

UNIT-V

PHP

PHP Introduction

What You Should Already Know

Before you continue you should have a basic understanding of the following: [HTML](#), [CSS](#), [JavaScript](#)

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

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

- **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

PHP Installation Steps

- Go to “php.net”
- Download PHP
- Install
- Environment variable
- Verify the Installation of PHP
- -version PHP 7.4.9
- Path: C:\Program Files\php-7.4.9



- Downloads-windows downloads-download zip file-copy zip file and paste C:Program Files-Extract file and rename folder like 'php-version'.

php.net/downloads.php

php Downloads Documentation Get Involved Help php 8.2

Current Stable PHP 8.2.7 (Changelog)

- [php-8.2.7.tar.gz \(sig\)](#) [18,622Kb] 08 Jun 2023
sha256: 7046f939f0e5116285341d55c06af1d50907e107ac2c70defc32ef880f88cde4
- [php-8.2.7.tar.bz2 \(sig\)](#) [14,907Kb] 08 Jun 2023
sha256: 5bfb2a35c67921bdcadd5c90cb290ad7537d24da113a5e8bc2d646b02de7488f
- [php-8.2.7.tar.xz \(sig\)](#) [11,735Kb] 08 Jun 2023
sha256: 4b9fb3dcd7184fe7582d7e44544ec7c5153852a2528de3b6754791258ffbdafa0
- [Windows downloads](#)

[GPG Keys for PHP 8.2](#)

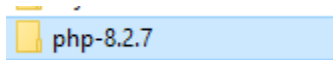
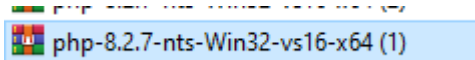
PHP 8.2 (8.2.7)

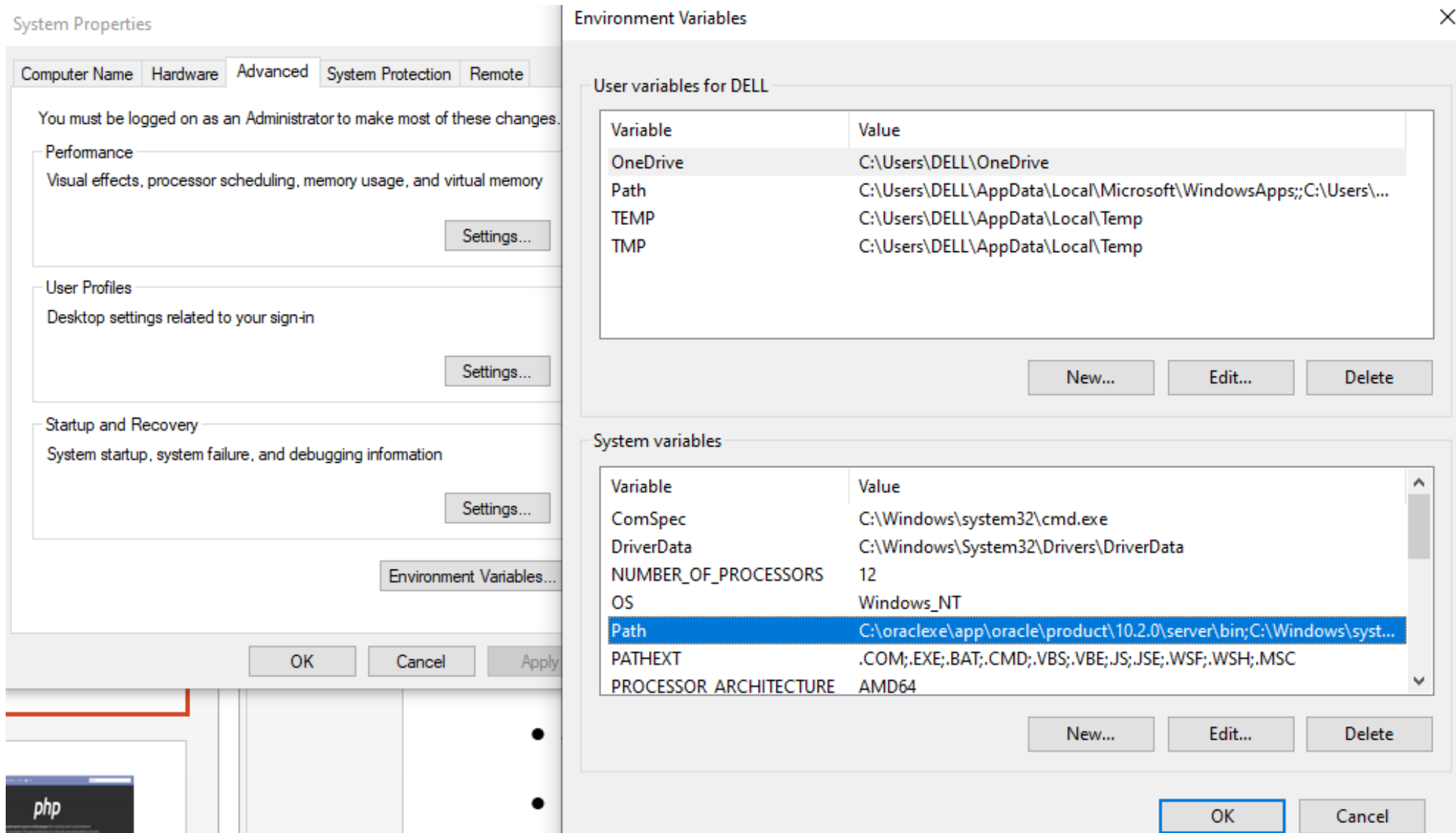
[Download source code](#) [26.03MB]

