Experiment – 7 – BASIC PHP

Roll No.: K041	**Name:** Anish Sudhan Nair
Prog/Yr/Sem: B.Tech. Cybersecurity	**Batch:** K2/A2
Date of Experiment: 18/02/2022	**Date of Submission:** 25/02/2022

Aim:

A. Apply various styling techniques Using External CSS.
B. Frames & iFrame
C. Transform property

Prerequisites:-

- Basic Tags of HTML , CSS and Java Script

Theory:

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

PHP is an amazing and popular language!

It is powerful enough to be at the core of the biggest blogging system on the web (WordPress)!
It is deep enough to run large social networks!
It is also easy enough to be a beginner's first server side language!

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data

- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

With PHP you are not limited to output HTML. You can output images, PDF files, and even Flash movies. You can also output any text, such as XHTML and XML.

Why PHP?

- PHP runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- PHP is compatible with almost all servers used today (Apache, IIS, etc.)
- PHP supports a wide range of databases
- PHP is free. Download it from the official PHP resource: www.php.net
- PHP is easy to learn and runs efficiently on the server side

What's new in PHP 7

- PHP 7 is much faster than the previous popular stable release (PHP 5.6)
- PHP 7 has improved Error Handling
- PHP 7 supports stricter Type Declarations for function arguments
- PHP 7 supports new operators (like the spaceship operator: <=>)

What Do I Need?

To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host With PHP Support

If your server has activated support for PHP you do not need to do anything.

Just create some .php files, place them in your web directory, and the server will automatically parse them for you.

You do not need to compile anything or install any extra tools.

Because PHP is free, most web hosts offer PHP support.

Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

- install a web server
- install PHP
- install a database, such as MySQL

- The official PHP website (PHP.net) has installation instructions for PHP: http://php.net/manual/en/install.php

Basic PHP Syntax

A PHP script can be placed anywhere in the document.

A PHP script starts with <?php and ends with ?>:

```
<?php
// PHP code goes here
?>
```

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

Below, we have an example of a simple PHP file, with a PHP script that uses a built-in PHP function "echo" to output the text "Hello World!" on a web page:

Example

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

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
```

```
?>

</body>
</html>
```

PHP Case Sensitivity

In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive.

In the example below, all three echo statements below are equal and legal:

Example

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

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

Note: However; all variable names are case-sensitive!

Look at the example below; only the first statement will display the value of the $color variable! This is because $color, $COLOR, and $coLOR are treated as three different variables:

Example

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

<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

Comments in PHP

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

Comments can be used to:

- Let others understand your code
- Remind yourself of what you did - Most programmers have experienced coming back to their own work a year or two later and having to re-figure out what they did. Comments can remind you of what you were thinking when you wrote the code

PHP supports several ways of commenting:

Example

Syntax for single-line comments:

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

<?php
// This is a single-line comment

# This is also a single-line comment
?>

</body>
</html>
```

Example

Syntax for multiple-line comments:

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

```
<?php
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
?>

</body>
</html>
```

PHP Variables

Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

Example

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

PHP Variables

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

Remember that PHP variable names are case-sensitive!

Output Variables

The PHP echo statement is often used to output data to the screen.

The following example will show how to output text and a variable:

Example

```php
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

PHP echo and print Statements

❮ PreviousNext ❯

PHP echo and print Statements

echo and print are more or less the same. They are both used to output data to the screen.

The differences are small: echo has no return value while print has a return value of 1 so it can be used in expressions. echo can take multiple parameters (although such usage is rare) while print can take one argument. echo is marginally faster than print.

The PHP echo Statement

The echo statement can be used with or without parentheses: echo or echo().

Display Text

The following example shows how to output text with the echo command (notice that the text can contain HTML markup):

Example

```php
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

Try it Yourself »

Display Variables

The following example shows how to output text and variables with the echo statement:

Example

```php
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
```

The PHP print Statement

The print statement can be used with or without parentheses: print or print().

Display Text

The following example shows how to output text with the print command (notice that the text can contain HTML markup):

Example

```php
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

Try it Yourself »

Display Variables

The following example shows how to output text and variables with the print statement:

Example

```php
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;
```

```
print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x + $y;
?>
```

PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

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

PHP String

A string is a sequence of characters, like "Hello world!".

A string can be any text inside quotes. You can use single or double quotes:

Example

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

Try it Yourself »

PHP Integer

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$x = 5985;
var_dump($x);
?>
```

Try it Yourself »

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$x = 10.365;
var_dump($x);
?>
```

Try it Yourself »

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

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

Booleans are often used in conditional testing. You will learn more about conditional testing in a later chapter of this tutorial.

PHP Array

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

Example

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

Try it Yourself »

You will learn a lot more about arrays in later chapters of this tutorial.

PHP Object

Classes and objects are the two main aspects of object-oriented programming.

A class is a template for objects, and an object is an instance of a class.

When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

Let's assume we have a class named Car. A Car can have properties like model, color, etc. We can define variables like $model, $color, and so on, to hold the values of these properties.

When the individual objects (Volvo, BMW, Toyota, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

If you create a __construct() function, PHP will automatically call this function when you create an object from a class.

Example

```
<?php
class Car {
  public $color;
  public $model;
  public function __construct($color, $model) {
    $this->color = $color;
    $this->model = $model;
  }
  public function message() {
    return "My car is a " . $this->color . " " . $this->model . "!";
  }
}

