DEPARTMENT OF COMPUTER STUDIES

**(Applications Development and Emerging Technologies)**

**PRE-SUMMATIVE ASSESSMENT**

**3**

**PHP ARRAYS AND FUNCTIONS**

**Student Name / Group**

**Name: Patrick Andrew Tria**

**Name** **Role**

**Members (if Group):**

**Section: 1A TN31**

**Professor: Mr. Abraham Magpantay**

1. **PROGRAM OUTCOME/S (PO) ADDRESSED BY THE LABORATORY EXERCISE**
   * Design, implement and evaluate computer-based systems or applications to meet desired needs and requirements.

**II. COURSE LEARNING OUTCOME/S (CLO) ADDRESSED BY THE LABORATORY EXERCISE**

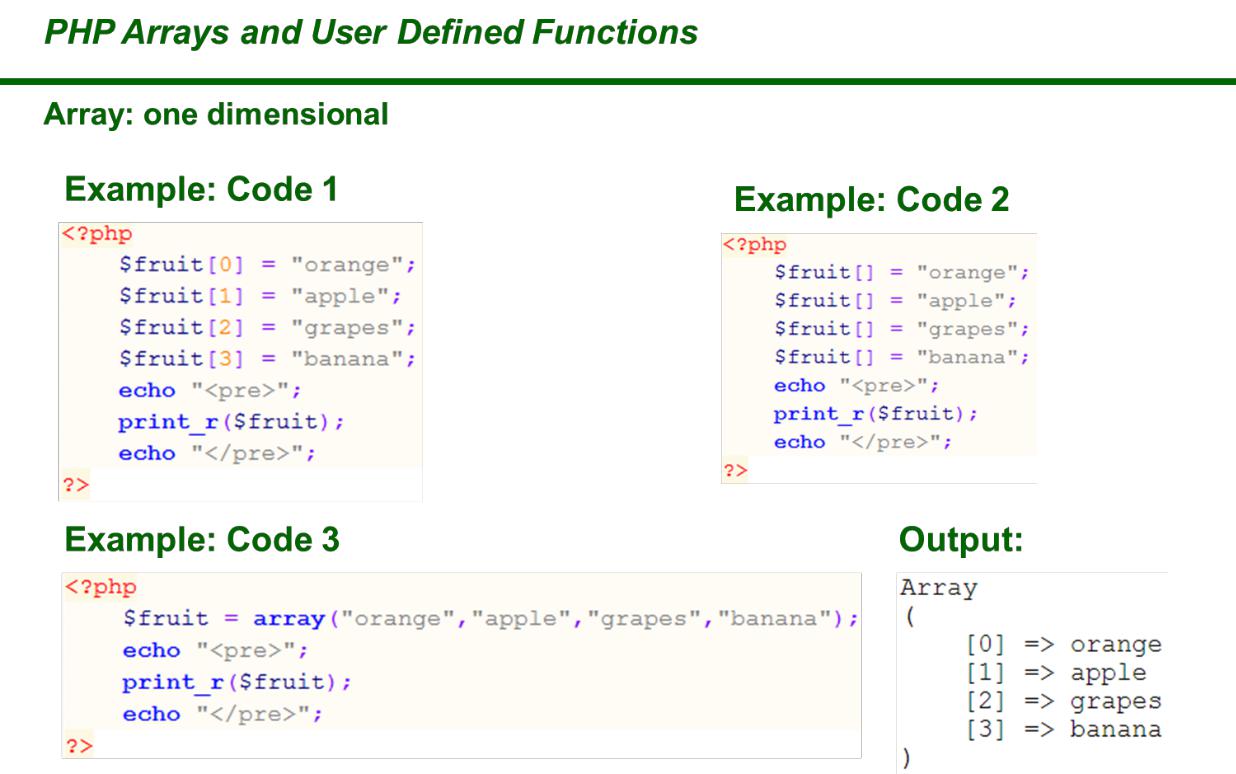
* + Understand and apply best practices and standards in the development of website.

