COMPUTER PROGRAMMING 7 (ICT 12)
GRADE 12 LEARNER'S MODULE #1 & 2 – 3rd Quarter, Weeks 1 - 4

CHAPTER 1: INTRODUCTION TO PHP

OBJECTIVES: In this lesson, you will learn to:

- Understand the use of PHP.
- Analyze the product of the PHP.
- Define the different version of the program.
- Understand the syntax of the program
- Understand the function of each algorithm.
- Create a PHP program.
- Understand on how to run the program.
- Insert a comment lines.

Lesson 1: Understanding PHP

INTRODUCTION:

The following is a quick introduction and summary of many aspects of the PHP language for those who have some programming experience. Although this overview is not intended to be an exhaustive examination of PHP, it is comprehensive enough for you to get started building non-trivial web applications with PHP.

INSTRUCTION / DISCUSSION:

Background

• Nerdy recursive acronym: PHP: Hypertext Preprocessor (originally named Personal Home Page Tools)

• Invented by Rasmus Lerdorf in 1994 and is now under the Apache Software Foundation. Licensed under the GPL and is free. Current version as of October 2012 is PHP 5.4.8.

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Common Uses of PHP

PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them. The other uses of PHP are:

- PHP can handle forms, i.e. gather data from files, save data to a file, thru email you can send data, return data to the user.
- You add, delete, modify elements within your database thru PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible:

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

"Hello World" Script in PHP

To get a feel of PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script. As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML (or XHTML if you're cutting-edge) you'll have PHP statements like this:

```
<html>
<head>
<title>Hello World</title>
<body>
<?php echo”Hello World!”;?>
</body>
</html>
```

All PHP code must be included inside one of the three special markup tags ate are recognized by the PHP Parser.

```
<?php PHP code goes here ?>
<?    PHP code goes here ?>
<script language="php"> PHP code goes here </script>
```

Most common tag is the <?php...?> and we will also use the same tag in our tutorial.

Lesson 2: Environment Set up

In order to develop and run PHP Web pages, three vital components need to be installed on your computer system.

Web Server - PHP will work with virtually all **Web Server** software, including Microsoft's Internet Information Server (IIS) but then most often used is freely available Apache Server. Download Apache for free here: http://httpd.apache.org/download.cgi

Database - PHP will work with virtually all database software, including Oracle and Sybase but most commonly used is freely available MySQL database. Download MySQL for free here: http://www.mysql.com/downloads/index.html

Lesson 3: Syntax Overview

Escaping to PHP

The PHP parsing engine needs a way to differentiate PHP code from other elements in the page. The mechanism for doing so is known as 'escaping to PHP.' There are four ways to do this:

Canonical PHP tags

The most universally effective PHP tag style is:

```
<?php...?>
```

If you use this style, you can be positive that your tags will always be correctly interpreted.

Short-open (SGML-style) tags

Short or short-open tags look like this:

```
<?...?>
```

Short tags are, as one might expect, the shortest **option You must** do one of two things to enable PHP to recognize the tags:

- Choose the --enable-short-tags configuration option when you're building PHP.
- Set the short_open_tag setting in your php.ini file to on. This option must be disabled to parse XML with PHP because the same syntax is used for XML tags.

ASP-style tags

ASP-style tags mimic the tags used by Active Server Pages to delineate code blocks. ASP-style tags look like this:

```
<%...%>
```

To use ASP-style tags, you will need to set the configuration option in your php.ini file.

HTML script tags

HTML script tags look like this:

```
<script language="PHP">...</script>
```

Commenting PHP Code

A *comment* is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP:

Single-line comments: They are generally used for short explanations or notes relevant to the local code. Here are the examples of single line comments.

```
<?
# This is a comment, and
# This is the second line of the comment
//   This is a comment too. Each style comments only print "An
example with single line comments";
?>
```

Multi-lines printing: Here are the examples to print multiple lines in a single print statement:

```
<?
# First Example print <<<END

This uses the "here document" syntax to output multiple lines with
$variable interpolation. Note that the here document terminator
must appear on a line with just a semicolon no extra whitespace!
END;
# Second Example
print "This spans
multiple lines. The newlines will be
output as well";
?>
```

Multi-lines comments: They are generally used to provide pseudocode algorithms and more detailed explanations when necessary. The multiline style of commenting is the same as in C. Here are the example of multi lines comments.

```
<?
/* This is a comment with multiline
      Author : Mohammad Mohtashim
      Purpose: Multiline Comments Demo
      Subject: PHP
*/
print "An example with multi line comments"; ?>
```

PHP is whitespace insensitive

Whitespace is the stuff you type that is typically invisible on the screen, including spaces, tabs, and carriage returns (end-of-line characters).

PHP whitespace insensitive means that it almost never matters how many whitespace characters you have in a row.one whitespace character is the same as many such characters.

For example, each of the following PHP statements that assigns the sum of 2 + 2 to the variable $four is equivalent:

```
$four =2+2;      // single spaces
$four <tab>=<tab2<tab>+<tab>2 ; // spaces and tabs
$four =
2+
2; //    multiple   lines
```

PHP is case sensitive

Yeah it is true that PHP is a case sensitive language. Try out the following example:

```
<html>
<body>
<?
$capital = 67;
print("Variable capital is $capital<br>");
print("Variable CaPiTaL is $CaPiTaL<br>");
?>
</body>
</html>
```

This will produce the following result:

```
Variable capital is 67
Variable CaPiTaL is
```

Statements are expressions terminated by semicolons

A *statement* in PHP is any expression that is followed by a semicolon (;).Any sequence of valid PHP statements that is enclosed by the PHP tags is a valid PHP program. Here is a typical statement in PHP, which in this case assigns a string of characters to a variable called $greeting:

```
$greeting = "Welcome to PHP!";
```