[Download tests package \(phpt\)](#) [15.54MB]

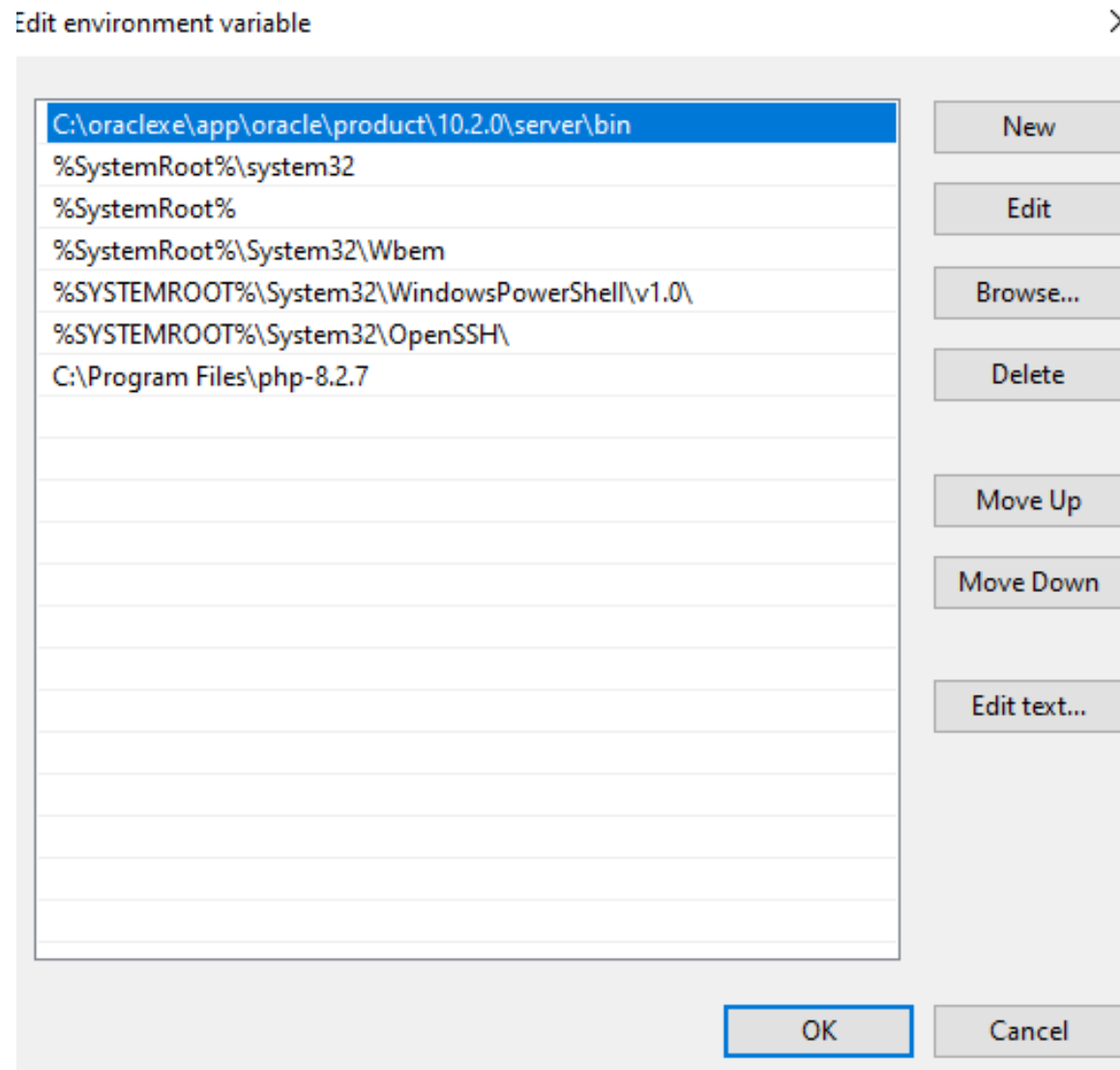
VS16 x64 Non Thread Safe (2023-Jun-07 10:56:22)