**III. INTENDED LEARNING OUTCOME/S (ILO) OF THE LABORATORY EXERCISE**

At the end of this exercise, students must be able to:

* + - To know the different approach of using arrays in PHP.
    - To use PHP lazy function foreach to iterate through array elements.
    - To implement one dimensional and multi-dimensional in the program.
    - To know use variables that is globally declared and locally declared.

**IV. BACKGROUND INFORMATION**



**Applications Development and Emerging Technologies** **Page 2 of 19**

**PHP Arrays**

An array stores multiple values in one single variable:

**Example**

<?php

$cars = array(“Volvo”, “BMW”, “Toyota”);

echo “I like “ . $cars[0] . “, “ . $cars[1] . “ and “ . $cars[2] . “.”; ?>

**What is an Array?**

An array is a special variable, which can hold more than one value at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

$cars1 = “Volvo”;

$cars2 = “BMW”;

$cars3 = “Toyota”;

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is to create an array!

An array can hold many values under a single name, and you can access the values by referring to an index number

**Create an Array in PHP**

In PHP, the array() function is used to create an array:

array();

In PHP, there are three types of arrays:

* **Indexed arrays** – Arrays with a numeric index
* **Associative arrays** – Arrays with named keys



**Applications Development and Emerging Technologies** **Page 3 of 19**

* **Multidimensional arrays** – Arrays containing one or more arrays

**Get The Length of an Array – The count() Function**

The count() function is used to return the length (the number of elements) of an array:

**Example**

<?php

$cars = array(“Volvo”, “BMW”, “Toyota”); echo count($cars);

?>

**PHP Indexed Arrays**

There are two ways to create indexed arrays:

The index can be assigned automatically (index always starts at 0), like this:

$cars = array(“Volvo”, “BMW”, “Toyota”);

or the index can be assigned manually:

$cars[0] = “Volvo”;

$cars[1] = “BMW”;

$cars[2] = “Toyota”;

The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:

**Example**

<?php

$cars = array(“Volvo”, “BMW”, “Toyota”);

echo “I like “ . $cars[0] . “, “ . $cars[1] . “ and “ . $cars[2] . “.”; ?>



**Applications Development and Emerging Technologies** **Page 4 of 19**

Loop Through an Indexed Array

To loop through and print all the values of an indexed array, you could use a for loop, like this:

**Example**

<?php

$cars = array(“Volvo”, “BMW”, “Toyota”); $arrlength = count($cars);

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

echo $cars[$x];

echo “<br>”;

}

?>

**PHP Associative Arrays**

Associative arrays are arrays that use named keys that you assign to them.

There are two ways to create an associative array:

$age = array(“Peter”=>”35”, “Ben”=>”37”, “Joe”=>”43”);

or:

$age[‘Peter’] = “35”;

$age[‘Ben’] = “37”;

$age[‘Joe’] = “43”;

The named keys can then be used in a script:

**Example**

<?php

$age = array(“Peter”=>”35”, “Ben”=>”37”, “Joe”=>”43”); echo “Peter is “ . $age[‘Peter’] . “ years old.”;

?>



**Applications Development and Emerging Technologies** **Page 5 of 19**

Loop Through an Associative Array

To loop through and print all the values of an associative array, you could use a foreach loop, like this:

**Example**

<?php

$age = array(“Peter”=>”35”, “Ben”=>”37”, “Joe”=>”43”);

foreach($age as $x => $x\_value) {

echo “Key=” . $x . “, Value=” . $x\_value; echo “<br>”;

}

?>

**PHP Multidimensional Arrays**

A multidimensional array is an array containing one or more arrays.

PHP supports multidimensional arrays that are two, three, four, five, or more levels deep. However, arrays more than three levels deep are hard to manage for most people.

**The dimension of an array indicates the number of indices you need to select an element.**

* For a two-dimensional array you need two indices to select an element
* For a three-dimensional array you need three indices to select an element

PHP – Two-dimensional Arrays

A two-dimensional array is an array of arrays (a three-dimensional array is an array of arrays of arrays).