Braces make blocks

Although statements cannot be combined like expressions, you can always put a sequence of statements anywhere a statement can go by enclosing them in a set of curly braces.
Here both statements are equivalent:

```
if (3 == 2 + 1)
   print("Good - I haven't totally lost my mind.<br>");
if (3 == 2 + 1)
{
    print("Good - I haven't totally");

    print("lost my mind.<br>");
}
```

Running PHP Script from Command Prompt

Yes you can run your PHP script on your command prompt. Assuming you have the following content in test.php file

```
<?php
    echo "Hello PHP!!!!!";
?>
```

Now run this script as command prompt as follows:

```
$ php test.php
```

It will produce the following result

```
Hello PHP!!!!!
```

Lesson 4: Variable Types

The main way to store information in the middle of a PHP program is by using a variable.
Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.
- PHP variables are Perl-like.

PHP has a total of eight data types which we use to construct our variables:

- **Integers:** are whole numbers, without a decimal point, like 4195.
- **Doubles:** are floating-point numbers, like 3.14159 or 49.1.
- **Booleans:** have only two possible values either true or false.
- **NULL:** is a special type that only has one value: NULL.
- **Strings:** are sequences of characters, like 'PHP supports string operations.'
- **Arrays:** are named and indexed collections of other values.
- **Objects:** are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources:** are special variables that hold references to resources external to PHP (such as database connections).

The first five are *simple types*, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

Integers

They are whole numbers, without a decimal point, like 4195. They are the simplest type .they correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so:

```
$int_var = 12345;
$another_int = -12345 + 12345;
```

Boolean

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so:

```
if (TRUE)
    print("This will always print<br>");
else
    print("This will never print<br>");
```

Interpreting other types as Booleans

Here are the rules for determine the "truth" of any value not already of the Boolean type:

- If the value is a number, it is false if exactly equal to zero and true otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
- Don't use double as Booleans.

NULL

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this:

```
$my_var = NULL;
```

Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string:

```
<?
$variable = "name";
$literally = 'My $variable will not print!\\n'; print($literally);
$literally = "My $variable will print!\\n";
print($literally);
?>
```

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP:

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of their values.
- The escape-sequence replacements are:
- \n is replaced by the newline character

SAN JACINTO CATHOLIC SCHOOL, INC.

Computer Programming 7- 8 (ICT 12) – Evaluation for Module # 1

Name: _____ Strand: ICT Section: St. Gabriel Date:_____ Score: ____

(WRITTEN WORK: 20%)

Activity 1: Write True if the statement is correct and False if it is not. Write your answer on the space provide.

_____1. PHP Syntax is C-Like.

_____2. All PHP code must be included inside one of the three special markup tags ate are recognized by the PHP Parser.

_____3. In order to develop and run PHP Web pages, three vital components need to be installed on your computer system.

_____4. All variables in PHP are denoted with a leading dollar sign ($).

_____5. Variables are assigned with the = operator.

Activity 2: Identification. Identify the following sentences. Write your answer on the space provided.

_____1. Replaced by the newline character.

_____2. Who invented PHP?

_____3. Year where first version of PHP.

_____4. PHP stand for _____.

_____5. Short or short-open tags look like this: _____

_____6. It is the portion of a program that exists only for the human reader and stripped out before displaying the programs result.

_____7. They are generally used for short explanations or notes relevant to the local code.

_____8. They are generally used to provide pseudocode algorithms and more detailed explanations when necessary.

_____9. It is the stuff you type that is typically invisible on the screen, including spaces, tabs, and carriage returns (end-of-line characters).

_____10. In PHP is any expression that is followed by a semicolon (;).

PERFORMANCE TASKS 60%

Activity 1: Hands-On Activity: Create the following programming below. Use the PHP programming language.

1. Print your full name.
2. Data types
3. String
4. Integer

	Excellent(10)	Good(8)	Satisfactory(5)	Needs Improvement(1)
Following Activity Directions	All directions were exceeded	You followed most directions	You followed some directions	None of the directions were followed
Use of Creativity	You used your own ideas and imagination	You used your own ideas most of the time	You used some imagination	You did not use your own ideas or imagination
Effort put into Activity	You took your time and worked hard on the activity	You worked hard for most of the time	You put a small effort into the activity	You rushed through and did not work hard

SAN JACINTO CATHOLIC SCHOOL, INC.

Computer Programming 7-8 (ICT 12) – Evaluation for Module # 2

Name: _____ Strand: ICT Section: St. Gabriel Date:_____ Score: ____

(WRITTEN WORK: 20%)

Activity 1: Write True if the statement is correct and False if it is not. Write your answer on the space provide.

_____1. Strings that are delimited by double quotes (as in *this*).

_____2. Certain character sequences beginning with backslash (/) are replaced with special characters.

_____3. Variable names (starting with &) are replaced with string representations of their values.

_____4. Use double as Booleans.

_____5. Values of type NULL are always true.

Activity 2: ESSAY: Answer the following question. (2 points each)

1. What is PHP - _____

_____

_____

2. What is the use of PHP - _____

_____

_____

3. How to insert a message into a PHP program - _____

_____

_____

Activity 2: Hands-On Activity: Create the following programming below.

Instruction: 1. Create a program regarding the use of PHP programming language.

	Excellent(10)	Good(8)	Satisfactory(5)	Needs Improvement(1)
Following Activity Directions	All directions were exceeded	You followed most directions	You followed some directions	None of the directions were followed
Use of Creativity	You used your own ideas and imagination	You used your own ideas most of the time	You used some imagination	You did not use your own ideas or imagination
Effort put into Activity	You took your time and worked hard on the activity	You worked hard for most of the time	You put a small effort into the activity	You rushed through and did not work hard