function arguments, it will display a fatal error when there is mismatch in the data type. When you want to declare a type for the return of the function, you have to add a colon (:) and the data type before the curly braces.

```
<?php declare(strict_types=1); // strict requirement ?>
<!DOCTYPE html>
<html>
<head>
```

```
<title>This is my PHP Script</title>
</head>
<body>
<?php
function summing(float $a, float $b) : float {
  return $a + $b;
}

echo summing(152, 105) . "\n";
echo summing(77, 331) . "\n";
echo summing(22, 95) . "\n";
echo summing(25, 49) . "\n";
echo summing(2, 4) . "\n";
echo summing(26, 42) . "\n";

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

$php main.php

```
<!DOCTYPE html>
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
257
40.8
```

117

74

6

68

</body>

</html>

Chapter Five: PHP Arrays

Arrays are labelled as a collection of values that are related. They can be data submitted in a specific form like the names of students in a class or the list of the populations of cities. This chapter will walk you through how to work up with arrays. You will learn how to loop through arrays with for and foreach loops. Modifying arrays may introduce the explode() and implode() functions which turn different arrays into strings and the strings into arrays. You can sort out arrays as well.

An array has elements in the form of keys and values. An array that holds information about vegetables and fruits will have the names of fruits and vegetables as keys the type as values. You can pack up an array with a single element if you have the given key.

```
<html>

<head>

<title>This is my PHP Script</title>

</head>

<body>

<?php

$vegetables = array("pumpkin", "tomato", "potato", "cabbage", "cauliflower", "ginger", "turmeric", "peas");

echo "Today I plan to cook " . $vegetables[0] . "\n";

echo "Today I plan to cook " . $vegetables[1] . "\n";

echo "Today I plan to cook " . $vegetables[2] . "\n";

echo "Today I plan to cook " . $vegetables[3] . "\n";

echo "Today I plan to cook " . $vegetables[4] . "\n";

echo "Today I plan to cook " . $vegetables[5] . "\n";

echo "Today I plan to cook " . $vegetables[6] . "\n";

echo "Today I plan to cook " . $vegetables[7] . "\n";

?>
```

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
Today I plan to cook pumpkin
Today I plan to cook tomato
Today I plan to cook potato
Today I plan to cook cabbage
Today I plan to cook cauliflower
Today I plan to cook ginger
Today I plan to cook turmeric
Today I plan to cook peas
</body>
</html>
```

An array is a special variable that can hold more than one value at a given time period.

Length of an Array

PHP allows you to count the length of an array with the help of the count() function.

```
<html>
<head>
<title>This is my PHP Script</title>
```

```
</head>
<body>
<?php
$vegetables = array("pumpkin", "tomato", "potato", "cabbage", "cauliflower", "ginger", "turmeric", "peas");
echo count($vegetables);
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
8</body>
</html>
```

PHP Indexed Arrays

You can create indexed arrays in two ways. You can assign the index automatically by staring it from 0. You have seen one way in the above example. I will explain the other way in the following example.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
```

```
<?php
$vegetables[0] = "pumpkin";
 $vegetables[1] = "tomato";
$vegetables[2] = "potato";
$vegetables[3] = "cabbage";
 $vegetables[4] = "cauliflower";
$vegetables[5] = "ginger";
$vegetables[6] = "turmeric";
 $vegetables[7] = "peas";
echo "Today I plan to cook " . $vegetables[0] . "\n";
echo "Today I plan to cook " . $vegetables[1] . "\n";
echo "Today I plan to cook " . $vegetables[2] . "\n";
echo "Today I plan to cook " . $vegetables[3] . "\n";
echo "Today I plan to cook " . $vegetables[4] . "\n";
echo "Today I plan to cook " . $vegetables[5] . "\n";
echo "Today I plan to cook " . $vegetables[6] . "\n";
echo "Today I plan to cook " . $vegetables[7] . "\n";
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
```

Today I plan to cook pumpkin

Today I plan to cook tomato

Today I plan to cook potato

Today I plan to cook cabbage

Today I plan to cook cauliflower

Today I plan to cook ginger

Today I plan to cook turmeric

Today I plan to cook peas

</body>

</html>

Looping through an Indexed Array

You can loop through an indexed array and print the values one by one. I will pair up the indexed array with the for loop to produce the results I need.

```
<html>

<head>

<title>This is my PHP Script</title>

</head>

<body>

<?php

$vegetables = array("pumpkin", "tomato", "potato", "cabbage", "cauliflower", "ginger", "turmeric", "peas");

$arrlength = count($vegetables);


for($a = 0; $a < $arrlength; $a++) {

  echo $vegetables[$a];
```

```
echo "\nToday I plan to cook " . $vegetables[0] . "\n";
echo "Today I plan to cook " . $vegetables[1] . "\n";
echo "Today I plan to cook " . $vegetables[2] . "\n";
echo "Today I plan to cook " . $vegetables[3] . "\n";
echo "Today I plan to cook " . $vegetables[4] . "\n";
echo "Today I plan to cook " . $vegetables[5] . "\n";
echo "Today I plan to cook " . $vegetables[6] . "\n";
echo "Today I plan to cook " . $vegetables[7] . "\n";
}
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
pumpkin
Today I plan to cook pumpkin
Today I plan to cook tomato
Today I plan to cook potato
Today I plan to cook cabbage
Today I plan to cook cauliflower
Today I plan to cook ginger
Today I plan to cook turmeric
```

Today I plan to cook peas
tomato
Today I plan to cook pumpkin
Today I plan to cook tomato
Today I plan to cook potato
Today I plan to cook cabbage
Today I plan to cook cauliflower
Today I plan to cook ginger
Today I plan to cook turmeric
Today I plan to cook peas
potato
Today I plan to cook pumpkin
Today I plan to cook tomato
Today I plan to cook potato
Today I plan to cook cabbage
Today I plan to cook cauliflower
Today I plan to cook ginger
Today I plan to cook turmeric
Today I plan to cook peas
cabbage
Today I plan to cook pumpkin
Today I plan to cook tomato
Today I plan to cook potato
Today I plan to cook cabbage
Today I plan to cook cauliflower
Today I plan to cook ginger

Today I plan to cook turmeric
Today I plan to cook peas
cauliflower
Today I plan to cook pumpkin
Today I plan to cook tomato
Today I plan to cook potato
Today I plan to cook cabbage
Today I plan to cook cauliflower
Today I plan to cook ginger
Today I plan to cook turmeric
Today I plan to cook peas
ginger
Today I plan to cook pumpkin
Today I plan to cook tomato
Today I plan to cook potato
Today I plan to cook cabbage
Today I plan to cook cauliflower
Today I plan to cook ginger
Today I plan to cook turmeric
Today I plan to cook peas
turmeric
Today I plan to cook pumpkin
Today I plan to cook tomato
Today I plan to cook potato
Today I plan to cook cabbage
Today I plan to cook cauliflower

Today I plan to cook ginger

Today I plan to cook turmeric

Today I plan to cook peas

peas

Today I plan to cook pumpkin

Today I plan to cook tomato

Today I plan to cook potato

Today I plan to cook cabbage

Today I plan to cook cauliflower

Today I plan to cook ginger

Today I plan to cook turmeric

Today I plan to cook peas

</body>

</html>

PHP Associative Arrays

Associative arrays use named keys. You can assign any name to them. You may create an associative array in two ways.

<html>

<head>

<title>This is my PHP Script</title>

</head>

<body>

<?php

$vegetablesfruits = array ("pumpkin"=> 'vegetable', "tomato" => 'fruit', "potato" => 'vegetable', "cabbage" => 'vegetable', "cauliflower" => 'vegetable', "orange" => 'fruit', "ginger" => 'vegetable', "turmeric" => 'spice', "peas" => 'vegetable');

```
echo "\nToday I plan to cook a " . $vegetablesfruits['pumpkin'] . "\n";
echo "Today I plan to cook a " . $vegetablesfruits['tomato'] . "\n";
echo "Today I plan to cook a " . $vegetablesfruits['potato'] . "\n";
echo "Today I plan to cook a " . $vegetablesfruits['cabbage'] . "\n";
echo "Today I plan to cook a " . $vegetablesfruits['cauliflower'] . "\n";
echo "Today I plan to cook a " . $vegetablesfruits['orange'] . "\n";
echo "Today I plan to cook a " . $vegetablesfruits['ginger'] . "\n";
echo "Today I plan to cook a " . $vegetablesfruits['turmeric'] . "\n";
echo "Today I plan to cook a " . $vegetablesfruits['peas'] . "\n";

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>

Today I plan to cook a vegetable
Today I plan to cook a fruit
Today I plan to cook a vegetable
Today I plan to cook a vegetable
Today I plan to cook a vegetable
Today I plan to cook a fruit
Today I plan to cook a vegetable
```

Today I plan to cook a spice

Today I plan to cook a vegetable

</body>

</html>

Looping Through the Associative Array

In the following example, I will loop through the associative array that I have created above. I will use a foreach loop to do that.

```
<!DOCTYPE html>

<html>

<head>

<title>This is my PHP Script</title>

</head>

<body>

<?php

$vegetablesfruits = array ("pumpkin"=> 'vegetable', "tomato" => 'fruit', "potato" => 'vegetable', "cabbage" => 'vegetable', "cauliflower" => 'vegetable', "orange" => 'fruit', "ginger" => 'vegetable', "turmeric" => 'spice', "peas" => 'vegetable');

foreach($vegetablesfruits as $x => $x_value) {


echo "\nToday I plan to cook " . $x . ", which technically is a ". $x_value . ".\n";

}

?>

</body>

</html>
```

$php main.php

```
<!DOCTYPE html>
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>

Today I plan to cook pumpkin, which technically is a vegetable.

Today I plan to cook tomato, which technically is a fruit.

Today I plan to cook potato, which technically is a vegetable.

Today I plan to cook cabbage, which technically is a vegetable.

Today I plan to cook cauliflower, which technically is a vegetable.

Today I plan to cook orange, which technically is a fruit.

Today I plan to cook ginger, which technically is a vegetable.

Today I plan to cook turmeric, which technically is a spice.

Today I plan to cook peas, which technically is a vegetable.
</body>
</html>
```

Multidimensional Arrays

A multidimensional array is an array that contains more than one arrays. PHP supports the arrays that are two or four or six levels deep. However, the arrays that are more than three levels in depth are hard to get over. You may need two indices to select a particular element when you are working

with two-dimensional arrays. You may need three-dimensional arrays when you are working with three-dimensional array.

I will create an array with three columns; the first indicates the name of the cars, the second indicates the position of stock, and the third indicates the number of cars that have been sold.