First, take a look at the following table:



**Applications Development and Emerging Technologies** **Page 6 of 19**

|  |  |  |
| --- | --- | --- |
| **Name** | **Stock** | **Sold** |
|  |  |  |
| Volvo | 22 | 18 |
|  |  |  |
| BMW | 15 | 13 |
|  |  |  |
| Saab | 5 | 2 |
|  |  |  |
| Land Rover | 17 | 15 |
|  |  |  |

We can store the data from the table above in a two-dimensional array, like this:

$cars = array (

array(“Volvo”,22,18),

array(“BMW”,15,13),

array(“Saab”,5,2),

array(“Land Rover”,17,15)

);

Now the two-dimensional $cars array contains four arrays, and it has two indices: row and column.

To get access to the elements of the $cars array we must point to the two indices (row and column):

**Example**

<?php

echo $cars[0][0].”: In stock: “.$cars[0][1].”, sold:

“.$cars[0][2].”.<br>”;

echo $cars[1][0].”: In stock: “.$cars[1][1].”, sold:

“.$cars[1][2].”.<br>”;

echo $cars[2][0].”: In stock: “.$cars[2][1].”, sold:

“.$cars[2][2].”.<br>”;

echo $cars[3][0].”: In stock: “.$cars[3][1].”, sold:

“.$cars[3][2].”.<br>”;

?>

We can also put a for loop inside another for loop to get the elements of the $cars array (we still have to point to the two indices):



**Applications Development and Emerging Technologies** **Page 7 of 19**

**Example**

<?php

for ($row = 0; $row < 4; $row++) { echo “<p><b>Row number $row</b></p>”; echo “<ul>”;

for ($col = 0; $col < 3; $col++) {

echo “<li>”.$cars[$row][$col].”</li>”;

}

echo “</ul>”;

}

?>

**PHP Sorting Arrays**

The elements in an array can be sorted in alphabetical or numerical order, descending or ascending.

In this chapter, we will go through the following PHP array sort functions:

* sort() – sort arrays in ascending order
* rsort() – sort arrays in descending order
* ssort() – sort associative arrays in ascending order, according to the value
* ksort() – sort associative arrays in ascending order, according to the key
* arsort() – sort associative arrays in descending order, according to the value
* krsort() – sort associative arrays in descending order, according to the key

Sort Array in Ascending Order – sort()

The following example sorts the elements of the $cars array in ascending alphabetical order:

**Example**

<?php

$cars = array(“Volvo”, “BMW”, “Toyota”);



**Applications Development and Emerging Technologies** **Page 8 of 19**

sort($cars);

?>

The following example sorts the elements of the $numbers array in ascending numerical order:

**Example**

<?php

$numbers = array(4, 6, 2, 22, 11);

sort($numbers);

?>

Sort Array in Descending Order – rsort()

The following example sorts the elements of the $cars array in descending alphabetical order:

**Example**

<?php

$cars = array(“Volvo”, “BMW”, “Toyota”); rsort($cars);

?>

The following example sorts the elements of the $numbers array in descending numerical order:

**Example**

<?php

$numbers = array(4, 6, 2, 22, 11);

rsort($numbers);

?>

Sort Array (Ascending Order), According to Value – ssort()



**Applications Development and Emerging Technologies** **Page 9 of 19**

The following example sorts an associative array in ascending order, according to the value:

**Example**

<?php

$age = array(“Peter”=>”35”, “Ben”=>”37”, “Joe”=>”43”); ssort($age);

?>

Sort Array (Ascending Order), According to Key – ksort()

The following example sorts an associative array in ascending order, according to the key:

**Example**

<?php

$age = array(“Peter”=>”35”, “Ben”=>”37”, “Joe”=>”43”); ksort($age);

?>

Sort Array (Descending Order), According to Value – arsort()

The following example sorts an associative array in descending order, according to the value:

**Example**

<?php

$age = array(“Peter”=>”35”, “Ben”=>”37”, “Joe”=>”43”); arsort($age);

?>



**Applications Development and Emerging Technologies** **Page 10 of 19**

Sort Array (Descending Order), According to Key – krsort()

The following example sorts an associative array in descending order, according to the key:

**Example**

<?php

$age = array(“Peter”=>”35”, “Ben”=>”37”, “Joe”=>”43”); krsort($age);

?>

**PHP Functions**

The real power of PHP comes from its functions.