- [Zip](#) [30.23MB]
sha256: c69ea318dd3185931be29431de2cd04260806acd2ede08dd8e869aec02995dba
- [Debug Pack](#) [24.5MB]
sha256: e2e256c0ee509d37f7638e85fcc8dcaed045728ec85d3692c28af95d9270a833
- [Development package \(SDK to develop PHP extensions\)](#) [1.23MB]
sha256: fd47fa805bb869eac37c6cb387a3c4f874144d686c757bd4b06b55f5402a216e





- Edit-new-copy path where php is installed like-C:\Program Files\php-8.2.7-OK.



- Check version-go to cmd prompt-right click-run as administrator

 Select Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>php -v
PHP 8.2.7 (cli) (built: Jun  7 2023 10:25:24) (NTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.2.7, Copyright (c) Zend Technologies

C:\Windows\system32>
```

Basic PHP Syntax

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.

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

<h1>My first PHP page</h1>

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

</body>
</html>
```

My first PHP page

Hello World!

What is XAMPP?

XAMPP is an open-source web server solution package. It is mainly used for web application testing on a local host webserver.

XAMPP stands for:

X = Cross-platform

A = Apache Server

M = MariaDB

P = PHP

P = Perl

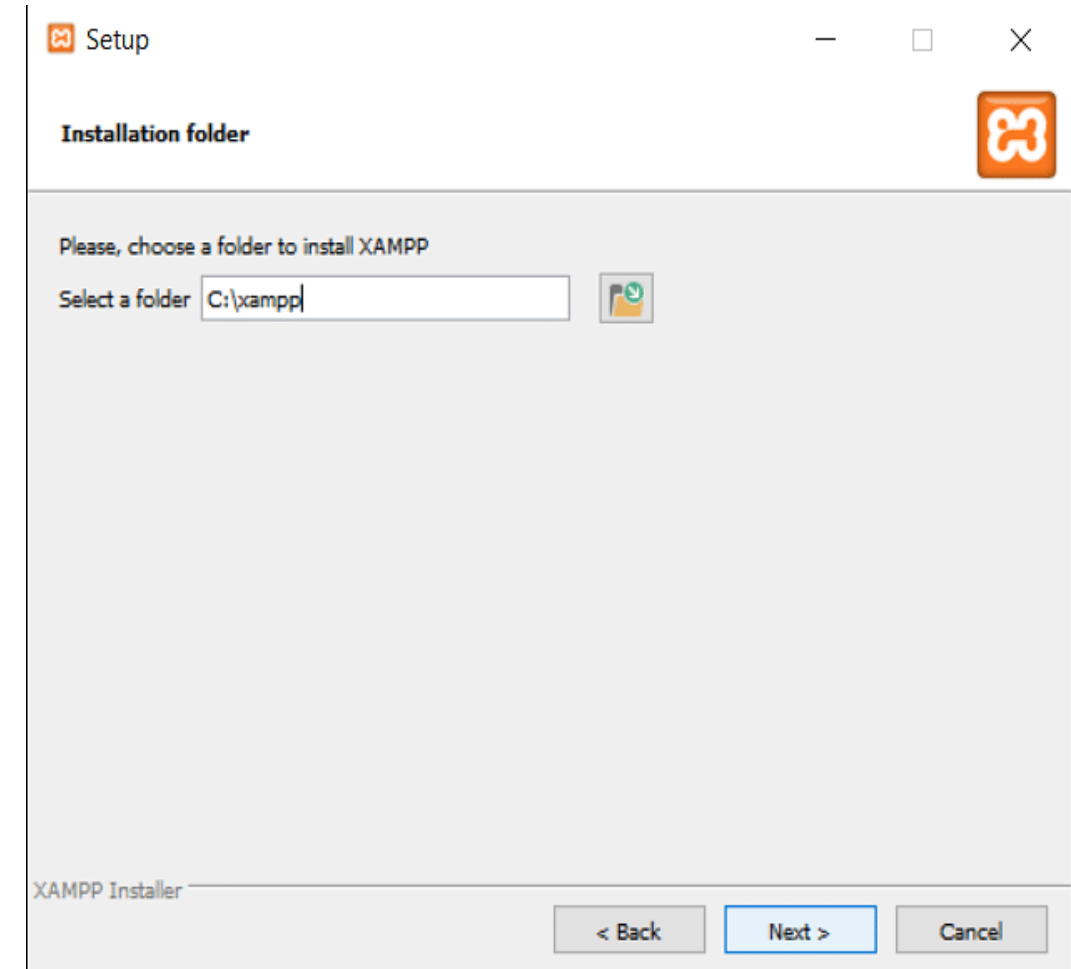
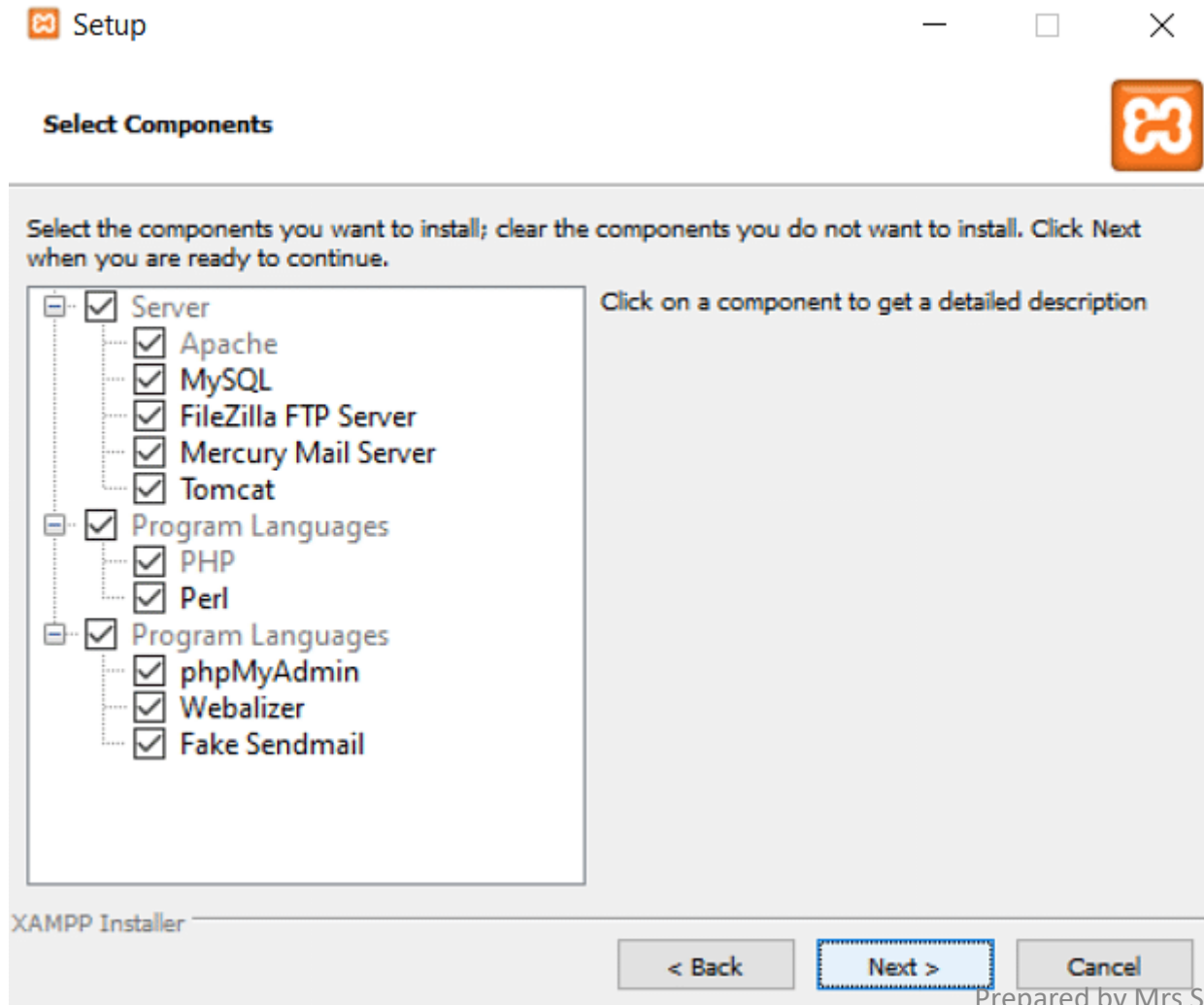
Now that you have a better understanding of the XAMPP software, let us move on to its installation.