```
<!DOCTYPE html>
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
$middleclasscars = array (
  array("Swift",40,22),
  array("Vitz",15,15),
  array("Passo",50,25),
  array("Cultus",25,15),
  array("City EXI",30,45),
  array("Civic",45,15)

);

echo $middleclasscars[0][0].": The cars that are in stock:
".$middleclasscars[0][1].", The cars that have been sold:
".$middleclasscars[0][2].".\n";
echo $middleclasscars[1][0].": The cars that are in stock:
".$middleclasscars[1][1].", The cars that are sold: ".$middleclasscars[1]
[2].".\n";
```

```
echo $middleclasscars[2][0].": The cars that are in stock:
".$middleclasscars[2][1].", The cars that are sold: ".$middleclasscars[2]
[2].".\n";

echo $middleclasscars[3][0].": The cars that are in stock:
".$middleclasscars[3][1].", The cars that are sold: ".$middleclasscars[3]
[2].".\n";

echo $middleclasscars[4][0].": The cars that are in stock:
".$middleclasscars[4][1].", The cars that are sold: ".$middleclasscars[4]
[2].".\n";

echo $middleclasscars[5][0].": The cars that are in stock:
".$middleclasscars[5][1].", The cars that are sold: ".$middleclasscars[5]
[2].".\n";

?>

</body>

</html>
```

$php main.php

```
<!DOCTYPE html>

<html>

<head>

<title>This is my PHP Script</title>

</head>

<body>

Swift: The cars that are in stock: 40, The cars that have been sold: 22

Vitz: The cars that are in stock: 15, The cars that are sold: 15

Passo: The cars that are in stock: 50, The cars that are sold: 25

Cultus: The cars that are in stock: 25, The cars that are sold: 15

City EXI: The cars that are in stock: 30, The cars that are sold: 45

Civic: The cars that are in stock: 45, The cars that are sold: 15

</body>
```

</html>

Now I will add to it a for loop inside another for loop.

```
<!DOCTYPE html>
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
$middleclasscars = array (
  array("Swift",40,22),
  array("Vitz",15,15) ,
  array("Passo",50,25),
  array("Cultus",25,15),
  array("City EXI",30,45),
  array("Civic",45,15)

);

echo $middleclasscars[0][0].": The cars that are in stock:
".$middleclasscars[0][1].", The cars that have been sold:
".$middleclasscars[0][2].".\n";

echo $middleclasscars[1][0].": The cars that are in stock:
".$middleclasscars[1][1].", The cars that are sold: ".$middleclasscars[1]
[2].".\n";

echo $middleclasscars[2][0].": The cars that are in stock:
".$middleclasscars[2][1].", The cars that are sold: ".$middleclasscars[2]
[2].".\n";
```

```
echo $middleclasscars[3][0].": The cars that are in stock:
".$middleclasscars[3][1].", The cars that are sold: ".$middleclasscars[3]
[2].".\n";

echo $middleclasscars[4][0].": The cars that are in stock:
".$middleclasscars[4][1].", The cars that are sold: ".$middleclasscars[4]
[2].".\n";

echo $middleclasscars[5][0].": The cars that are in stock:
".$middleclasscars[5][1].", The cars that are sold: ".$middleclasscars[5]
[2].".\n";

for ($rows = 0; $rows < 6; $rows++) {
  echo "<p><b>This is Row number $rows</b></p>\n";
  echo "<ul>\n";
  for ($cols = 0; $cols < 3; $cols++) {
    echo "<li>".$middleclasscars[$rows][$cols]."</li>\n" ;
  }
  echo "</ul>\n";
}
?>
</body>
</html>
```

$php main.php

```
<!DOCTYPE html>
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
```

Swift: The cars that are in stock: 40, The cars that have been sold: 22

Vitz: The cars that are in stock: 15, The cars that are sold: 15

Passo: The cars that are in stock: 50, The cars that are sold: 25

Cultus: The cars that are in stock: 25, The cars that are sold: 15

City EXI: The cars that are in stock: 30, The cars that are sold: 45

Civic: The cars that are in stock: 45, The cars that are sold: 15

This is Row number 0

Swift

40

22

This is Row number 1

Vitz

15

15

This is Row number 2

Passo

50

25

This is Row number 3


```
<li>Cultus</li>
<li>25</li>
<li>15</li>
</ul>
<p><b>This is Row number 4</b></p>
<ul>
<li>City EXI</li>
<li>30</li>
<li>45</li>
</ul>
<p><b>This is Row number 5</b></p>
<ul>
<li>Civic</li>
<li>45</li>
<li>15</li>
</ul>
</body>
</html>
```

Sorting Arrays

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
```

```
$vegetables = array("pumpkin", "tomato", "potato", "cabbage", "cauliflower", "ginger", "turmeric", "peas");
sort($vegetables);
$length = count($vegetables);
for($x = 0; $x < $length; $x++) {
  echo "I am buying " . $vegetables[$x] . " from the vegetable market.";
  echo "\n";
}
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
I am buying cabbage from the vegetable market.
I am buying cauliflower from the vegetable market.
I am buying ginger from the vegetable market.
I am buying peas from the vegetable market.
I am buying potato from the vegetable market.
I am buying pumpkin from the vegetable market.
I am buying tomato from the vegetable market.
I am buying turmeric from the vegetable market.
</body>
```

```
</html>
```

The following example will sort the array in the descending order.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
$vegetables = array("pumpkin", "tomato", "potato", "cabbage", "cauliflower", "ginger", "turmeric", "peas");
rsort($vegetables);
$length = count($vegetables);
for($x = 0; $x < $length; $x++) {
  echo "I am buying " . $vegetables[$x] . " from the vegetable market.";
  echo "\n";
}
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
I am buying turmeric from the vegetable market.
```

I am buying tomato from the vegetable market.

I am buying pumpkin from the vegetable market.

I am buying potato from the vegetable market.

I am buying peas from the vegetable market.

I am buying ginger from the vegetable market.

I am buying cauliflower from the vegetable market.

I am buying cabbage from the vegetable market.

</body>

</html>

All I did was to add ‘r’ to the sort keyword.

The following example to reverse the array to ascending order. I have added ‘a’ to the sort keyword.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
<?php
$vegetables = array("pumpkin", "tomato", "potato", "cabbage", "cauliflower", "ginger", "turmeric", "peas");
asort($vegetables);
$length = count($vegetables);
for($x = 0; $x < $length; $x++) {
  echo "I am buying " . $vegetables[$x] . " from the vegetable market.";
  echo "\n";
}
```

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

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
I am buying pumpkin from the vegetable market.
I am buying tomato from the vegetable market.
I am buying potato from the vegetable market.
I am buying cabbage from the vegetable market.
I am buying cauliflower from the vegetable market.
I am buying ginger from the vegetable market.
I am buying turmeric from the vegetable market.
I am buying peas from the vegetable market.
</body>
</html>
```

The following example will sort out the arrays according to the keys of the array.

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
```

```
<?php
  $vegetablesfruits = array ("pumpkin"=> 'vegetable', "tomato" => 'fruit', "potato" => 'vegetable', "cabbage" => 'vegetable', "cauliflower" => 'vegetable', "orange" => 'fruit', "ginger" => 'vegetable', "turmeric" => 'spice', "peas" => 'vegetable');
ksort($vegetablesfruits);

foreach($vegetablesfruits as $x => $x_value) {
  echo "\nToday I plan to cook " . $x . ", which technically is a ". $x_value . ".\n";
  echo "\n";
}
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>
```

Today I plan to cook cabbage, which technically is a vegetable.

Today I plan to cook cauliflower, which technically is a vegetable.

Today I plan to cook ginger, which technically is a vegetable.

Today I plan to cook orange, which technically is a fruit.

Today I plan to cook peas, which technically is a vegetable.

Today I plan to cook potato, which technically is a vegetable.

Today I plan to cook pumpkin, which technically is a vegetable.

Today I plan to cook tomato, which technically is a fruit.

Today I plan to cook turmeric, which technically is a spice.

</body>

</html>

Now I will sort as per the descending order of the values. The keyword will be arsort() for the purpose.

```
<html>

<head>

<title>This is my PHP Script</title>

</head>

<body>

<?php

  $vegetablesfruits = array ("pumpkin"=> 'vegetable', "tomato" => 'fruit', "potato" => 'vegetable', "cabbage" => 'vegetable', "cauliflower" => 'vegetable', "orange" => 'fruit', "ginger" => 'vegetable', "turmeric" => 'spice', "peas" => 'vegetable');

arsort($vegetablesfruits);

foreach($vegetablesfruits as $x => $x_value) {

  echo "\nToday I plan to cook " . $x . ", which technically is a ". $x_value . ".\n";

  echo "\n";

}

?>

</body>

</html>
```

$php main.php

<html>

<head>

<title>This is my PHP Script</title>

</head>

<body>

Today I plan to cook pumpkin, which technically is a vegetable.

Today I plan to cook potato, which technically is a vegetable.

Today I plan to cook cabbage, which technically is a vegetable.

Today I plan to cook cauliflower, which technically is a vegetable.

Today I plan to cook ginger, which technically is a vegetable.

Today I plan to cook peas, which technically is a vegetable.

Today I plan to cook turmeric, which technically is a spice.

Today I plan to cook tomato, which technically is a fruit.

Today I plan to cook orange, which technically is a fruit.

</body>

</html>

This will be the descending order as per the keys of the array. I will use the krsort keyword for the purpose.

<html>

<head>

<title>This is my PHP Script</title>

</head>

<body>

<?php

```
  $vegetablesfruits = array ("pumpkin"=> 'vegetable', "tomato" => 'fruit', "potato" => 'vegetable', "cabbage" => 'vegetable', "cauliflower" => 'vegetable', "orange" => 'fruit', "ginger" => 'vegetable', "turmeric" => 'spice', "peas" => 'vegetable');
krsort($vegetablesfruits);

foreach($vegetablesfruits as $x => $x_value) {
  echo "\nToday I plan to cook " . $x . ", which technically is a ". $x_value . ".\n";
  echo "\n";
}
?>
</body>
</html>
```

$php main.php

```
<html>
<head>
<title>This is my PHP Script</title>
</head>
<body>

Today I plan to cook turmeric, which technically is a spice.
Today I plan to cook tomato, which technically is a fruit.
Today I plan to cook pumpkin, which technically is a vegetable.
Today I plan to cook potato, which technically is a vegetable.
Today I plan to cook peas, which technically is a vegetable.
Today I plan to cook orange, which technically is a fruit.
Today I plan to cook ginger, which technically is a vegetable.
```

Today I plan to cook cauliflower, which technically is a vegetable.
Today I plan to cook cabbage, which technically is a vegetable.

</body>
</html>

Chapter Six: PHP Superglobals

Super global variables are basically built-in variables that are mostly available in different scopes. I will discuss each of them in the following code samples, and see what they can do.