PHP has more than 1000 built-in functions, and in addition you can create your own custom functions.

PHP has over 1000 built-in functions that can be called directly, from within a script, to perform a specific task.

Please check out our PHP reference for a complete overview of the [PHP built-in](https://www.w3schools.com/php/php_ref_overview.asp) [functions](https://www.w3schools.com/php/php_ref_overview.asp).

PHP User Defined Functions

Besides the built-in PHP functions, it is possible to create your own functions.

* A function is a block of statements that can be used repeatedly in a program.
* A function will not execute automatically when a page loads.
* A function will be executed by a call to the function.



**Applications Development and Emerging Technologies** **Page 11 of 19**

Create a User Defined Function in PHP

A user-defined function declaration starts with the word function:

**Syntax**

function *functionName*() {

*code to be executed*;

}

**Note:** A function name must start with a letter or an underscore. Functionnames are NOT case-sensitive.

**Tip:** Give the function a name that reflects what the function does!

In the example below, we create a function named “writeMsg()”. The opening curly brace ( { ) indicates the beginning of the function code, and the closing curly brace ( } ) indicates the end of the function. The function outputs “Hello world!”. To call the function, just write its name followed by brackets ():

**Example**

<?php

function writeMsg() {

echo “Hello world!”;

}

writeMsg(); // call the function

?>

PHP Function Arguments

Information can be passed to functions through arguments. An argument is just like a variable.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

The following example has a function with one argument ($fname). When the familyName() function is called, we also pass along a name (e.g. Jani), and the



**Applications Development and Emerging Technologies** **Page 12 of 19**

name is used inside the function, which outputs several different first names, but an equal last name:

**Example**

<?php

function familyName($fname) {

echo “$fname Refsnes.<br>”;

}

familyName(“Jani”);

familyName(“Hege”);

familyName(“Stale”);

familyName(“Kai Jim”);

familyName(“Borge”);

?>

The following example has a function with two arguments ($fname and $year):

**Example**

<?php

function familyName($fname, $year) {

echo “$fname Refsnes. Born in $year <br>”;

}

familyName(“Hege”, “1975”);

familyName(“Stale”, “1978”);

familyName(“Kai Jim”, “1983”);

?>

PHP is a Loosely Typed Language

In the example above, notice that we did not have to tell PHP which data type the variable is.

PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.

In PHP 7, type declarations were added. This gives us an option to specify the expected data type when declaring a function, and by adding

the strict declaration, it will throw a “Fatal Error” if the data type mismatches.

**Applications Development and Emerging Technologies** **Page 13 of 19**

In the following example we try to send both a number and a string to the function without using strict:

**Example**

<?php

function addNumbers(int $a, int $b) {

return $a + $b;

}

echo addNumbers(5, “5 days”);

* since strict is NOT enabled “5 days” is changed to int(5), and it will return 10

?>

To specify strict we need to set declare(strict\_types=1);. This must be on the very first line of the PHP file.

In the following example we try to send both a number and a string to the function, but here we have added the strict declaration:

**Example**

<?php declare(strict\_types=1); // strict requirement

function addNumbers(int $a, int $b) {

return $a + $b;

}

echo addNumbers(5, “5 days”);

* since strict is enabled and “5 days” is not an integer, an error will be thrown

?>

The strict declaration forces things to be used in the intended way.

PHP Default Argument Value

The following example shows how to use a default parameter. If we call the function setHeight() without arguments it takes the default value as argument:

**Example**

<?php declare(strict\_types=1); // strict requirement function setHeight(int $minheight = 50) {



**Applications Development and Emerging Technologies** **Page 14 of 19**

echo “The height is : $minheight <br>”;

}

setHeight(350);

setHeight(); // will use the default value of 50

setHeight(135);

setHeight(80);

?>

PHP Functions – Returning values

To let a function return a value, use the return statement:

**Example**

<?php declare(strict\_types=1); // strict requirement function sum(int $x, int $y) {

$z = $x + $y;

return $z;

}

echo “5 + 10 = “ . sum(5, 10) . “<br>”; echo “7 + 13 = “ . sum(7, 13) . “<br>”; echo “2 + 4 = “ . sum(2, 4);

?