$myCar = new Car("black", "Volvo");
echo $myCar -> message();
echo "<br>";
$myCar = new Car("red", "Toyota");
echo $myCar -> message();
?>
```

Try it Yourself »

PHP NULL Value

Null is a special data type which can have only one value: NULL.

A variable of data type NULL is a variable that has no value assigned to it.

Tip: If a variable is created without a value, it is automatically assigned a value of NULL.

Variables can also be emptied by setting the value to NULL:

Example

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

Try it Yourself »

PHP Strings

PHP String Functions

In this chapter we will look at some commonly used functions to manipulate strings.

strlen() - Return the Length of a String

The PHP strlen() function returns the length of a string.

Example

Return the length of the string "Hello world!":

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

Try it Yourself »

str_word_count() - Count Words in a String

The PHP str_word_count() function counts the number of words in a string.

Example

Count the number of word in the string "Hello world!":

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

Try it Yourself »

strrev() - Reverse a String

The PHP strrev() function reverses a string.

Example

Reverse the string "Hello world!":

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

Try it Yourself »

strpos() - Search For a Text Within a String

The PHP strpos() function searches for a specific text within a string. If a match is found, the function returns the character position of the first match. If no match is found, it will return FALSE.

Example

Search for the text "world" in the string "Hello world!":

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

Try it Yourself »

Tip: The first character position in a string is 0 (not 1).

str_replace() - Replace Text Within a String

The PHP str_replace() function replaces some characters with some other characters in a string.

Example

Replace the text "world" with "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
?>
```

Try it Yourself »

PHP pi() Function

The pi() function returns the value of PI:

Example

```
<?php
echo(pi()); // returns 3.1415926535898
?>
```

Try it Yourself »

PHP min() and max() Functions

The min() and max() functions can be used to find the lowest or highest value in a list of arguments:

Example

```
<?php
echo(min(0, 150, 30, 20, -8, -200));  // returns -200
echo(max(0, 150, 30, 20, -8, -200));  // returns 150
?>
```

Try it Yourself »

PHP abs() Function

The abs() function returns the absolute (positive) value of a number:

Example

```
<?php
echo(abs(-6.7));  // returns 6.7
?>
```

Try it Yourself »

PHP sqrt() Function

The sqrt() function returns the square root of a number:

Example

```
<?php
echo(sqrt(64));  // returns 8
?>
```

Try it Yourself »

PHP round() Function

The round() function rounds a floating-point number to its nearest integer:

Example

```
<?php
echo(round(0.60));  // returns 1
echo(round(0.49));  // returns 0
?>
```

Try it Yourself »

Random Numbers

The rand() function generates a random number:

Example

```
<?php
echo(rand());
?>
```

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

Operator	**Name**	**Example**	**Result**	**Show it**
+	Addition	$x + $y	Sum of $x and $y	Try it »
-	Subtraction	$x - $y	Difference of $x and $y	Try it »
*	Multiplication	$x * $y	Product of $x and $y	Try it »
/	Division	$x / $y	Quotient of $x and $y	Try it »
%	Modulus	$x % $y	Remainder of $x divided by $y	Try it »
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power	Try it »

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	**Same as...**	**Description**	**Show it**
x = y	x = y	The left operand gets set to the value of the expression on the right	Try it »
x += y	x = x + y	Addition	Try it »
x -= y	x = x - y	Subtraction	Try it »
x *= y	x = x * y	Multiplication	Try it »
x /= y	x = x / y	Division	Try it »
x %= y	x = x % y	Modulus	Try it »

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	**Name**	**Example**	**Result**	**Show it**
==	Equal	$x == $y	Returns true if $x is equal to $y	Try it »
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type	Try it »
!=	Not equal	$x != $y	Returns true if $x is not equal to $y	Try it »
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y	Try it »
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type	Try it »
>	Greater than	$x > $y	Returns true if $x is greater than $y	Try it »
<	Less than	$x < $y	Returns true if $x is less than $y	Try it »
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y	Try it »

<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y	Try it »
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.	Try it »

PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.

The PHP decrement operators are used to decrement a variable's value.

Operator	**Name**	**Description**	**Show it**
++$x	Pre-increment	Increments $x by one, then returns $x	Try it »
$x++	Post-increment	Returns $x, then increments $x by one	Try it »
--$x	Pre-decrement	Decrements $x by one, then returns $x	Try it »
$x--	Post-decrement	Returns $x, then decrements $x by one	Try it »

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result	Show it
and	And	$x and $y	True if both $x and $y are true	Try it »
or	Or	$x or $y	True if either $x or $y is true	Try it »
xor	Xor	$x xor $y	True if either $x or $y is true, but not both	Try it »
&&	And	$x && $y	True if both $x and $y are true	Try it »
||	Or	$x || $y	True if either $x or $y is true	Try it »
!	Not	!$x	True if $x is not true	Try it »

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result	Show it
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2	Try it »

.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1	Try it »

PHP Array Operators

The PHP array operators are used to compare arrays.

Operator	**Name**	**Example**	**Result**	**Show it**
+	Union	$x + $y	Union of $x and $y	Try it »
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs	Try it »
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types	Try it »
!=	Inequality	$x != $y	Returns true if $x is not equal to $y	Try it »
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y	Try it »
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y	Try it »

PHP Conditional Assignment Operators

The PHP conditional assignment operators are used to set a value depending on conditions:

Operator	**Name**	**Example**	**Result**	**Show it**
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE	Try it »
??	Null coalescing	$x = *expr1* ?? *expr2*	Returns the value of $x. The value of $x is *expr1* if *expr1* exists, and is not NULL. If *expr1* does not exist, or is NULL, the value of $x is *expr2*. Introduced in PHP 7	

PHP if...else...elseif Statements

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif...else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

PHP - The if Statement

The if statement executes some code if one condition is true.