PHP $GLOBALS

$GLOBALS is a super variable in PHP, which you can use to access different global variables from any point in the PHP script such as methods and functions. PHP tends to store variables in the array called $GLOBALS[index]. The index has the name of the variable.

```
<!DOCTYPE html>
<html>
<body>

<?php
$a = 55;
$b = 15;

function additionofnumbers() {
  $GLOBALS['c'] = $GLOBALS['a'] + $GLOBALS['b'];
}

additionofnumbers();
echo $c;
?>

</body>
</html>
```

$php main.php

```
<!DOCTYPE html>
```

```
<html>
<body>

70
</body>
</html>
```

PHP Superglobal = $_SERVER

Super global variables are kind of built-in variables that are available in different scopes. $_SERVER is packed up with the information about paths, headers, and script locations.

```
<html>
<body>

<form method="post your information" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="firstname">
  <input type="Please submit here">
</form>

<?php
if ($_SERVER["REQUEST"] == "POST") {
  $uname = $_REQUEST['firstname'];
  if (empty($uname)) {
    echo "Name is currently empty. Please fill in the requisite information.";
  } else {
    echo $uname ;
  }
}
```

```
?>

</body>
</html>
```

$php main.php

```
<html>
<body>

<form method="post your information" action="main.php">
  Name: <input type="text" name="firstname">
  <input type="Please submit here">
</form>


</body>
</html>
```

PHP $_POST Variable

PHP $_POST is a super global variable you can use to gather data after submitting the HTML form. You can use it to pass different variables. The following example shows a form that has an input coupled with a submit button. When a user submits data by clicking on the button, the form data processes to file that you specify in your HTML form's action attribute.

```
<html>
<body>

<form method="post your information" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="firstname">
  <input type="Please submit here">
```

```
</form>

<?php
if ($_SERVER["REQUEST"] == "POST") {
  $uname = $_POST['firstname'];
  if (empty($uname)) {
    echo "Name is currently empty. Please fill in the requisite information.";
  } else {
    echo $uname;
  }
}
?>

</body>
</html>
```

$php main.php

```
<html>
<body>

<form method="post your information" action="main.php">
  Name: <input type="text" name="firstname">
  <input type="Please submit here">
</form>

</body>
</html>
```

In this section, I will discuss the miscellaneous features one by one so that you can easily grasp the concepts.

PHP Date & Time

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "The date as of today is " . date("Y/m/d") . "\n";
echo "The date as of today is " . date("Y.m.d") . "\n";
echo "The date as of today is " . date("Y-m-d") . "\n";
echo "The date as of today is " . date("l");
?>

</body>
</html>
```

$php main.php

```
<!DOCTYPE html>
<html>
<body>

The date as of today is 2020/12/31
The date as of today is 2020.1231
The date as of today is 2020-12-31
The date as of today is Thursday
</body>
</html>
```

By using this code, you can add date to your website in different styles and formats. This will give your website style. You also can add copyright symbol to your website by a simple code.

```
<!DOCTYPE html>
<html>
<body>


© 2004-<?php echo date("Y");?>

</body>
</html>
```

$php main.php

```
<!DOCTYPE html>
<html>
<body>


© 2004-2020
</body>
</html>
```

You can write and add the following code to your website to display the time to your users.

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My dear visitor, the time at the moment is " . date("h:i:sa");
```

?>

</body>
</html>

$php main.php

<!DOCTYPE html>
<html>
<body>

My dear visitor, the time at the moment is 10:13:16am
</body>
</html>

Chapter Seven: Object Oriented Programming

The normal procedural programming revolves around simple writing procedures or functions that perform simple data operations. As compared to that, object-oriented programming revolves around the objects that contain data and functions. There are several advantages to object-oriented programming that you must consider when moving on to advanced PHP programming.

- Object Oriented Programming speeds up coding and execution.
- Object Oriented Programming gives off proper structure to programs.
- Object Oriented Programming aids in keeping your code clean. It also makes the maintenance of the code easy.
- Object Oriented Programming makes debugging of the code easier.
- Object Oriented Programming helps you create applications that you can reuse. Therefore, it cuts down the amount of code and the development time.

PHP Classes

I will create a class named as guns. It will have different properties like name, capacity, color, and weight. I will define different variables like $gname, $gcapacity, $gcolor, and $gweight to hold specific values for the properties. When I will create individual objects, all of them will inherit the properties and behaviors of the classes.

In the first step of making a class, you have to define it. You can use the class keyword to define the class, which may be followed by the name and curly braces. Here is the gun class defined.

```
<!DOCTYPE html>

<html>

<body>


<?php

class Guns {
```

```
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity;
  public $gweight;

  // Class Methods
  function setting_the_name($gname) {
    $this->gname = $gname;
  }
  function getting_the_name() {
    return $this->gname;
  }

function setting_the_color($gcolor) {
    $this->gcolor = $gcolor;
  }
  function getting_the_color() {
    return $this->gcolor;
  }

function setting_the_capacity($gcapacity) {
    $this->gcapacity = $gcapacity;
  }
  function getting_the_capacity() {
    return $this->gcapacity;
  }
```

```
function setting_the_weight($gweight) {
    $this->gweight = $gweight;
  }
  function getting_the_weight() {
    return $this->gweight;
  }
}
?>

</body>
</html>
```

Class Objects

Now that I have defined the class, I can move on to build objects on its foundation. I can create one or more objects as per the demands of the program. each object has got all the method and properties properly defined in the class, however it will have varying values for properties. You can use the keyword ‘new’ for creating objects out of a class. Take a look at the following instances.

```
<!DOCTYPE html>
<html>
<body>

<?php
class Guns {
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity ;
```

```
  public $gweight;

  // Class Methods
  function setting_the_name($gname) {
    $this->gname = $gname;
  }
  function getting_the_name() {
    return $this->gname;
  }

function setting_the_color($gcolor) {
    $this->gcolor = $gcolor;
  }
  function getting_the_color() {
    return $this->gcolor;
  }

function setting_the_capacity($gcapacity) {
    $this->gcapacity = $gcapacity;
  }
  function getting_the_capacity() {
    return $this->gcapacity;
  }

function setting_the_weight($gweight) {
    $this->gweight = $gweight;
  }
  function getting_the_weight() {
```

```
    return $this->gweight;
  }
}

$bazooka = new Guns();
$revolver = new Guns();
$bazooka->setting_the_name('This is a hard-hitting bazooka ready for waging war.');
$revolver->setting_the_name('This is a revolver to kill the guards at the gates silently.');

echo $bazooka->getting_the_name();
echo "\n";
echo $revolver->getting_the_name();
?>

</body>
</html>
```

$php main.php

```
<!DOCTYPE html>
<html>
<body>

This is a hard-hitting bazooka ready for waging war.
This is a revolver to kill the guards at the gates silently.
</body>
</html>
```

Adding More Methods to the Guns Class

In the following example, I will expand the class by adding more methods.

```
<!DOCTYPE html>
<html>
<body>

<?php
class Guns {
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity;
  public $gweight;

  // Class Methods
  function setting_the_name($gname) {
    $this->gname = $gname;
  }
  function getting_the_name() {
    return $this->gname;
  }

function setting_the_color($gcolor) {
    $this->gcolor = $gcolor;
  }
  function getting_the_color() {
    return $this->gcolor ;
  }
```

```
function setting_the_capacity($gcapacity) {
    $this->gcapacity = $gcapacity;
  }
  function getting_the_capacity() {
    return $this->gcapacity;
  }

function setting_the_weight($gweight) {
    $this->gweight = $gweight;
  }
  function getting_the_weight() {
    return $this->gweight;
  }
}

$bazooka = new Guns();
$revolver = new Guns();

$bazooka->setting_the_name('This is a hard-hitting bazooka ready for waging war.');
$bazooka->setting_the_color('The color of my bazooka is black.');
$bazooka->setting_the_capacity('I can load up to six rockets to my bazooka. This is a beast for the enemy.');


echo "Name of the bazooka : " . $bazooka->getting_the_name();
echo "\n";
echo "Color of the bazooka : " . $bazooka->getting_the_color();
echo "\n";
```

```
echo "Capacity of the bazooka : " . $bazooka->getting_the_capacity();
echo "\n";


?>

</body>
</html>
```

$php main.php

```
<!DOCTYPE html>
<html>
<body>

Name of the bazooka : This is a hard-hitting bazooka ready for waging war.
Color of the bazooka : The color of my bazooka is black.
Capacity of the bazooka : I can load up to six rockets to my bazooka. This is a beast for the enemy.

</body>
</html>
```

Adding Multiple Instances

In the following code snippet, I will add more than two instances to the same class.

```
<!DOCTYPE html>
<html>
<body>

<?php
```

```
class Guns {
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity;
  public $gweight;

  // Class Methods
  function setting_the_name($gname) {
    $this->gname = $gname;
  }
  function getting_the_name() {
    return $this->gname;
  }

function setting_the_color($gcolor) {
    $this->gcolor = $gcolor;
  }
  function getting_the_color() {
    return $this->gcolor;
  }

function setting_the_capacity($gcapacity) {
    $this->gcapacity = $gcapacity ;
  }
  function getting_the_capacity() {
    return $this->gcapacity;
  }
```

```
function setting_the_weight($gweight) {
    $this->gweight = $gweight;
  }
  function getting_the_weight() {
    return $this->gweight;
  }
}

$bazooka = new Guns();
$revolver = new Guns();
$Ak47 = new Guns();
$shotgun = new Guns();

$bazooka->setting_the_name('This is a hard-hitting bazooka ready for waging war.');
$bazooka->setting_the_color('The color of my bazooka is black.');
$bazooka->setting_the_capacity('I can load up to six rockets to my bazooka. This is a beast for the enemy.');

$revolver->setting_the_name('This is a revolver to kill the guards at the gates silently.');
$revolver->setting_the_color('The color of the revolver is silver grey.');
$revolver->setting_the_capacity('I can load up to ten bullets to my revolver. This is a beast for the enemy.');

$Ak47->setting_the_name('This is an Ak47 to kill the guards at the gates silently.');
$Ak47->setting_the_color('The color of the gun is brown.');
$Ak47->setting_the_capacity('I can load up to twenty bullets to my gun. This is a beast for the enemy.');
```

```
$shotgun->setting_the_name('This is a shotgun to kill the enemy with a bang.');
$shotgun->setting_the_color('The color of the gun is brown.');
$shotgun->setting_the_capacity('I can load up to two bullets to my gun. This is a beast for the enemy.');

echo "Name of the bazooka : " . $bazooka->getting_the_name();
echo "\n";
echo "Color of the bazooka : " . $bazooka->getting_the_color();
echo "\n";
echo "Capacity of the bazooka : " . $bazooka->getting_the_capacity();
echo "\n";

echo "Name of the revolver : " . $revolver->getting_the_name();
echo "\n";
echo "Color of the revolver : " . $revolver->getting_the_color();
echo "\n";
echo "Capacity of the revolver : " . $revolver->getting_the_capacity();
echo "\n";

echo "Name of the rifle : " . $Ak47->getting_the_name();
echo "\n";
echo "Color of the rifle : " . $Ak47->getting_the_color();
echo "\n";
echo "Capacity of the rifle : " . $Ak47->getting_the_capacity();
echo "\n";

echo "Name of the gun : " . $shotgun->getting_the_name();
echo "\n";
```

```
echo "Color of the gun : " . $shotgun->getting_the_color();
echo "\n";
echo "Capacity of the gun : " . $shotgun->getting_the_capacity();
echo "\n";

?>

</body>
</html>
```

$php main.php

```
<!DOCTYPE html>
<html>
<body>
```

Name of the bazooka : This is a hard-hitting bazooka ready for waging war.