PHP Return Type Declarations

PHP 7 also supports Type Declarations for the return statement. Like with the type declaration for function arguments, by enabling the strict requirement, it will throw a “Fatal Error” on a type mismatch.

To declare a type for the function return, add a colon ( : ) and the type right before the opening curly ( { )bracket when declaring the function.

In the following example we specify the return type for the function:

**Example**

<?php declare(strict\_types=1); // strict requirement function addNumbers(float $a, float $b) : float {

return $a + $b;



**Applications Development and Emerging Technologies** **Page 15 of 19**

}

echo addNumbers(1.2, 5.2);

?>

You can specify a different return type, than the argument types, but make sure the return is the correct type:

**Example**

<?php declare(strict\_types=1); // strict requirement function addNumbers(float $a, float $b) : int {

return (int)($a + $b);

}

echo addNumbers(1.2, 5.2);

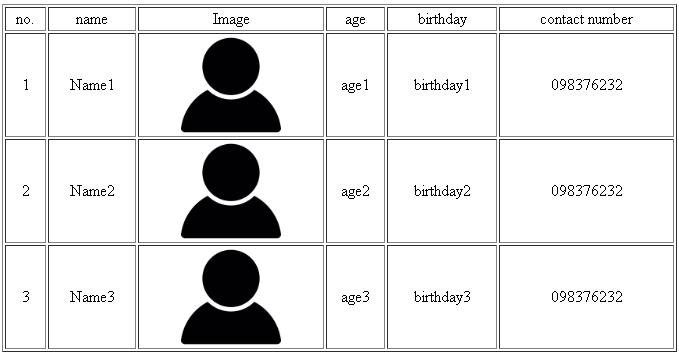
?>

1. **GRADING SYSTEM / RUBRIC (please see separate sheet)**

**VI. LABORATORY ACTIVITY**

* 1. **USING ARRAYS**, create a program using PHP that will contain 10 differentnames, images, age, birthday and contact number then display it in alphabetical order (use array sorting), with integration of HTML and CSS.

**Sample Output:**



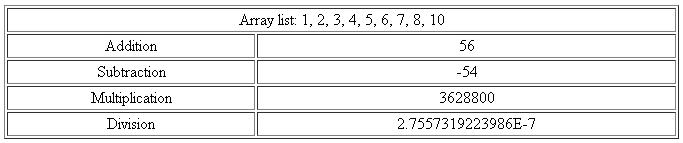
**Applications Development and Emerging Technologies** **Page 16 of 19**

1. **Using ARRAYS**, create a program using PHP that will contain 10 differentnumbers and get the sum, difference, product and quotient of all values in the array.

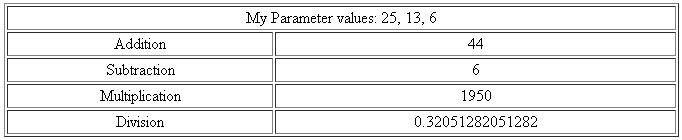
**Sample Output:**

1. **USING USER DEFINED FUNCTION**
   * Create a program that will get the sum, difference, product and quotient of 3 given parameters.

Example your\_function(param1,param2,param3)



**Sample Output:**



***Snip and paste your source codes here. Snip it directly from the IDE so that colors of the codes are preserved for readability. Include additional pages if necessary.***

**VII. QUESTION AND ANSWER**

1. **For your understanding what is an array?**

Array allows collection of elements (values or variables) to be grouped and identified through array index, keys or etc. Arrays allows the application of operations on an entire set of values at once instead of doing it individually.

1. **Differences between simple variable and a variable array?**

A simple variable can only contain a sing value, with variable array, it can contain more than one value at a time under the same variable.