Syntax

```
if (condition) {
  code to be executed if condition is true;
}
```

Example

Output "Have a good day!" if the current time (HOUR) is less than 20:

```
<?php
$t = date("H");
if ($t < "20") {
  echo "Have a good day!";
}
?>
```

Try it Yourself »

PHP - The if...else Statement

The if...else statement executes some code if a condition is true and another code if that condition is false.

Syntax

```
if (condition) {
  code to be executed if condition is true;
} else {
  code to be executed if condition is false;
}
```

Example

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```
<?php
$t = date("H");
if ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```

Try it Yourself »

PHP - The if...elseif...else Statement

The if...elseif...else statement executes different codes for more than two conditions.

Syntax

```
if (condition) {
  code to be executed if this condition is true;
} elseif (condition) {
  code to be executed if first condition is false and this condition is true;
} else {
  code to be executed if all conditions are false;
}
```

Example

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```
<?php
$t = date("H");
if ($t < "10") {
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```

Try it Yourself »

The PHP switch Statement

Use the switch statement to **select one of many blocks of code to be executed**.

Syntax

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use break to prevent the code from running into the next case automatically.
The default statement is used if no match is found.

Example

```
<?php
$favcolor = "red";
switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>
```

Try it Yourself »

PHP Loops

Often when you write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.

Loops are used to execute the same block of code again and again, as long as a certain condition is true.

In PHP, we have the following loop types:

- while - loops through a block of code as long as the specified condition is true
- do...while - loops through a block of code once, and then repeats the loop as long as the specified condition is true
- for - loops through a block of code a specified number of times
- foreach - loops through a block of code for each element in an array

The following chapters will explain and give examples of each loop type.

❮ PreviousNext ❯

ADVERTISEMENT

The PHP while Loop

The while loop executes a block of code as long as the specified condition is true.

Syntax

```
while (condition is true) {
  code to be executed;
}
```

Examples

The example below displays the numbers from 1 to 5:

Example

```
<?php
$x = 1;
```

```
while($x <= 5) {
  echo "The number is: $x <br>";
  $x++;
}
?>
```

Try it Yourself »

Example Explained

- $x = 1; - Initialize the loop counter ($x), and set the start value to 1
- $x <= 5 - Continue the loop as long as $x is less than or equal to 5
- $x++; - Increase the loop counter value by 1 for each iteration

This example counts to 100 by tens:

Example

```
<?php
$x = 0;
while($x <= 100) {
  echo "The number is: $x <br>";
  $x+=10;
}
?>
```

Try it Yourself »

Example Explained

- $x = 0; - Initialize the loop counter ($x), and set the start value to 0
- $x <= 100 - Continue the loop as long as $x is less than or equal to 100
- $x+=10; - Increase the loop counter value by 10 for each iteration

The PHP do...while Loop

The do...while loop will always execute the block of code once, it will then check the condition, and repeat the loop while the specified condition is true.

Syntax

```
do {
  code to be executed;
} while (condition is true);
```

Examples

The example below first sets a variable $x to 1 ($x = 1). Then, the do while loop will write some output, and then increment the variable $x with 1. Then the condition is checked (is $x less than, or equal to 5?), and the loop will continue to run as long as $x is less than, or equal to 5:

Example

```
<?php
$x = 1;
do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
?>
```

Try it Yourself »

Note: In a do...while loop the condition is tested AFTER executing the statements within the loop. This means that the do...while loop will execute its statements at least once, even if the condition is false. See example below.

This example sets the $x variable to 6, then it runs the loop, **and then the condition is checked**:

Example

```
<?php
$x = 6;
do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
?>
```

Try it Yourself »

The PHP for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (init counter; test counter; increment counter) {
  code to be executed for each iteration;
}
```

Parameters:

- *init counter*: Initialize the loop counter value
- *test counter*: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.
- *increment counter*: Increases the loop counter value

Examples

The example below displays the numbers from 0 to 10:

Example

```
<?php
for ($x = 0; $x <= 10; $x++) {
  echo "The number is: $x <br>";
}
?>
```

Try it Yourself »

Example Explained

- $x = 0; - Initialize the loop counter ($x), and set the start value to 0
- $x <= 10; - Continue the loop as long as $x is less than or equal to 10
- $x++ - Increase the loop counter value by 1 for each iteration

This example counts to 100 by tens:

Example

```
<?php
for ($x = 0; $x <= 100; $x+=10) {
  echo "The number is: $x <br>";
}
?>
```

Try it Yourself »

Example Explained

- $x = 0; - Initialize the loop counter ($x), and set the start value to 0
- $x <= 100; - Continue the loop as long as $x is less than or equal to 100
- $x+=10 - Increase the loop counter value by 10 for each iteration

The PHP foreach Loop

The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

Syntax

foreach ($*array* as $*value*) {
  code to be executed;
}

For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

Examples

The following example will output the values of the given array ($colors):

Example

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

Try it Yourself »

The following example will output both the keys and the values of the given array ($age):

Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
foreach($age as $x => $val) {
  echo "$x = $val<br>";
}
?>
```

PHP Break and Continue

❮ PreviousNext ❯

HP Break

You have already seen the break statement used in an earlier chapter of this tutorial. It was used to "jump out" of a switch statement.