Color of the bazooka : The color of my bazooka is black.

Capacity of the bazooka : I can load up to six rockets to my bazooka. This is a beast for the enemy.

Name of the revolver : This is a revolver to kill the guards at the gates silently.

Color of the revolver : The color of the revolver is silver grey.

Capacity of the revolver : I can load up to ten bullets to my revolver. This is a beast for the enemy.

Name of the rifle : This is an Ak47 to kill the guards at the gates silently.

Color of the rifle : The color of the gun is brown.

Capacity of the rifle : I can load up to twenty bullets to my gun. This is a beast for the enemy.

Name of the gun : This is a shotgun to kill the enemy with a bang.

Color of the gun : The color of the gun is brown.

Capacity of the gun : I can load up to two bullets to my gun. This is a beast for the enemy.

</body>

</html>

PHP The_construct Function

A PHP constructor permits you to kick off an object's properties upon the creation of the object. If you create a _construct() function, PHP will call the function when you build an object. The construct function starts with a couple of underscores(__).

```
<!DOCTYPE html>

<html>

<body>

<?php

class Guns {

  // Class Properties

  public $gname ;

  public $gcolor;

  public $gcapacity;

  // Class Methods

  function __construct($gname, $gcolor, $gcapacity) {

    $this->gname = $gname;

    $this->gcolor = $gcolor;

    $this->gcapacity = $gcapacity;

  }
```

```
  function getting_the_name() {
    return $this->gname;
  }

  function getting_the_color() {
    return $this->gcolor;
  }

  function getting_the_capacity() {
    return $this->gcapacity;
  }

}

$bazooka = new Guns('This is a hard-hitting bazooka ready for waging war.', 'The color of my bazooka is black.', 'I can load up to six rockets to my bazooka. This is a beast for the enemy.');

$revolver = new Guns('This is a revolver to kill the guards at the gates silently.', 'The color of the revolver is silver grey.', 'I can load up to ten bullets to my revolver. This is a beast for the enemy.');

$Ak47 = new Guns('This is an Ak47 to kill the guards at the gates silently.', 'The color of the gun is brown.', 'I can load up to twenty bullets to my gun. This is a beast for the enemy.');

$shotgun = new Guns('This is a shotgun to kill the enemy with a bang.', 'The color of the gun is brown.', 'I can load up to two bullets to my gun. This is a beast for the enemy.');

echo "Name of the bazooka : " . $bazooka->getting_the_name();

echo "\n";

echo "Color of the bazooka : " . $bazooka->getting_the_color();

echo "\n";
```

```
echo "Capacity of the bazooka : " . $bazooka->getting_the_capacity();
echo "\n";

echo "Name of the revolver : " . $revolver->getting_the_name();
echo "\n";
echo "Color of the revolver : " . $revolver->getting_the_color();
echo "\n";
echo "Capacity of the revolver : " . $revolver->getting_the_capacity();
echo "\n";

echo "Name of the rifle : " . $Ak47->getting_the_name();
echo "\n";
echo "Color of the rifle : " . $Ak47->getting_the_color();
echo "\n";
echo "Capacity of the rifle : " . $Ak47->getting_the_capacity();
echo "\n";

echo "Name of the gun : " . $shotgun->getting_the_name();
echo "\n";
echo "Color of the gun : " . $shotgun->getting_the_color();
echo "\n";
echo "Capacity of the gun : " . $shotgun->getting_the_capacity();
echo "\n";

?>

</body>
</html>
```

$php main.php

<!DOCTYPE html>

<html>

<body>

Name of the bazooka : This is a hard-hitting bazooka ready for waging war.

Color of the bazooka : The color of my bazooka is black.

Capacity of the bazooka : I can load up to six rockets to my bazooka. This is a beast for the enemy.

Name of the revolver : This is a revolver to kill the guards at the gates silently.

Color of the revolver : The color of the revolver is silver grey.

Capacity of the revolver : I can load up to ten bullets to my revolver. This is a beast for the enemy.

Name of the rifle : This is an Ak47 to kill the guards at the gates silently.

Color of the rifle : The color of the gun is brown.

Capacity of the rifle : I can load up to twenty bullets to my gun. This is a beast for the enemy.

Name of the gun : This is a shotgun to kill the enemy with a bang.

Color of the gun : The color of the gun is brown.

Capacity of the gun : I can load up to two bullets to my gun. This is a beast for the enemy.

</body>

</html>

The __destruct Function of PHP

A destructor is usually called when you have to destruct an object that you had earlier on created. You can also call it if you have to stop or exit the

script. If you create the __destruct() function, PHP will call it at the end of your script. The destruct function kicks off with underscores just as the constructor does. I will destruct an object in the following example.

```
<!DOCTYPE html>
<html>
<body>

<?php
class Guns {
  // Class Properties
  var $gname;
  var $gcolor;
  var $gcapacity;

  // Class Methods
  function __construct($gname, $gcolor, $gcapacity) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
  }
  function getting_the_name() {
    return $this->gname;
  }

  function getting_the_color() {
    return $this->gcolor;
  }

  function getting_the_capacity() {
```

```
    return $this->gcapacity;
  }
function __destruct() {
    echo "$this->gname\n. $this->gcolor\n. $this->gcapacity\n";
}
}
$bazooka = new Guns('This is a hard-hitting bazooka ready for waging war.', 'The color of my bazooka is black.', 'I can load up to six rockets to my bazooka. This is a beast for the enemy.');
$revolver = new Guns('This is a revolver to kill the guards at the gates silently.', 'The color of the revolver is silver grey.', 'I can load up to ten bullets to my revolver. This is a beast for the enemy.');
$Ak47 = new Guns('This is an Ak47 to kill the guards at the gates silently.', 'The color of the gun is brown.', 'I can load up to twenty bullets to my gun. This is a beast for the enemy.');
$shotgun = new Guns('This is a shotgun to kill the enemy with a bang.', 'The color of the gun is brown.', 'I can load up to two bullets to my gun. This is a beast for the enemy.');
echo "Name of the bazooka : " . $bazooka->getting_the_name();
echo "\n";
echo "Color of the bazooka : " . $bazooka->getting_the_color();
echo "\n";
echo "Capacity of the bazooka : " . $bazooka->getting_the_capacity();
echo "\n";

echo "Name of the revolver : " . $revolver->getting_the_name();
echo "\n";
echo "Color of the revolver : " . $revolver->getting_the_color();
echo "\n";
```

```
echo "Capacity of the revolver : " . $revolver->getting_the_capacity();
echo "\n";

echo "Name of the rifle : " . $Ak47->getting_the_name();
echo "\n";
echo "Color of the rifle : " . $Ak47->getting_the_color();
echo "\n";
echo "Capacity of the rifle : " . $Ak47->getting_the_capacity();
echo "\n";

echo "Name of the gun : " . $shotgun->getting_the_name();
echo "\n";
echo "Color of the gun : " . $shotgun->getting_the_color();
echo "\n";
echo "Capacity of the gun : " . $shotgun->getting_the_capacity();
echo "\n";

?>

</body>
</html>
```

$php main.php

```
<!DOCTYPE html>
<html>
<body>

Name of the bazooka : This is a hard-hitting bazooka ready for waging war.
Color of the bazooka : The color of my bazooka is black.
```

Capacity of the bazooka : I can load up to six rockets to my bazooka. This is a beast for the enemy.

Name of the revolver : This is a revolver to kill the guards at the gates silently.

Color of the revolver : The color of the revolver is silver grey.

Capacity of the revolver : I can load up to ten bullets to my revolver. This is a beast for the enemy.

Name of the rifle : This is an Ak47 to kill the guards at the gates silently.

Color of the rifle : The color of the gun is brown.

Capacity of the rifle : I can load up to twenty bullets to my gun. This is a beast for the enemy.

Name of the gun : This is a shotgun to kill the enemy with a bang.

Color of the gun : The color of the gun is brown.

Capacity of the gun : I can load up to two bullets to my gun. This is a beast for the enemy.

</body>

</html>

This is a shotgun to kill the enemy with a bang.

. The color of the gun is brown.

. I can load up to two bullets to my gun. This is a beast for the enemy.

This is an Ak47 to kill the guards at the gates silently.

. The color of the gun is brown.

. I can load up to twenty bullets to my gun. This is a beast for the enemy.

This is a revolver to kill the guards at the gates silently.

. The color of the revolver is silver grey.

. I can load up to ten bullets to my revolver. This is a beast for the enemy.

This is a hard-hitting bazooka ready for waging war.

. The color of my bazooka is black.

. I can load up to six rockets to my bazooka. This is a beast for the enemy.

