SAN JACINTO CATHOLIC SCHOOL, INC.

ComProg 7-12ICT

San Jacinto, Pangasinan

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**COMPUTER PROGRAMMING 7 (ICT 12)**

GRADE 12 LEARNER’S MODULE #3 & 4 – 3rd Quarter, Weeks 5 - 8

**CHAPTER 1: INTRODUCTION TO PHP**

**OBJECTIVES**: In this lesson, you will learn to:

* Create a variable program.
* Understand the use of variable.
* Understanding the different types of variables.
* Understanding the use of constant.
* Use of different operators.

**Lesson 5: Variable Naming**

**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:**

**Variable Naming**

Rules for naming a variable is:

* Variable names must begin with a letter or underscore character.
* A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , ( , ) . & , etc

There is no size limit for variables.

**PHP – Variables**

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types:

* Local variables
* Function parameters
* Global variables
* Static variables

**PHP Local Variables**

A variable declared in a function is considered local; that is, it can be referenced solely in that function. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function:

<?

$x = 4;

$x = 0;

print "\$x inside function is $x.";

}

assignx();

print "\$x outside of function is $x.";

?>

This will produce the following result.



$x inside function is 0.

$x outside of function is 4.

**PHP Function Parameters**

PHP Functions are covered in detail in PHP Function Chapter. In short, a function is a small unit of program which can take some input in the form of parameters and does some processing and may return a value.

Function parameters are declared after the function name and inside parentheses. They are declared much like a typical variable would be:

<?

* multiply a value by 10 and return it to the caller function multiply ($value) {

$value = $value \* 10; return $value;

}

$retval = multiply (10);

Print "Return value is $retval\n";

?>

This will produce the following result.



Return value is 100

**PHP Global Variables**

In contrast to local variables, a global variable can be accessed in any part of the program. However, in order to be modified, a global variable must be explicitly declared to be global in the function in which it is to be modified.

This is accomplished, conveniently enough, by placing the keyword **GLOBAL** in front of the variable that should be recognized as global. Placing this keyword in front of an already existing variable tells PHP to use the variable having that name. Consider an example:

<?

$somevar = 15;

function addit() {

GLOBAL $somevar;

$somevar++;

print "Somevar is $somevar";

}

addit();

?>

This will produce the following result.



Somevar is 16

**PHP Static Variables**

The final type of variable scoping that I discuss is known as static. In contrast to the variables declared as function parameters, which are destroyed on the function's exit, a static variable will not lose its value when the function exits and will still hold that value should the function be called again.

You can declare a variable to be static simply by placing the keyword STATIC in front of the variable name.

**Lesson 6: Constant**

A constant is a name or an identifier for a simple value. A constant value cannot change during the execution of the script. By default, a constant is case-sensitive. By convention, constant identifiers are always uppercase. A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. If you have defined a constant, it can never be changed or undefined.

To define a constant you have to use define() function and to retrieve the value of a constant, you have to simply specifying its name. Unlike with variables, you do not need to have a constant with a $. You can also use the function constant() to read a constant's value if you wish to obtain the constant's name dynamically.

**constant() function**

As indicated by the name, this function will return the value of the constant.

This is useful when you want to retrieve value of a constant, but you do not know its name, i.e., it is stored in a variable or returned by a function.

**constant() example**

<?php

define("MINSIZE", 50);

echo MINSIZE;

echo constant("MINSIZE"); // same thing as the previous line

?>

Only scalar data (boolean, integer, float and string) can be contained in constants.

**Differences between constants and variables are**

* There is no need to write a dollar sign ($) before a constant, where as in Variable one has to write a dollar sign.
* Constants cannot be defined by simple assignment, they may only be defined using the define() function.
* Constants may be defined and accessed anywhere without regard to variable scoping rules.
* Once the Constants have been set, may not be redefined or undefined.

**Valid and invalid constant names**

// Valid constant names

define("ONE", "first thing");

define("TWO2", "second thing");

define("THREE\_3", "third thing")

// Invalid constant names

define("2TWO", "second thing");

define("\_\_THREE\_\_", "third value");

**PHP Magic constants**

PHP provides a large number of predefined constants to any script which it runs.

There are five magical constants that change depending on where they are used. For example, the value of \_\_LINE\_\_ depends on the line that it's used on in your script. These special constants are case-insensitive and are as follows:

The following table lists a few "magical" PHP constants along with their description:

|  |  |
| --- | --- |
| **Name** | **Description** |
|  |  |
| \_\_LINE\_\_ | The current line number of the file. |
|  |  |
| \_\_FILE\_\_ | The full path and filename of the file. If used inside an include, the |
|  | name of the included file is returned. Since PHP |
|  | 4.0.2, \_\_FILE\_\_ always contains an absolute path whereas in older |
|  | versions it contained relative path under some circumstances. |
|  |  |
| \_FUNCTION\_ | The function name. (Added in PHP 4.3.0) As of PHP 5 this constant |
|  | returns the function name as it was declared (case-sensitive). In PHP |
|  | 4 its value is always lowercased. |
|  |  |
| \_\_CLASS\_\_ | The class name. (Added in PHP 4.3.0) As of PHP 5 this constant |
|  | returns the class name as it was declared (case-sensitive). In PHP 4 |
|  | its value is always lowercased. |
|  |  |
| \_\_METHOD\_\_ | The class method name. (Added in PHP 5.0.0) The method name is |
|  | returned as it was declared (case-sensitive). |
|  |  |

**Lesson 6: Operators**

**What is Operator?** Simple answer can be given using expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator. PHP language supports following type of operators.

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Assignment Operators
* Conditional (or ternary) Operators Let’s have a look on all operators one by one.