The break statement can also be used to jump out of a loop.

This example jumps out of the loop when **x** is equal to **4**:

Example

```php
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    break;
  }
  echo "The number is: $x <br>";
}
?>
```

Try it Yourself »

PHP Continue

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

This example skips the value of **4**:

Example

```php
<?php
for ($x = 0; $x < 10; $x++) {
  if ($x == 4) {
    continue;
  }
  echo "The number is: $x <br>";
}
?>
```

Try it Yourself »

Break and Continue in While Loop

You can also use break and continue in while loops:

Break Example

```
<?php
$x = 0;
while($x < 10) {
  if ($x == 4) {
    break;
  }
  echo "The number is: $x <br>";
  $x++;
}
?>
```

Try it Yourself »

Continue Example

```
<?php
$x = 0;
while($x < 10) {
  if ($x == 4) {
    $x++;
    continue;
  }
  echo "The number is: $x <br>";
  $x++;
}
?>
```

Try it Yourself »

PHP Functions

HP Built-in 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 functions.

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.

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. Function names 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
?>
```

Try it Yourself »

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 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");
?>
```

Try it Yourself »

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");
?>
```

Try it Yourself »

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] . ".";
?>
```

Try it Yourself »

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
- **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] . ".";
?>
```

Try it Yourself »

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>";
}
?>
```

Try it Yourself »

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:

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
<?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>";
?>
```

Try it Yourself »

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):

Example

```php
<?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>";
}
?>
```

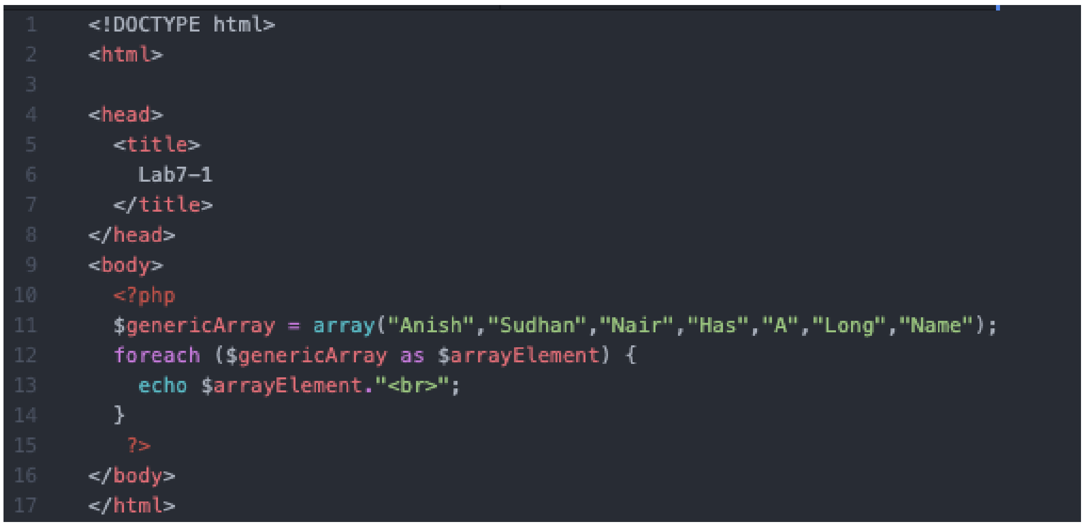
PART-B

Roll No.: K041	**Name:** Anish Sudhan Nair
Prog/Yr/Sem: B.Tech. Cybersecurity	**Batch:** K2/A2
Date of Experiment: 18/02/2022	**Date of Submission:** 25/02/2022

Code:-

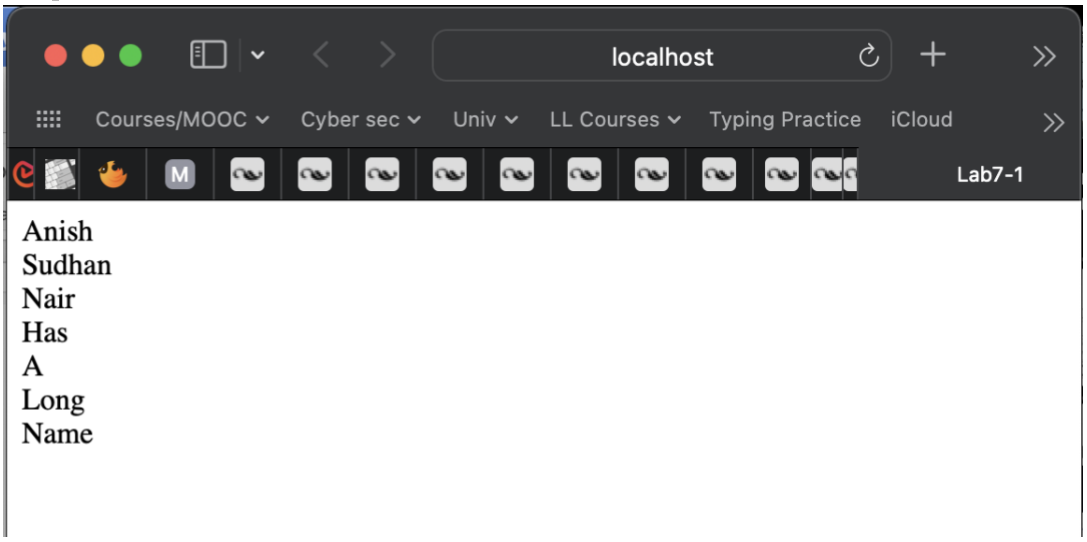
1. Print all element of array using for loop.

Code:



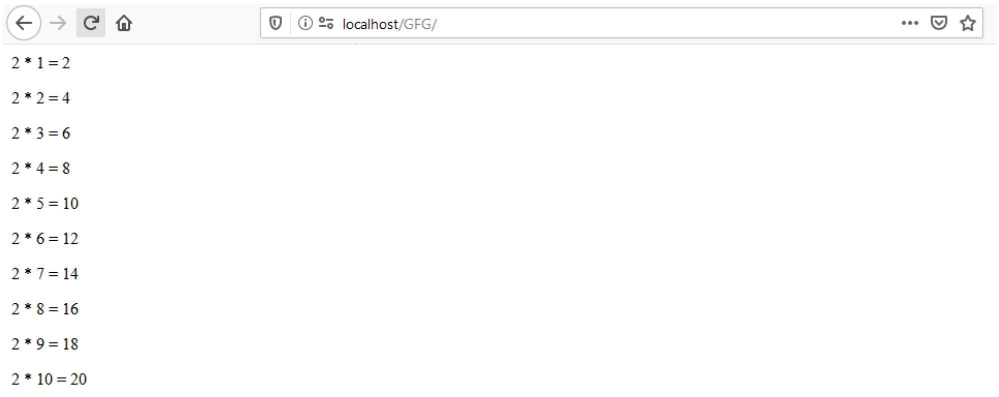
```
<!DOCTYPE html>
<html>