PHP Access Modifiers

In the following example, I will add two access modifiers to the two methods. If you try to call the additional two functions for color and capacity, you will see a fatal error as the two are considered as private or protected even though the properties in the code are public.

```
<!DOCTYPE html>
<html>
<body>

<?php
class Guns {
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity;

  // Class Methods
  function setting_the_name ($n) {
    $this->gname = $n;
  }

  protected function setting_the_color($n) {
    $this->gcolor = $n;
  }

  private function getting_the_capacity($n) {
```

```
    $this->gcapacity = $n;
}
}
$bazooka = new Guns();
$bazooka->setting_the_name('This is a hard-hitting bazooka ready for waging war.');
$bazooka->setting_the_color('The color of my bazooka is black.');
$bazooka->setting_the_capacity('I can load up to six rockets to my bazooka. This is a beast for the enemy.');

$revolver = new Guns
$revolver->setting_the_name('This is a revolver to kill the guards at the gates silently.');
$revolver->setting_the_color('The color of the revolver is silver grey.');
$revolver->setting_the_capacity('I can load up to ten bullets to my revolver. This is a beast for the enemy.');

$Ak47 = new Guns
$Ak47->setting_the_name('This is an Ak47 to kill the guards at the gates silently.');
$Ak47->setting_the_color('The color of the gun is brown.');
$Ak47->setting_the_capacity('I can load up to twenty bullets to my gun. This is a beast for the enemy.');

$shotgun = new Guns
$shotgun->setting_the_name('This is a shotgun to kill the enemy with a bang.');
$shotgun->setting_the_color('The color of the gun is brown.');
$shotgun->setting_the_capacity('I can load up to two bullets to my gun. This is a beast for the enemy.');
```

```
echo "Name of the bazooka : " . $bazooka->getting_the_name();
echo "\n";
echo "Color of the bazooka : " . $bazooka->getting_the_color();
echo "\n";
echo "Capacity of the bazooka : " . $bazooka->getting_the_capacity();
echo "\n";

echo "Name of the revolver : " . $revolver->getting_the_name();
echo "\n";
echo "Color of the revolver : " . $revolver->getting_the_color();
echo "\n";
echo "Capacity of the revolver : " . $revolver->getting_the_capacity();
echo "\n";

echo "Name of the rifle : " . $Ak47->getting_the_name();
echo "\n";
echo "Color of the rifle : " . $Ak47->getting_the_color();
echo "\n";
echo "Capacity of the rifle : " . $Ak47->getting_the_capacity();
echo "\n";

echo "Name of the gun : " . $shotgun->getting_the_name();
echo "\n";
echo "Color of the gun : " . $shotgun->getting_the_color();
echo "\n";
echo "Capacity of the gun : " . $shotgun->getting_the_capacity();
echo "\n";
```

```
?>

</body>

</html>
```

$php main.php

PHP Parse error: syntax error, unexpected '$revolver' (T_VARIABLE) in /home/cg/root/8955485/main.php on line 36

Inheritance Classes

Inheritance in PHP works as well as it does in Python and other programming languages. Inheritance in PHP happens when a particular class tends to derive from another class. The child class is likely to inherit the public and private properties of its parent class. You can add up to it additional methods or properties to make the program more complex and functional. PHP gives the extends keyword to create inherited classes.

```
<!DOCTYPE html>

<html>

<body>


<?php

class Guns {

  // Class Properties

  public $gname;

  public $gcolor;

  public $gcapacity;


  public function __construct($gname, $gcolor, $gcapacity) {

    $this->gname = $gname;

    $this->gcolor = $gcolor;

    $this->gcapacity = $gcapacity;
```

```
  }

  public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname} . {$this->gcolor} . {$this->gcapacity}\n";
  }
}

class bazooka extends Guns {
  public function msg() {
    echo "I have a bazooka to win over the enemy should they emerge from their death holes in the future.\n";
}
}

$bazooka1 = new bazooka ('This is a hard-hitting bazooka ready for waging war.', 'The color of my bazooka is black.', 'I can load up to six rockets to my bazooka. This is a beast for the enemy.');
$bazooka1->msg();
$bazooka1->intro();

class revolver extends Guns {
  public function msg() {
    echo "I have a revolver to win over the enemy should they emerge from their death holes in the future.";
}
}

$revolver1 = new Revolver('This is a revolver to kill the guards at the gates silently.', 'The color of the revolver is silver grey.', 'I can load up to ten bullets to my revolver. This is a beast for the enemy.');
```

```
$revolver1->msg();
$revolver1->intro();

class ak47 extends Guns {
  public function msg() {
    echo "I have an ak47 to win over the enemy should they emerge from their death holes in the future.";
}
}

$Ak471 = new ak47('This is an Ak47 to kill the guards at the gates silently.', 'The color of the gun is brown.', 'I can load up to twenty bullets to my gun. This is a beast for the enemy.');
$Ak471->msg();
$Ak471->intro();

class shotgun extends Guns {
  public function msg() {
    echo "I have a shotgun to win over the enemy should they emerge from their death holes in the future.";
}
}

$shotgun1 = new Guns('This is a shotgun to kill the enemy with a bang.', 'The color of the gun is brown.', 'I can load up to two bullets to my gun. This is a beast for the enemy.');
$shotgun1->msg();
$shotgun1->intro();
```

?>

</body>

</html>

$php main.php

<!DOCTYPE html>

<html>

<body>

I have a bazooka to win over the enemy should they emerge from their death holes in the future.

My gun is too powerful for you to handle on the battlefield. Try something else. This is a hard-hitting bazooka ready for waging war. . The color of my bazooka is black. . I can load up to six rockets to my bazooka. This is a beast for the enemy.

I have a revolver to win over the enemy should they emerge from their death holes in the future. My gun is too powerful for you to handle on the battlefield. Try something else. This is a revolver to kill the guards at the gates silently. . The color of the revolver is silver grey. . I can load up to ten bullets to my revolver. This is a beast for the enemy.

I have an ak47 to win over the enemy should they emerge from their death holes in the future. My gun is too powerful for you to handle on the battlefield. Try something else. This is an Ak47 to kill the guards at the gates silently. . The color of the gun is brown. . I can load up to twenty bullets to my gun. This is a beast for the enemy.

In the above example, I have inherited four classes from the parent class to describe four different guns. All the four classes had access to the properties of the parent class because I had made the functions public. I also added an exclusive method to the inherited class

Protected Access Modifiers for the Parent Class

<!DOCTYPE html>

```
<html>
<body>

<?php
class Guns {
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity;

  public function __construct($gname, $gcolor, $gcapacity) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
  }

  protected function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity}\n";
  }
}

class bazooka extends Guns {
  public function msg() {
    echo "I have a bazooka to win over the enemy should they emerge from their death holes in the future.\n";
}
}
```

```
$bazooka1 = new bazooka ('This is a hard-hitting bazooka ready for waging war.', 'The color of my bazooka is black.', 'I can load up to six rockets to my bazooka. This is a beast for the enemy.');
$bazooka1->msg();


class revolver extends Guns {
  public function msg() {
    echo "I have a revolver to win over the enemy should they emerge from their death holes in the future.\n";
}
}

$revolver1 = new Revolver('This is a revolver to kill the guards at the gates silently.', 'The color of the revolver is silver grey.', 'I can load up to ten bullets to my revolver. This is a beast for the enemy.');
$revolver1->msg();


class ak47 extends Guns {
  public function msg() {
    echo "I have an ak47 to win over the enemy should they emerge from their death holes in the future.\n";
}
}

$Ak471 = new ak47('This is an Ak47 to kill the guards at the gates silently.', 'The color of the gun is brown.', 'I can load up to twenty bullets to my gun. This is a beast for the enemy.');
$Ak471->msg();
```

```
class shotgun extends Guns {
  public function msg() {
    echo "I have a shotgun to win over the enemy should they emerge from their death holes in the future.\n";
}
}

$shotgun1 = new shotgun('This is a shotgun to kill the enemy with a bang.', 'The color of the gun is brown.', 'I can load up to two bullets to my gun. This is a beast for the enemy.');
$shotgun1->msg();

?>

</body>
</html>
```

$php main.php

```
<!DOCTYPE html>
<html>
<body>

I have a bazooka to win over the enemy should they emerge from their death holes in the future.

I have a revolver to win over the enemy should they emerge from their death holes in the future.

I have an ak47 to win over the enemy should they emerge from their death holes in the future.

I have a shotgun to win over the enemy should they emerge from their death holes in the future.

</body>
```

```
</html>
```

When I try to use the protected function for the inherited classes, I will get a fatal error in return. Once protected, you cannot use it for the inherited class. See the following example to clear your concept.

```
<!DOCTYPE html>
<html>
<body>

<?php
class Guns {
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity;

  public function __construct($gname, $gcolor, $gcapacity) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
  }

  protected function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity}\n";
  }
}

class bazooka extends Guns {
```

```
  public function msg() {
    echo "I have a bazooka to win over the enemy should they emerge from their death holes in the future.\n";
}
}

$bazooka1 = new bazooka ('This is a hard-hitting bazooka ready for waging war.', 'The color of my bazooka is black.', 'I can load up to six rockets to my bazooka. This is a beast for the enemy.');
$bazooka1->msg();
$bazooka1->intro();

class revolver extends Guns {
  public function msg() {
    echo "I have a revolver to win over the enemy should they emerge from their death holes in the future.\n";
}
}

$revolver1 = new Revolver('This is a revolver to kill the guards at the gates silently.', 'The color of the revolver is silver grey.', 'I can load up to ten bullets to my revolver. This is a beast for the enemy.');
$revolver1->msg();
$revolver1->intro();

class ak47 extends Guns {
  public function msg() {
    echo "I have an ak47 to win over the enemy should they emerge from their death holes in the future.\n";
}
```

```
}

$Ak471 = new ak47('This is an Ak47 to kill the guards at the gates silently.', 'The color of the gun is brown.', 'I can load up to twenty bullets to my gun. This is a beast for the enemy.');

$Ak471->msg();

$Ak471->intro();

class shotgun extends Guns {

  public function msg() {

    echo "I have a shotgun to win over the enemy should they emerge from their death holes in the future.\n";

}

}

$shotgun1 = new shotgun('This is a shotgun to kill the enemy with a bang.', 'The color of the gun is brown.', 'I can load up to two bullets to my gun. This is a beast for the enemy.');

$shotgun1->msg();

$shotgun1->intro();

?>

</body>

</html>
```

$php main.php

```
<!DOCTYPE html>

<html>

<body>
```

I have a bazooka to win over the enemy should they emerge from their death holes in the future.

PHP Fatal error: Uncaught Error: Call to protected method Guns::intro() from context '' in /home/cg/root/4530372/main.php:31

Stack trace:

#0 {main}

  thrown in /home/cg/root/4530372/main.php on line 31

You can see that public methods worked well, but it returned an error when the editor came to the protected methods. However, there is a way out to keep the methods protected and yet use them for the inherited class. First, look at the following example. I will elaborate upon it after that.

```
<!DOCTYPE html>
<html>
<body>

<?php
class Guns {
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity;

  public function __construct($gname, $gcolor, $gcapacity) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
  }
```

```
  protected function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity}\n";
  }
}

class bazooka extends Guns {
  public function msg() {
    echo "I have a bazooka to win over the enemy should they emerge from their death holes in the future.\n";
    $this -> intro();
}
}

$bazooka1 = new bazooka ('This is a hard-hitting bazooka ready for waging war.', 'The color of my bazooka is black.', 'I can load up to six rockets to my bazooka. This is a beast for the enemy.');
$bazooka1->msg();

class revolver extends Guns {
  public function msg() {
    echo "I have a revolver to win over the enemy should they emerge from their death holes in the future.\n";
    $this -> intro();
}
}

$revolver1 = new Revolver('This is a revolver to kill the guards at the gates silently.', 'The color of the revolver is silver grey.', 'I can load up to ten
```

```
bullets to my revolver. This is a beast for the enemy.');
$revolver1->msg();

class ak47 extends Guns {
  public function msg() {
    echo "I have an ak47 to win over the enemy should they emerge from their death holes in the future.\n";
    $this -> intro();
}
}

$Ak471 = new ak47('This is an Ak47 to kill the guards at the gates silently.', 'The color of the gun is brown.', 'I can load up to twenty bullets to my gun. This is a beast for the enemy.');
$Ak471->msg();


class shotgun extends Guns {
  public function msg() {
    echo "I have a shotgun to win over the enemy should they emerge from their death holes in the future.\n";
    $this -> intro();
}
}

$shotgun1 = new shotgun('This is a shotgun to kill the enemy with a bang.', 'The color of the gun is brown.', 'I can load up to two bullets to my gun. This is a beast for the enemy.');
$shotgun1->msg();
```

?>

</body>

</html>

$php main.php

<!DOCTYPE html>

<html>

<body>

I have a bazooka to win over the enemy should they emerge from their death holes in the future.