**Arithmetic Operators**

The following arithmetic operators are supported by PHP language:

Assume variable A holds 10 and variable B holds 20 then:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
|  |  |  |
| + | Adds two operands | A + B will give 30 |
|  |  |  |
| - | Subtracts second operand from the first | A - B will give -10 |
|  |  |  |
| \* | Multiply both operands | A \* B will give 200 |
|  |  |  |
| / | Divide the numerator by denominator | B / A will give 2 |
|  |  |  |
| % | Modulus Operator and remainder of after an | B % A will give 0 |
|  | integer division |  |
|  |  |  |
| ++ | Increment operator, increases integer value by | A++ will give 11 |
|  | one |  |
|  |  |  |
| -- | Decrement operator, decreases integer value | A-- will give 9 |
|  | by one |  |
|  |  |  |

**Example**

Try the following example to understand all the arithmetic operators. Copy and paste following PHP program in test.php file and keep it in your PHP Server's document root and browse it using any browser.

<html>

<head><title>Arithmetical Operators</title><head>

<body>

<?php

$a = 42;

$b = 20;

$c = $a + $b;

echo "Addition Operation Result: $c <br/>"; $c = $a - $b;

echo "Subtraction Operation Result: $c <br/>"; $c = $a \* $b;

echo "Multiplication Operation Result: $c <br/>"; $c = $a / $b;

echo "Division Operation Result: $c <br/>"; $c = $a % $b;

echo "Modulus Operation Result: $c <br/>";

$c = $a++;

echo "Increment Operation Result: $c <br/>"; $c = $a--;

echo "Decrement Operation Result: $c <br/>";

?>

</body>

</html>

This will produce the following result:

Addition Operation Result: 62

Subtraction Operation Result: 22

Multiplication Operation Result: 840

Division Operation Result: 2.1

Modulus Operation Result: 2

Increment Operation Result: 42

Decrement Operation Result: 43

Prepared by: Mr. Christian U. de Guzman Checked by: Mrs. Lailanie L. De Guzman, M.E.d

Subject Teacher Gr.12 Coordinator

SAN JACINTO CATHOLIC SCHOOL, INC.

**Computer Programming 7(ICT 12) – Evaluation for Module # 3**

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. There are five magical constants that change depending on where they are used.

\_\_\_\_\_\_\_\_\_\_2. Variable names must begin with a letter or underscore character.

\_\_\_\_\_\_\_\_\_\_3. A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , ( , ) . & , etc.

\_\_\_\_\_\_\_\_\_\_4. There is no size limit for variables.

\_\_\_\_\_\_\_\_\_\_5. Scope can be defined as the range of availability a variable has to the program in which it is declared.

\_\_\_\_\_\_\_\_\_\_6. A variable declared in a function is considered local; that is, it can be referenced solely in that function.

\_\_\_\_\_\_\_\_\_\_7. Any assignment outside of that function will be considered to be an entirely different variable from the one contained in the function.

\_\_\_\_\_\_\_\_\_\_8. PHP Functions are covered in detail in PHP Function Chapter.

\_\_\_\_\_\_\_\_\_\_9. A small unit of program which can take some input in the form of parameters and does some processing and may return a value.

\_\_\_\_\_\_\_\_\_\_10. In contrast to local variables, a global variable can be accessed in any part of the program.

**Activity 2:** ESSAY: How to run a PHP program? 15 points.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PERFORMANCE TASKS 60%**

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

1. Constant.

2. Method/Post

3. Variable

SAN JACINTO CATHOLIC SCHOOL, INC.

**Computer Programming 7 (ICT 12) – Evaluation for Module # 4**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Strand: ICT Section: St. Gabriel Date:\_\_\_\_\_\_\_ Score: \_\_\_\_

**(WRITTEN WORK: 20%)**

**Activity 1:** Read the following statements. Write **True** if the statement is correct, and **false** if it is not on the space provided.

\_\_\_\_\_\_\_\_\_\_1. The final type of variable scoping that I discuss is known as static

\_\_\_\_\_\_\_\_\_\_2. Only scalar data (boolean, integer, float and string) can be contained in constants.

\_\_\_\_\_\_\_\_\_\_3. To define a constant you have to use define() function and to retrieve the value of a constant.

\_\_\_\_\_\_\_\_\_\_4. Once the Constants have been set, may not be redefined or undefined.

\_\_\_\_\_\_\_\_\_\_5. A constant is a name or an identifier for a simple value.

\_\_\_\_\_\_\_\_\_\_6. Constants cannot be defined by simple assignment, they may only be defined using the define() function.

\_\_\_\_\_\_\_\_\_\_7. Constants may be defined and accessed anywhere without regard to variable scoping rules.

\_\_\_\_\_\_\_\_\_\_8. As indicated by the name, this function will return the value of the constant.

\_\_\_\_\_\_\_\_\_\_9. There is no need to write a dollar sign ($) before a constant, where as in Variable one has to write a dollar sign.

\_\_\_\_\_\_\_\_\_\_10. PHP provides a large number of predefined constants to any script which it runs.

**Activity 2**: Identify what is being asked for. Write your answer on the space provided.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1. Adds two operands

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2. Modulus Operator and remainder of after an integer division.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3. Increment operator, increases integer value by one.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4. Avoid saving your file.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5. Decrement operator, decreases integer value by one.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6. PHP can run the program without installing it.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7. Subtracts second operand from the first.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8. Divide the numerator by denominator

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_9. By pressing F5 your program will start to refresh the page.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_10. Multiply both operands.

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

Instruction: 1. Create a program on how to use mathematical operators by using variable.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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 |