1. **Give an instance where you can use or apply an array.**

One instance where you can apply array is in library systems. Each array would store its own sets of values like product number, genre, title, date published, status, and etc. Therefore, an array would be equivalent to a book with its subcategories.

1. **What are the different array sorting? describe each.**

The different array sorting are, sort(), rsort(), ssort(), ksort(), arsort(), krsort(). The sort() sorts array in ascending while rsort() sorts it by descending order. The ssort() sorts associative arrays in ascending order depending on the value while arsort() sorts it in descending. The ksort() sorts associative arrays via ascending order depending on the key and krsort() for descending order.

1. **What is Function?**

Function is a block of organized, reusable code that is used to perform a specific action.They are self-contained modules of codes that can accomplish specific task and usually take in data, process and returns the value once called.

1. **Give the different usage of functions.**

The usage of a function allows modularity and repeatability of codes, it shortens and clean out your coding making the readability and efficiency of the program faster. An example where functions are best used would be recursions.

**VIII. REFERENCES**

1. <https://www.w3schools.com/css/>
2. <https://www.w3schools.com/html/>



1. <https://www.w3schools.com/php/php_variables.asp>
2. <https://www.w3schools.com/php/php_arrays.asp>
3. <https://www.w3schools.com/php/php_arrays_indexed.asp>
4. <https://www.w3schools.com/php/php_arrays_associative.asp>
5. <https://www.w3schools.com/php/php_arrays_multidimensional.asp>
6. <https://www.w3schools.com/php/php_arrays_sort.asp>
7. <https://www.w3schools.com/php/php_functions.asp>

**Note: The following rubrics/metrics will be used to grade students’ output.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Program (100** | **(Excellent)** | **(Good)** | **(Fair)** | **(Poor)** |
| **pts.)** |  |  |  |  |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Program** | Program executes | Program executes | Program executes | Program does not |
| **execution (20pts)** | correctly with no | with less than 3 | with more than 3 | execute **(10-** |
|  | syntax or runtime | errors **(15-17pts)** | errors **(12-14pts)** | **11pts)** |
|  | errors **(18-20pts)** |  |  |  |
| **Correct output** | Program displays | Output has minor | Output has | Output is incorrect |
| **(20pts)** | correct output | errors **(15-17pts)** | multiple errors | **(10-11pts)** |
|  | with no errors |  | **(12-14pts)** |  |
|  | **(18-20pts)** |  |  |  |
| **Design of output** | Program displays | Program displays | Program does not | Output is poorly |
| **(10pts)** | more than | minimally | display the | designed **(5pts)** |
|  | expected **(10pts)** | expected output | required output |  |
|  |  | **(8-9pts)** | (**6-7pts)** |  |
| **Design of logic** | Program is | Program has | Program has | Program is |
| **(20pts)** | logically well | slight logic errors | significant logic | incorrect **(10-** |
|  | designed **(18-** | that do no | errors **(3-5pts)** | **11pts)** |
|  | **20pts)** | significantly |  |  |
|  |  | affect the results |  |  |
|  |  | **(15-17pts)** |  |  |
| **Standards** | Program code is | Few inappropriate | Several | Program is poorly |
| **(20pts)** | stylistically well | design choices | inappropriate | written **(10-11pts)** |
|  | designed **(18-** | (i.e. poor variable | design choices |  |
|  | **20pts)** | names, improper | (i.e. poor variable |  |
|  |  | indentation) **(15-** | names, improper |  |
|  |  | **17pts)** | indentation) **(12-** |  |
|  |  |  | **14pts)** |  |
| **Delivery** | The program was | The program was | The program was | The program was |
| **(10pts)** | delivered on time. | delivered a day | delivered two | delivered more |
|  | **(10pts)** | after the deadline. | days after the | than two days |
|  |  | **(8-9pts)** | deadline. **(6-7pts)** | after the deadline. |
|  |  |  |  | **(5pts)** |



**Applications Development and Emerging Technologies** **Page 19 of 19**