My gun is too powerful for you to handle on the battlefield. Try something else. This is a hard-hitting bazooka ready for waging war.

. The color of my bazooka is black.

. I can load up to six rockets to my bazooka. This is a beast for the enemy.

I have a revolver to win over the enemy should they emerge from their death holes in the future.

My gun is too powerful for you to handle on the battlefield. Try something else. This is a revolver to kill the guards at the gates silently.

. The color of the revolver is silver grey.

. I can load up to ten bullets to my revolver. This is a beast for the enemy.

I have an ak47 to win over the enemy should they emerge from their death holes in the future.

My gun is too powerful for you to handle on the battlefield. Try something else. This is an Ak47 to kill the guards at the gates silently.

. The color of the gun is brown.

. I can load up to twenty bullets to my gun. This is a beast for the enemy.

I have a shotgun to win over the enemy should they emerge from their death holes in the future.

My gun is too powerful for you to handle on the battlefield. Try something else. This is a shotgun to kill the enemy with a bang.

. The color of the gun is brown.

. I can load up to two bullets to my gun. This is a beast for the enemy.

</body>

</html>

Overriding the Inherited Methods

You can override the methods of the inherited class by redefining them. One method which is also the simplest is to use the same name and confuse the editor. I will override some methods in the following example.

```
<!DOCTYPE html>

<html>

<body>

<?php

class Guns {

  // Class Properties

  public $gname;

  public $gcolor;

  public $gcapacity;

  public function __construct($gname, $gcolor, $gcapacity) {

    $this->gname = $gname ;

    $this->gcolor = $gcolor;

    $this->gcapacity = $gcapacity;

  }
```

```
  public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity}\n";
  }
}

class bazooka extends Guns {
    public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
  }

  public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$bazooka1 = new bazooka ('This is a hard-hitting bazooka ready for waging war.', 'The color of my bazooka is black.', 'I can load up to six rockets to my bazooka. This is a beast for the enemy.', 55);
$bazooka1->intro();

class revolver extends Guns {
    public $gweight;
```

```
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
}
public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$revolver1 = new revolver('This is a revolver to kill the guards at the gates silently.', 'The color of the revolver is silver grey.', 'I can load up to ten bullets to my revolver. This is a beast for the enemy.', 30);
$revolver1->intro();

class ak47 extends Guns {
  public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor ;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
}
public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} .
```

```
{$this->gweight}\n";
}
}

$Ak471 = new ak47('This is an Ak47 to kill the guards at the gates silently.', 'The color of the gun is brown.', 'I can load up to twenty bullets to my gun. This is a beast for the enemy.', 20);
$Ak471->intro();


class shotgun extends Guns {
  public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
}
public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$shotgun1 = new shotgun('This is a shotgun to kill the enemy with a bang.', 'The color of the gun is brown.', 'I can load up to two bullets to my gun. This is a beast for the enemy.', 15);
$shotgun1->intro();
```

?>

</body>

</html>

$php main.php

<!DOCTYPE html>

<html>

<body>

My gun is too powerful for you to handle on the battlefield. Try something else. This is a hard-hitting bazooka ready for waging war.

. The color of my bazooka is black.

. I can load up to six rockets to my bazooka. This is a beast for the enemy. . 55

My gun is too powerful for you to handle on the battlefield. Try something else. This is a revolver to kill the guards at the gates silently.

. The color of the revolver is silver grey.

. I can load up to ten bullets to my revolver. This is a beast for the enemy. . 30

My gun is too powerful for you to handle on the battlefield. Try something else. This is an Ak47 to kill the guards at the gates silently.

. The color of the gun is brown.

. I can load up to twenty bullets to my gun. This is a beast for the enemy. . 20

My gun is too powerful for you to handle on the battlefield. Try something else. This is a shotgun to kill the enemy with a bang.

. The color of the gun is brown.

. I can load up to two bullets to my gun. This is a beast for the enemy. . 15

</body>

</html>

PHP Final Keyword

The final keyword may be used to restrict inheritance function of classes or to put a stopper to the method of overriding.

```
<!DOCTYPE html>
<html>
<body>

<?php
final class Guns {
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity;

  public function __construct($gname, $gcolor, $gcapacity) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity ;
  }

  public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity}\n";
  }
}
```

```
class bazooka extends Guns {
    public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
  }

  public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$bazooka1 = new bazooka ('This is a hard-hitting bazooka ready for waging war.', 'The color of my bazooka is black.', 'I can load up to six rockets to my bazooka. This is a beast for the enemy.', 55);
$bazooka1->intro();

class revolver extends Guns {
    public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
}
```

```
public function intro() {
    echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$revolver1 = new revolver('This is a revolver to kill the guards at the gates silently.', 'The color of the revolver is silver grey.', 'I can load up to ten bullets to my revolver. This is a beast for the enemy.', 30);
$revolver1->intro();

class ak47 extends Guns {
  public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
}
public function intro() {
    echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$Ak471 = new ak47('This is an Ak47 to kill the guards at the gates silently.', 'The color of the gun is brown.', 'I can load up to twenty bullets to my gun. This is a beast for the enemy.', 20);
```

```
$Ak471->intro();


class shotgun extends Guns {
  public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
}
public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$shotgun1 = new shotgun('This is a shotgun to kill the enemy with a bang.', 'The color of the gun is brown.', 'I can load up to two bullets to my gun. This is a beast for the enemy.', 15);
$shotgun1->intro();

?>

</body>
</html>
```

$php main.php

PHP Fatal error: Class bazooka may not inherit from final class (Guns) in /home/cg/root/4530372/main.php on line 35

The return message is self-explanatory.

Prevention of the Overriding of Methods

Now I will restrict the overriding of methods.

```
<!DOCTYPE html>
<html>
<body>

<?php
final class Guns {
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity;

  final public function __construct($gname, $gcolor, $gcapacity) {
    $this->gname = $gname;
    $this->gcolor = $gcolor ;
    $this->gcapacity = $gcapacity;
  }

  public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity}\n";
  }
}

class bazooka extends Guns {
```

```
    public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
  }

  public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$bazooka1 = new bazooka ('This is a hard-hitting bazooka ready for waging war.', 'The color of my bazooka is black.', 'I can load up to six rockets to my bazooka. This is a beast for the enemy.', 55);
$bazooka1->intro();

class revolver extends Guns {
    public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
}
public function intro() {
```

```
        echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$revolver1 = new revolver('This is a revolver to kill the guards at the gates silently.', 'The color of the revolver is silver grey.', 'I can load up to ten bullets to my revolver. This is a beast for the enemy.', 30);
$revolver1->intro();

class ak47 extends Guns {
  public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity ;
    $this->gweight = $gweight;
}
public function intro() {
        echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$Ak471 = new ak47('This is an Ak47 to kill the guards at the gates silently.', 'The color of the gun is brown.', 'I can load up to twenty bullets to my gun. This is a beast for the enemy.', 20);
$Ak471->intro();
```

```
class shotgun extends Guns {
  public $gweight;
    public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
    $this->gweight = $gweight;
}
public function intro() {
      echo "My gun is too powerful for you to handle on the battlefield. Try something else. {$this->gname}\n . {$this->gcolor}\n . {$this->gcapacity} . {$this->gweight}\n";
}
}

$shotgun1 = new shotgun('This is a shotgun to kill the enemy with a bang.', 'The color of the gun is brown.', 'I can load up to two bullets to my gun. This is a beast for the enemy.', 15);
$shotgun1->intro();


?>

</body>
</html>
```

$php main.php

```
PHP Fatal error:  Class bazooka may not inherit from final class (Guns) in /home/cg/root/4530372/main.php on line 35
```

PHP Abstract Classes

Abstract classes or methods happen in a code when the parent class names a method but it requires the child class to wrap up the tasks. An abstract class usually contains one abstract method. You have to declare the abstract method. The keyword for the task is called abstract.

During the process of inheritance from abstract classes, you have to define the child class with the same name. If you define the abstract method as protected, the child class method ought to be defined as public or protected. It should not be defined as private. The number and type of the requisite arguments need to be the same. There may be optional arguments in the child classes.

The methods of the child class need to have the same name as the abstract class does. The access modifiers must be the same so should be the requisite arguments. There may be some optional arguments that you like to add.

```
<!DOCTYPE html>
<html>
<body>

<?php
abstract class Guns {
  // Class Properties
  public $gname;
  public $gcolor;
  public $gcapacity;
  public $weight;

  public function __construct($gname, $gcolor, $gcapacity, $gweight) {
    $this->gname = $gname;
    $this->gcolor = $gcolor;
    $this->gcapacity = $gcapacity;
```

```
    $this->gweight = $gweight;
  }

  abstract public function intro() : string ;
}

class Bazooka extends Guns {
  public function intro() : string {
      return "My gun is too powerful for you to handle on the battlefield. Try something else. $this->gname \n . $this->gcolor\n . $this->gcapacity . $this->gweight \n";
}
}


class Revolver extends Guns {
public function intro() : string {
      return "My gun is too powerful for you to handle on the battlefield. Try something else. $this->gname \n . $this->gcolor \n . $this->gcapacity . $this->gweight \n";
}
}


class Ak47 extends Guns {
public function intro() : string {
      return "My gun is too powerful for you to handle on the battlefield. Try something else. $this->gname \n . $this->gcolor \n . $this->gcapacity . $this->gweight\n";
}
```

```
}
class Shotgun extends Guns {
  public function intro() : string {
      return "My gun is too powerful for you to handle on the battlefield. Try something else. $this->gname \n . $this->gcolor \n . $this->gcapacity . $this->gweight \n";

}
}

$bazooka = new bazooka ('This is a hard-hitting bazooka ready for waging war.', 'The color of my bazooka is black.', 'I can load up to six rockets to my bazooka. This is a beast for the enemy.', 55);
echo $bazooka->intro();
echo "\n";

$revolver = new revolver('This is a revolver to kill the guards at the gates silently.', 'The color of the revolver is silver grey.', 'I can load up to ten bullets to my revolver. This is a beast for the enemy.', 30);
echo $revolver->intro();
echo "\n";

$ak47 = new ak47('This is an Ak47 to kill the guards at the gates silently.', 'The color of the gun is brown.', 'I can load up to twenty bullets to my gun. This is a beast for the enemy.', 20);
echo $ak47->intro();
echo "\n";

$shotgun = new shotgun('This is a shotgun to kill the enemy with a bang.', 'The color of the gun is brown.', 'I can load up to two bullets to my gun. This is a beast for the enemy.', 15);
```

```
echo $shotgun->intro();
echo "\n";

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

$php main.php