Why Do You Need XAMPP?

To run any PHP program, you will need Apache or MYSQL databases, both supported by XAMPP.

You can download XAMPP through the official website, <https://www.apachefriends.org/download.html>

- On completing the download of the setup file, begin the installation process and, in the “Select Components” section, select all the required components.



- How to Start a New PHP Project in XAMPP?
- Before you run or start writing any program in PHP, you should start Apache and MYSQL.
- After starting both servers, you have to write a program in Notepad.
- After writing it, save that file as "program.php".
- Then copy that file program.php to C:/Program Files/XAMPP/htdocs.
- Open the browser and type <http://localhost>.
- Now run your code in that browser.

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)

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

```
<!DOCTYPE html>
<html>
<body>
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
echo $txt;
echo "<br>";
echo $x;
echo "<br>";
echo $y;
?>
</body>
</html>
```

o/p-Hello world!

5

10.5

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:

```
<!DOCTYPE html>
<html>
<body>
<?php
$txt = "FY-MCA-I";
echo "Good morning $txt!";
?>
</body>
</html>
```

O/P-Good morning FY-MCA-I!

The following example will output the sum of two variables:

```
<!DOCTYPE html>

<html>

<body>

<?php

$x = 5;

$y = 4;

echo $x + $y;

?>

</body>

</html>

o/p-9
```


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):

```
<!DOCTYPE html>
<html>
<body>
<?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.";
?>
</body>
</html>
```

O/P-

PHP is Fun!

Hello world!

I'm about to learn PHP!

This string was made with multiple parameters.

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):

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

O/P-

</body>

</html>

PHP is Fun!

Hello world!

I'm about to learn PHP!

PHP Data Types

- PHP supports the following data types:
- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP String

```
<!DOCTYPE html>

<html><body>

<?php

$x = "Hello world!";

$y = 'Hello world!';

echo $x;

echo "<br>";

echo $y;

?></body></html>

O/P-

Hello world!

Hello world!
```

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:

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

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

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

o/p-

int(5985)

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:

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

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

```
</body>
```

```
</html>
```

o/p-

float(10.365)

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

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

Booleans are often used in conditional testing.

```
<?php $var=10;  
var_dump($var>10);  
var_dump($var==true);  
?>
```

Output

This will produce following result –
bool(false) bool(true)

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:

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

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

O/P-

```
array(3) { [0]=> string(5) "Volvo" [1]=> string(3) "BMW"  
           [2]=>  
             string(6) "Toyota"  
           }
```

PHP String Functions

strlen() - Return the Length of a String

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

```
<!DOCTYPE html>
<html>
<body>
<?php
echo strlen("Hello world!");
?>
</body>
</html>
```

O/P-
12

strrev() - Reverse a String

The PHP `strrev()` function reverses a string.

```
<!DOCTYPE html>
<html>
<body>
<?php
echo strrev("Hello world!");
?>
</body>
</html>
```

O/P-
!dlrow olleH

str_word_count() - Count Words in a String

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

```
<!DOCTYPE html>
<html>
<body>
<?php
echo str_word_count("Hello world!");
?>
```

```
</body>
</html>
O/P-
2
```

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.