<head>
  <title>
    Lab7-1
  </title>
</head>
<body>
  <?php
  $genericArray = array("Anish","Sudhan","Nair","Has","A","Long","Name");
  foreach ($genericArray as $arrayElement) {
    echo $arrayElement."<br>";
  }
   ?>
</body>
</html>
```

Output:



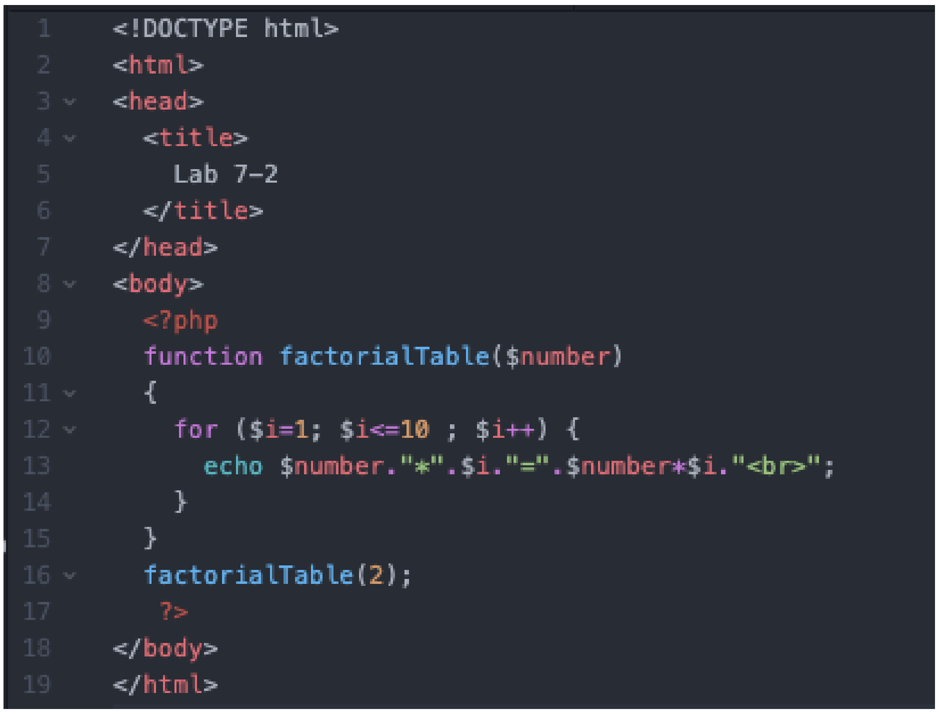
```
Anish
Sudhan
Nair
Has
A
Long
Name
```

2. Write a PHP code to get following output



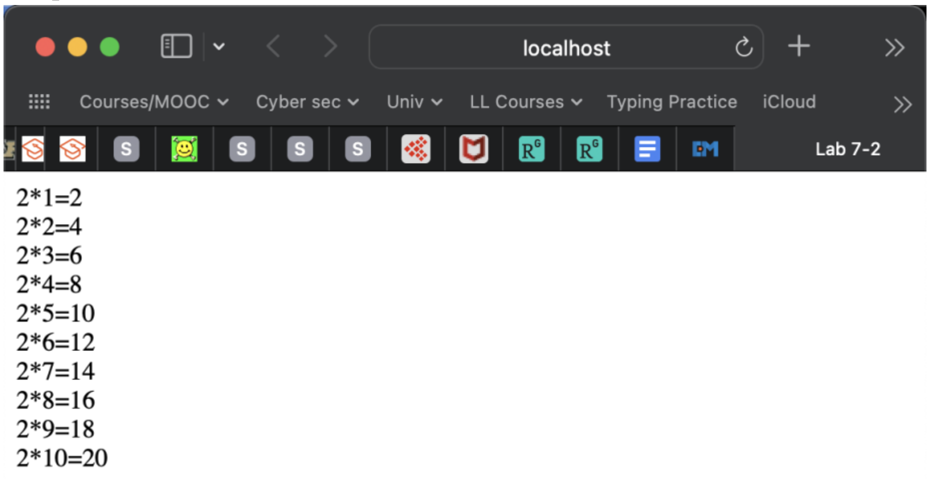
```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20
```

Code:



```
<!DOCTYPE html>
<html>
<head>
  <title>
    Lab 7-2
  </title>
</head>
<body>
  <?php
  function factorialTable($number)
  {
    for ($i=1; $i<=10 ; $i++) {
      echo $number."*".$i."=".$number*$i."<br>";
    }
  }
  factorialTable(2);
   ?>
</body>
</html>
```

Output:



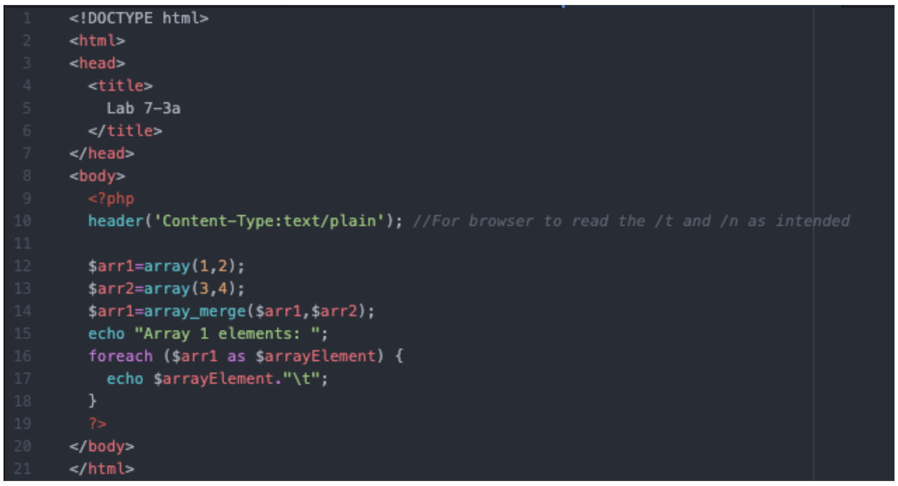
```
2*1=2
2*2=4
2*3=6
2*4=8
2*5=10
2*6=12
2*7=14
2*8=16
2*9=18
2*10=20
```

3. a

Given two array *arr1* and *arr2* and the task is to append one array to another array.
Examples:

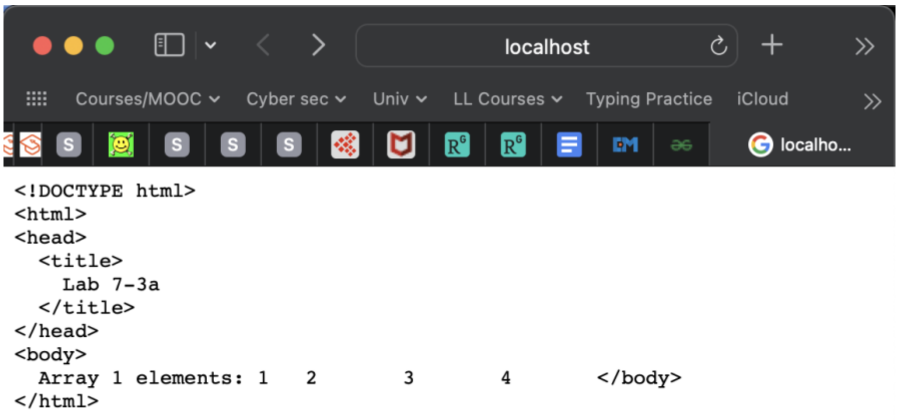
```
Input : arr1 = [ 1, 2 ]
        arr2 = [ 3, 4 ]


Output : arr1 = [ 1, 2, 3, 4 ]
```

Code:



```
<!DOCTYPE html>
<html>
<head>
  <title>
    Lab 7-3a
  </title>
</head>
<body>
  <?php
  header('Content-Type:text/plain'); //For browser to read the /t and /n as intended

  $arr1=array(1,2);
  $arr2=array(3,4);
  $arr1=array_merge($arr1,$arr2);
  echo "Array 1 elements: ";
  foreach ($arr1 as $arrayElement) {
    echo $arrayElement."\t";
  }
  ?>
</body>
</html>
```

Output:



```
<!DOCTYPE html>
<html>
<head>
  <title>
    Lab 7-3a
  </title>
</head>
<body>
  Array 1 elements: 1	2	3	4	</body>
</html>
```

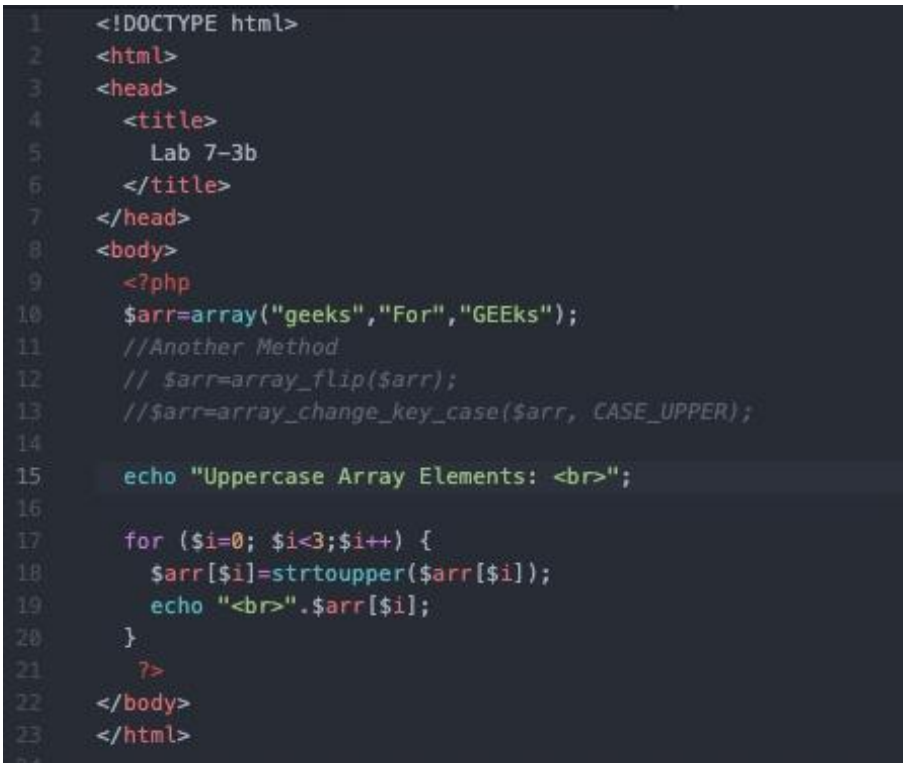
3.b

You are given an array of strings. You have to change all of the strings present in the given array to uppercase no matter which case they are currently. Print the resultant array.

Examples:

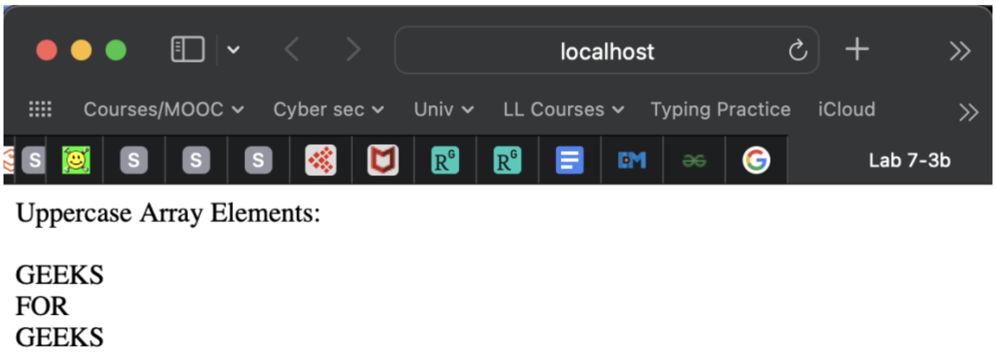
```
Input : arr[] = ("geeks", "For", "GEEks")
Output : Array ([0]=>GEEKS [1]=>FOR [2]=>GEEKS)

Input :  arr[] = ("geeks")
Output : Array ([0]=>GEEKS)
```

Code:



```
<!DOCTYPE html>
<html>
<head>
  <title>
    Lab 7-3b
  </title>
</head>
<body>
  <?php
  $arr=array("geeks","For","GEEks");
  //Another Method
  // $arr=array_flip($arr);
  //$arr=array_change_key_case($arr, CASE_UPPER);

  echo "Uppercase Array Elements: <br>";

  for ($i=0; $i<3;$i++) {
    $arr[$i]=strtoupper($arr[$i]);
    echo "<br>".$arr[$i];
  }
   ?>
</body>
</html>
```

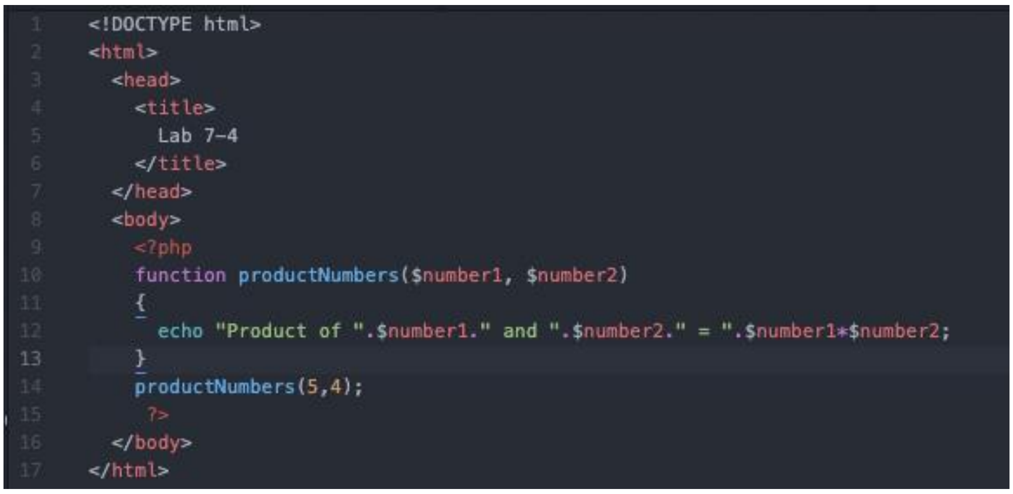
Output:



```
Uppercase Array Elements:

GEEKS
FOR
GEEKS
```

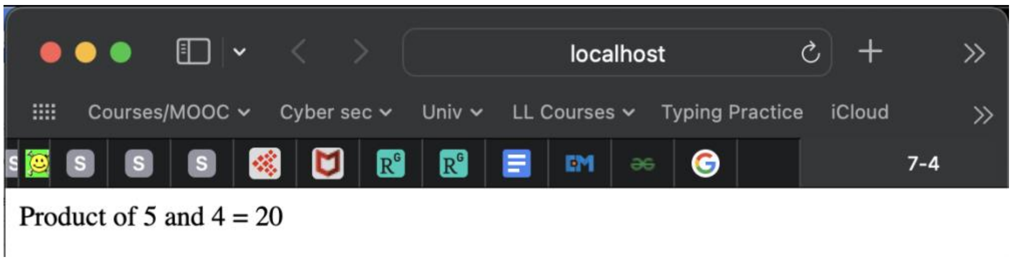
4. Write a PHP code to using function to find product of 2 numbers.

Code:



```
<!DOCTYPE html>
<html>
  <head>
    <title>
      Lab 7-4
    </title>
  </head>
  <body>
    <?php
    function productNumbers($number1, $number2)
    {
      echo "Product of ".$number1." and ".$number2." = ".$number1*$number2;
    }
    productNumbers(5,4);
     ?>
  </body>
</html>
```

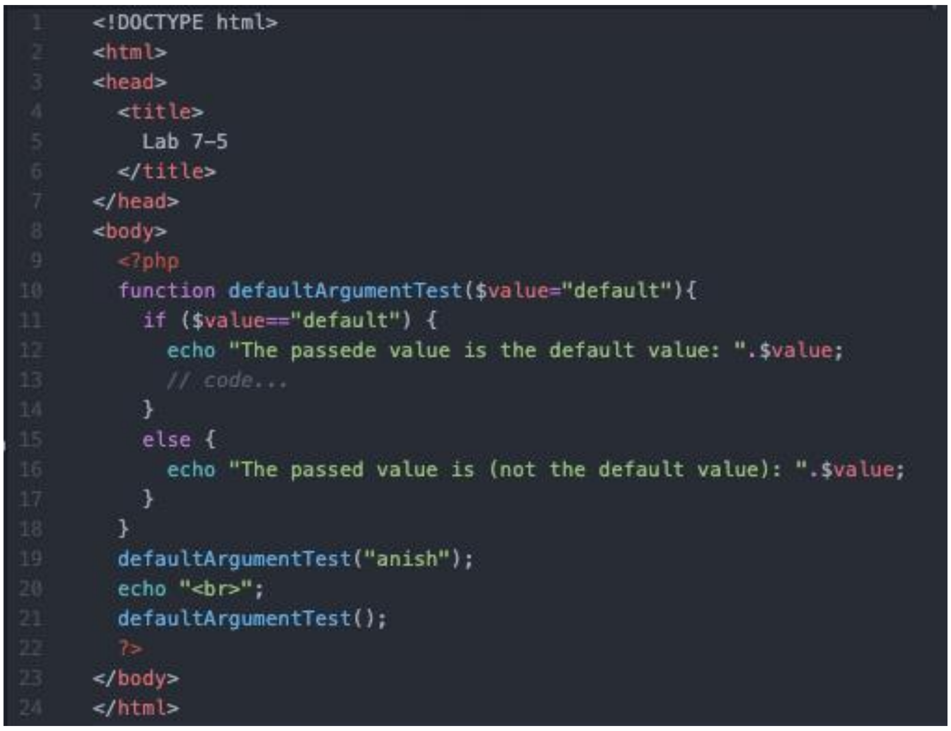
Output:



```
Product of 5 and 4 = 20
```

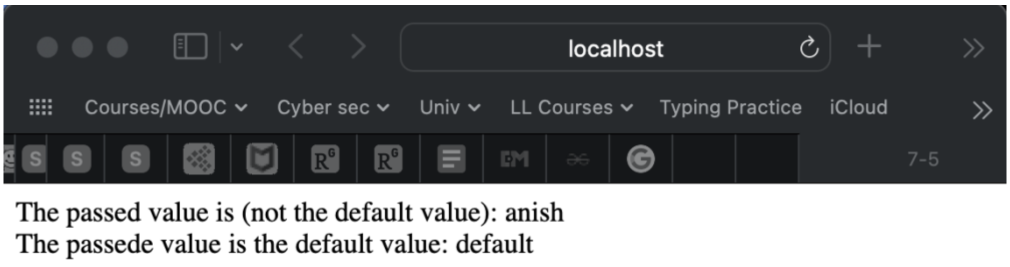
5 Write a PHP code using function to demonstrate use of default argument.

Code:



```
<!DOCTYPE html>
<html>
<head>
  <title>
    Lab 7-5
  </title>
</head>
<body>
  <?php
  function defaultArgumentTest($value="default"){
    if ($value=="default") {
      echo "The passede value is the default value: ".$value;
      // code...
    }
    else {
      echo "The passed value is (not the default value): ".$value;
    }
  }
  defaultArgumentTest("anish");
  echo "<br>";
  defaultArgumentTest();
  ?>
</body>
</html>
```

Output:



```
The passed value is (not the default value): anish
The passede value is the default value: default
```

Input and Output:-

Observation and Learning:-

We observed the use of php in web pages by writing php code within html and further we used a server (php-local-server package in my case) to view the output. PHP has functional use and is used for server side scripting; this experiment establishes itself as an introduction to the same.

Conclusion:-

We learnt to work with conditional statements, loops, arrays, array and string pre built functions and also user defined functions.

Questions:-