```
<!DOCTYPE html>
<html>
<body>
```

My gun is too powerful for you to handle on the battlefield. Try something else. This is a hard-hitting bazooka ready for waging war.

. The color of my bazooka is black.

. I can load up to six rockets to my bazooka. This is a beast for the enemy.
.

My gun is too powerful for you to handle on the battlefield. Try something else. This is a revolver to kill the guards at the gates silently.

. The color of the revolver is silver grey.

. I can load up to ten bullets to my revolver. This is a beast for the enemy.
.

My gun is too powerful for you to handle on the battlefield. Try something else. This is an Ak47 to kill the guards at the gates silently.

. The color of the gun is brown.

. I can load up to twenty bullets to my gun. This is a beast for the enemy. .

My gun is too powerful for you to handle on the battlefield. Try something else. This is a shotgun to kill the enemy with a bang.

. The color of the gun is brown.

. I can load up to two bullets to my gun. This is a beast for the enemy. .

</body>

</html>

PHP Class Interfaces

PHP interfaces will allow you to specify what kind of methods a PHP class needs to implement. PHP interfaces eases off the use a different classes in the similar manner. When more than one classes use an interface, the process is named as polymorphism. PHP interfaces are usually declared with the help of the interface keyword.

Interface works in the same way as abstract classes do. The only difference is that PHP interfaces carry properties while abstract classes carry them as per the needs of the programmer. Interface methods ought to be public while abstract methods may be either public or protected. All interface methods usually are abstract that is why they can never be implemented in the code.

The keyword that is used for the purpose is titled as implements keyword.

```
<!DOCTYPE html>
<html>
<body>

<?php
interface Guns {

  public function description();

  }

class Bazooka implements Guns {
  public function description() {
```

```
        echo "My gun is too powerful for you to handle on the battlefield. Try something else. \n";
}
}

class Revolver implements Guns {
public function description() {
        echo "My gun is too powerful for you to handle on the battlefield. Try something else.\n";
}
}

class Ak47 implements Guns {
public function description() {
        echo "My gun is too powerful for you to handle on the battlefield. Try something else.\n";
}
}

class Shotgun implements Guns {
   public function description() {
        echo "My gun is too powerful for you to handle on the battlefield. Try something else. \n";

}
}

$bazooka = new bazooka ();
```

```
$revolver = new revolver();
$ak47 = new ak47();
$shotgun = new shotgun();
$guns = array($bazooka, $revolver, $ak47, $shotgun);

foreach($guns as $gun) {
    $gun->description();
}

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

$php main.php

```
<!DOCTYPE html>
<html>
<body>

My gun is too powerful for you to handle on the battlefield. Try something else.
My gun is too powerful for you to handle on the battlefield. Try something else.
My gun is too powerful for you to handle on the battlefield. Try something else.
My gun is too powerful for you to handle on the battlefield. Try something else.
</body>
</html>
```

Chapter Eight: Database Creation in PHP

MySQL is the database that you can use only on the website. You have to connect it to a server to run it successfully. You can store data in MySQL or SQL databases in the form of tables. A table is a collection of interrelated data and it comprises of rows and columns. You can use databases to store information categorically. A company may build a database in tables such as customers, employees, products, and orders.

You have to download the PHP server with a MySQL Database, you may download for free from the internet. MySQL is used to store huge volumes of data.

Make the Connection

The first step is to build a connection between PHP and database.

```
<!DOCTYPE html>
<html>
<body>

<?php
$theservername = "this is a localhost";
$theusername = "this is the username box";
$thepassword = "this is the password box";

// Create connection
$myconnection = new mysqli($theservername, $theusername, $thepassword);

// Check connection
if ($myconnection->connect_error) {
  die("Connection has fully failed: " . $myconnection->connect_error);
}
echo "You have been connected successfully to the database";
```

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

$php main.php

```
<!DOCTYPE html>
<html>
<body>

Connection has fully failed: php_network_getaddresses: getaddrinfo failed: Name or service not knownPHP Warning: mysqli::__construct(): php_network_getaddresses: getaddrinfo failed: Name or service not known in /home/cg/root/4530372/main.php on line 11

PHP Warning: mysqli::__construct(): (HY000/2002): php_network_getaddresses: getaddrinfo failed: Name or service not known in /home/cg/root/4530372/main.php on line 11
```

If your connection fails, you will get the above mentioned result.

MySQLi Procedural

This is another example for connecting PHP to MySQLi database.

```
<!DOCTYPE html>
<html>
<body>

<?php
$theservername = "this is a localhost";
$theusername = "this is the username box";
$thepassword = "this is the password box";

// Create connection
```

```
$myconnection = new mysqli_connect($theservername, $theusername, $thepassword);

// Check connection
if (!$myconnection) {
  die("Connection has fully failed: " . $myconnection->connect_error);
}
echo "You have been connected successfully to the database";

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

Database Creation

In this example, I will write the code that will help you create a database to load up on your website.

```
<!DOCTYPE html>
<html>
<body>

<?php
$theservername = "this is a localhost";
$theusername = "this is the username box";
$thepassword = "this is the password box";

// Create connection
$myconnection = new mysqli_connect($theservername, $theusername, $thepassword);

// Check connection
```

```
if (!$myconnection) {
  die("Connection has fully failed: " . $myconnection->connect_error);
}
echo "You have been connected successfully to the database";

// This will help you create a database
$sql = "I am trying to create a database";
if ($myconnection->query($sql) === TRUE) {
  echo "Thank you, you have got your database. You are now free to use it.";
} else {
  echo "There is an error in the creation of the database: " . $myconnection->error;
}

$myconnection->close();

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

Database Creation in Procedural MySQLi

```
<!DOCTYPE html>
<html>
<body>

<?php
$theservername = "this is a localhost";
$theusername = "this is the username box";
```

```
$thepassword = "this is the password box";

// Create connection
$myconnection = new mysqli_connect($theservername, $theusername, $thepassword);

// Check connection
if (!$myconnection) {
  die("Connection has fully failed: " . $myconnection->connect_error);
}
echo "You have been connected successfully to the database";

// This will help you create a database
$sql = "I am trying to create a database";
if (mysqli_query($myconnection, $sql)) {
    echo "Thank you, you have got your database. You are now free to use it.";
} else {
  echo "There is an error in the creation of the database: " . mysqli_error($myconnection);
}

$mysqli_close($myconnection);

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

Table Creation

Here is the code block to create a table to be used with PHP.

```
<!DOCTYPE html>
<html>
<body>

<?php
$theservername = "this is a localhost";
$theusername = "this is the username box";
$thepassword = "this is the password box";

// Create connection
$myconnection = new mysqli_connect($theservername, $theusername, $thepassword);
if (!$myconnection) {
  die("Connection has fully failed: " . $myconnection->connect_error);
}
echo "You have been connected successfully to the database";

$sql = "CREATING THE TABLE BuyingDetails (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
Serial# VARCHAR(30) NOT NULL,
BuyerName VARCHAR(30) NOT NULL,
PostalAddress VARCHAR(30) NOT NULL,
Age VARCHAR(30) NOT NULL,
Nameofcountry VARCHAR(30) NOT NULL,
MoneySpent VARCHAR(30) ,
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
)";
```

```
if (mysqli_query($myconnection, $sql)) {
    echo "Thank you, you have got your table. You can use it as per need.";
} else {
  echo "There is an error in the creation of the database: " . mysqli_error($myconnection);
}

$mysqli_close($myconnection);

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

Chapter Nine: PHP and SQL Database Commands

When you are building a website with PHP's help, you have to learn how to use SQL database because a website without a database is like a human body that only has the skeleton but lacks flesh and muscles. A database in SQL consists of tables. You have to identify the table by the name to easily locate the information you have stored in it. I will use the following database to experiment with SQL commands.

Serial#	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The design of the database should be correct before you start its creation. Otherwise, you will certainly have to move back and change it by tinkering the tables, merging them up, or moving up or down different columns to establish sensible relationships that MySQL may use. In this chapter, I will explain the use of different SQL commands to explain how each of them behaves.

SQL Column Selection

The following statement will select columns from the table. I will use the select command to explain how it works. The database namely BuyingDetails looks like the following:

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria	Street 7	45	United States	1000

	Fellar	Orington			
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The command is as under:

SELECT BuyerName, Nameofcountry FROM BuyingDetails

BuyerName	Nameofcountry
Temeria Fellar	United States
Pastor Kemari	Canada
Tim Miller	United States
Jasmine Tim Kesan	Sweden
Kevin Nevada	Spain
Mark Flagrar	United States
Stanley Cooper	Italy
Justin Goldminer	Canada
Sylvinia Marko	United Kingdom
Adam Toffin	United Kingdom

If you are not sure about which column you need to work on, you can use the * command to select all the columns of the table. The original table is as under:

Serial	BuyerName	PostalAddress	A	Nameofcountry	MoneySpent

#			ge		
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the SELECT statement:

SELECT * FROM BuyingDetails;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000

7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The command has selected all the columns of the table. This is the easiest way to see details of all columns. You can note down the names of columns from here and select the requisite columns by using the same SELECT command.

Another command known as DISTINCT syntax is used to eliminate duplicate entries in the database so that you may save time wasted on reading through the same entries repeatedly.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000
11	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the statement to make the selection:

SELECT DISTINCT column 1, column 2, column 3, column 4, column 5, column 6,

FROM BuyingDetails;

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The following command will select even the duplicate entries from the database.

SELECT * FROM BuyingDetails;

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000

5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000
11	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

WHERE Examples

You can use the WHERE clause to select all the buyers from a single country from the table. A demonstration will explain them to you in the most appropriate way.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000

10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The command is as under:

SELECT * FROM BuyingDetails

WHERE NameofCountry= ‘United States’;

Here is the result:

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
3	Tim Miller	Street 55 Irina	45	United States	1500
6	Mark Flagrar	Street 90 Time Square	25	United States	18000

In the following example, I will pick up the buyers from the United Kingdom. Here is the command.

SELECT * FROM BuyingDetails

WHERE Nameofcountry = ‘United Kingdom’ ;

This is the table in the original form.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000

8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the result:

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Text Fields

You can select the rows based on the serial number of the columns. Here is the table in its original form.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000

10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The following is the command:

SELECT * FROM BuyingDetails

WHERE Serial#=1;

WHERE Serial#=2;

WHERE Serial#=3;

WHERE Serial#=4;

WHERE Serial#=4;

WHERE Serial#=5;

WHERE Serial#=6;

WHERE Serial#=7;

WHERE Serial#=8;

WHERE Serial#=9;

WHERE Serial#=10;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000

10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The SQL AND, OR Operators

The WHERE clause may be paired up with AND, OR, and NOT operators. You can filter the table with these clauses. This is the current state of the table.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1000
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

SELECT * FROM BuyingDetails

WHERE Nameofcountry= ‘United States’ AND Moneyspent= 1000;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
3	Tim Miller	Street 55 Irina	45	United States	1000

The example of OR is as under. Here is the original form of the table.

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1000
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	55000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

SELECT * FROM Buyingdetails

WHERE Nameofcountry= ‘United States’ OR Moneyspent= 55000;

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
3	Tim Miller	Street 55 Irina	45	United States	1500
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	55000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The NOT Clause

In the following example, I will show how you can use the NOT clause to eliminate one row or column and display the rest. I suppose that the table in its original condition which is as under:

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The NOT clause is as under:

```
SELECT * FROM Buyingdetails
WHERE NOT Nameofcountry = ‘Spain’;
```

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500

4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The Double Not Clause

You have seen how we can use the NOT clause to remove elements. However, if you have to do away with two clauses, you can use the double NOT clause. The table is back to its original condition which is as follows:

Serial#	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam	Street 55	21	United Kingdom	55000

	Toffin	Nion Valley			

The following statement will select all the fields from the table where country is NOT Spain and NOT Italy.

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Deciphering the SQL ORDER BY Keyword

The ORDER BY keyword is generally used for sorting the results in either the ascending or descending orders. The ORDER BY keyword will sort the details as a default function. You will have to use the DESC keyword to bring it back into the descending order. The following is the present unordered state of the table.

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada	15	Spain	5000

		Olumpur			
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

I will write down the command:

SELECT * FROM BuyingDetails

ORDER BY Country

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000
1	Temeria Fellar	Street 7 Orington	45	United States	1000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
3	Tim Miller	Street 55 Irina	45	United States	1500

We can make further experiment with the ORDER BY clause by sorting the table according to the name of buyers. First of all, I will bring back the table into its original form.

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the code to do the job.

```
SELECT * FROM BuyingDetails
ORDER BY BuyerName;
```

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000

6	Mark Flagrar	Street 90 Time Square	25	United States	18000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
3	Tim Miller	Street 55 Irina	45	United States	1500
1	Temeria Fellar	Street 7 Orington	45	United States	1000

ORDER BY DESC

The following example will order the table in the descending order. The present state of the table is as under:

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

I will now write down the code to reorder the table in the descending order.

SELECT * FROM BuyingDetails

ORDER BY Nameofcountry DESC;

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000

SQL Insert Into Clause

You can use the INSERT INTO statement to add new rows and columns to an existing table. I will demonstrate with a real example of our table that we have been using for all examples.

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim	Street 87	20	Sweden	2000

	Kesan				
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The insert into statement is as under:

INSERT INTO BuyingDetails (BuyerName, PostalAddress, Age, Nameofcountry, MoneySpent)

VALUES (‘Eliseh Bentley’, ‘Street 45 Aliano Cameo’, ‘33’, ‘Mexico’, ‘5000’)

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55	21	United Kingdom	55000

		Nion Valley			
11	Eliseh Bentley	Street 45 Aliano Cameo	33	Mexico	5000

The UPDATE Statement

In this example, I will explain how you can use the UPDATE statement and what it does when you are building a website. When you have created a table, you can use the UPDATE statement to refresh the details in the specific rows and columns. One important thing to keep in mind is that you should not omit the WHERE clause from the code you will have unpredictable results. The update you want will not happen. Let us do some coding now. The following is the current state of the table.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the code for updating the table:

```
UPDATE BuyingDetails
SET BuyerName = ‘Alfred Simon’, Nameofcountry= ‘Germany’
WHERE Serial# = 1;
```


Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Alfred Simon	Street 7 Orington	45	Germany	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Along the same line, you can update multiple entries in the table. The WHERE clause helps you select the exact spot for making the update. See the following example. I will bring back the table to the former state for ease of reading and comprehension.

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000

6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The following example will update the country name from United Kingdom to Germany.

UPDATE BuyingDetails

SET Nameofcountry= ‘Germany’

WHERE Nameofcountry= ‘United States’;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	Germany	15000
10	Adam Toffin	Street 55 Nion Valley	21	Germany	55000

This clause is useful especially if an employee of your organization fills in the table with wrong entries.

What If You Fail To Use the WHERE Clause?

In this code sample, I will explain what dangers the omission of WHERE clause carries. There are of course no dangers while you are in the learning phase. However, when you are a business owner and you are operating a large database for business, the danger level is higher in the wake of simple mistakes. I will present the table in its original form for reference.

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Now I will write the code but omit the WHERE clause to see what the results are.

UPDATE BuyingDetails

SET Nameofcountry= ‘United States’

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor	Street 10	55	United States	500

	Kemari	Ontario			
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	United States	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	United States	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	United States	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	United States	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United States	15000
10	Adam Toffin	Street 55 Nion Valley	21	United States	55000

The DELETE Statement

When you are creating a table and working on updating it, you may face a situation in which you might have to delete some entries. You can use the DELETE statement to do away with the entries you don't want to see in your table. Like we had to keep on an eye on the WHERE clause in the UPDATE statement, we have to do the same when using the DELETE statement. Otherwise, we may delete precious data from the table. In addition, your entire table may be deleted with a single click. Here is the table in the present form.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90	25	United States	18000

		Time Square			
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the statement that I will be using to delete entries from the table:

DELETE FROM BuyingDetails WHERE BuyerName= ‘Adam Toffin’;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000

However, in some cases, you may want to remove all the entries from a table. This includes the attributes, structures and all the indices. There is a statement for that but you have to be extra careful in using it. Here is the present state of the table.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria	Street 7	45	United States	1000

	Fellar	Orington			
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

DELETE FROM BuyingDetails;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**

The most important thing is that the table will remain intact. Only the entries will be deleted.

TOP, ROWNUM, LIMIT Clauses

Here are some interesting clauses to use. You can use the SELECT TOP to specify the total number of entries that you want to return. The SELECT TOP clause will be useful when dealing with large tables with thousands of entries. If you choose to return the entire table, it will degrade the performance of the database. However, if you return only the entries you want to work on, it will hardly affect the performance unless they are too large. Here is the table on which I will use the TOP clause.

Serial	**BuyerName**	**PostalAddress**	**A**	**Nameofcountry**	**MoneySpent**

#			ge		
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the code to select the top 7 entries from the table.

SELECT TOP 7 * FROM BuyingDetails;

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000

7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000

There is another statement in SQL that does the same job but is easier to use. I am bringing back the table to the original form and then I will use that clause.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

SELECT * FROM BuyingDetails

LIMIT 7;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim	Street 87	20	Sweden	2000

	Kesan				
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000

Here is yet another technique to do the same job. When you have more than one methods to do a technical job, you feel more confident while working. Multiple choices allow you to experiment different things if one does not work. Here is the present state of the table.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the code that is equivalent to the above two methods.

```
SELECT * FROM BuyingDetails
WHERE ROWNUM <=7
```


Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000

The TOP 50 Percent

This clause is useful if you are working with large databases. This will select the top 50 percent of the table and display the entries. Here is the table.

Serial #	BuyerName	PostalAddress	Age	Nameofcountry	MoneySpent
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000

9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The TOP clause is as under:

SELECT TOP 50 PERCENT * FROM BuyingDetails;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000

You can add a WHERE clause to the TOP 50 percent clause for a customized result. See the following example.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000

8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the code:

SELECT TOP 2 “ FROM BuyingDetails

WHERE Nameofcountry= ‘United States’;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
3	Tim Miller	Street 55 Irina	45	United States	1500

The Wildcards

SQL is amazing in the sense that it allows you significant space for experimentation. I will use a statement to select the buyers that belong to the country that start with ‘Uni.’ I will use the following table.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin	Street 10	30	Canada	10000

	Goldminer	Trafalgar Square			
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the statement to select the specific countries that start with ‘Uni.’

SELECT * FROM BuyingDetails

WHERE Country LIKE ‘Uni%’;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
3	Tim Miller	Street 55 Irina	45	United States	1500
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

The following wildcard will select all the countries that end with ‘dom.’

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley	Street 45	33	Italy	15000

	Cooper	Insta Debian			
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the statement:

SELECT * FROM BuyingDetails

WHERE Nameofcountry LIKE ‘%dom%’;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

If you use the _ wildcards, you can be more selective in viewing the entries of the table. Here is the table in its original form.

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
1	Temeria Fellar	Street 7 Orington	45	United States	1000
2	Pastor Kemari	Street 10 Ontario	55	Canada	500
3	Tim Miller	Street 55 Irina	45	United States	1500
4	Jasmine Tim Kesan	Street 87	20	Sweden	2000
5	Kevin Nevada	Street 76 Kevada Olumpur	15	Spain	5000
6	Mark Flagrar	Street 90 Time Square	25	United States	18000
7	Stanley Cooper	Street 45 Insta Debian	33	Italy	15000
8	Justin Goldminer	Street 10 Trafalgar Square	30	Canada	10000
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000

10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Here is the wildcard statement:

SELECT * FROM BuyingDetails

WHERE Nameofcountry LIKE ‘_United Kingdom’;

Serial #	**BuyerName**	**PostalAddress**	**Age**	**Nameofcountry**	**MoneySpent**
9	Sylvinia Marko	Street 34 Nion Valley	33	United Kingdom	15000
10	Adam Toffin	Street 55 Nion Valley	21	United Kingdom	55000

Conclusion

Now that you have made it to the end of the book, the next step is to practice more. PHP is not an easy language when you pair it up with MySQL. You have to consider the details, the slight nuances in the code, and the no-go areas. I hope the book has equipped you well with the advanced level programming in PHP. I have given considerable space to object oriented programming and database so that you can take a step further and enter functional programming.

References

Nixon, R. (2014, June). *Learning PHP, MySQL, JavaScript, CSS & HTML5* X-Files.

PHP $_REQUEST . (n.d.). Www.W3schools.com. Retrieved from https://www.w3schools.com/php/php_superglobals_request.asp