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

```
<!DOCTYPE html>
<html>
<body>
<?php
echo strpos("Hello world!", "world");
?>
```

```
</body>
```

```
</html>
```

O/P-6

str_replace() - Replace Text Within a String

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

```
<!DOCTYPE html>
<html>
<body>
<?php
echo str_replace("world", "Dolly", "Hello world!");
?>
```

```
</body>
</html>
O/P-
Hello Dolly!
```

PHP Constants

A constant is an identifier (name) for a simple value. The value cannot be changed during the script.

A valid constant name starts with a letter or underscore (no \$ sign before the constant name).

Create a PHP Constant

To create a constant, use the `define()` function.

Syntax

`define(name, value, case-insensitive)`

Parameters:

- *name*: Specifies the name of the constant
- *value*: Specifies the value of the constant
- *case-insensitive*: Specifies whether the constant name should be case-insensitive. Default is false

```
<!DOCTYPE html>
<html>
<body>
<?php
// case-sensitive constant name
define("GREETING", "Welcome to FY-MCA-I!");
echo GREETING;
?>
</body>

</html>
```

O/P-

Welcome to FY-MCA-I!

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

<?php
// case-insensitive constant name
define("GREETING", "Welcome to FY-MCA-I!",
true);
echo greeting;
?>

</body>

</html>
```

O/P-

Welcome to FY-MCA-I!

PHP Operators

- 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

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

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<?php
```

```
$x = 10;
```

```
$y = 6;
```

```
echo $x + $y;
```

```
?>
```

```
</body>
```

```
</html>
```

O/P-16

PHP Assignment Operators

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

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

```
<?php
$x = 20;
$x += 100;
```

```
echo $x;
?>
```

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

O/P
120

PHP Comparison Operators

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

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

```
<?php  
$x = 100;  
$y = "100";
```

```
var_dump($x == $y); //  
returns true because values  
are equal  
?>
```

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

```
D/P  
Bool(true)
```

```
<!DOCTYPE html>
<html>
<body>
<?php
$x = 5;
$y = 10;
echo ($x <=> $y); // returns -1 because $x is less than $y
echo "<br>";
$x = 10;
$y = 10;
echo ($x <=> $y); // returns 0 because values are equal
echo "<br>";
$x = 15;
$y = 10;
echo ($x <=> $y); // returns +1 because $x is greater than $y
?>
</body>
</html>
```

O/P-

-1

0

1

PHP Increment / Decrement Operators

Operator	Name	Description
++\$x	Pre-increment	Increments \$x by one, then returns \$x
\$x++	Post-increment	Returns \$x, then increments \$x by one
--\$x	Pre-decrement	Decrements \$x by one, then returns \$x
\$x--	Post-decrement	Returns \$x, then decrements \$x by one

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

```
<?php
$x = 10;
echo ++$x;
?>
```

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

O/P-11

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

```
<?php
$x = 10;
echo $x--;
?>
```

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

O/P-10

PHP Logical Operators

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

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

```
<?php
$x = 100;
$y = 50;
```

```
if ($x == 100 and $y == 50) {
    echo "Hello world!";
}
?>
```

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

O/P-
Hello world!

PHP String Operators

Operator	Name	Example	Result
.	Concatenation	\$txt1 . \$txt2	Concatenation of \$txt1 and \$txt2
.=	Concatenation assignment	\$txt1 .= \$txt2	Appends \$txt2 to \$txt1

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

```
<?php
$txt1 = "Hello";
$txt2 = " world!";
echo $txt1 . $txt2;
?>
```

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

O/P-Hello world!

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

```
<?php
$txt1 = "Hello";
$txt2 = " world!";
$txt1 .= $txt2;
echo $txt1;
?>
```

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

O/P-Hello world!

- PHP Array Operators
- The PHP array operators are used to compare arrays.

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

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

```
<?php
$x = array("a" => "red", "b"
=> "green");
$y = array("c" => "blue",
"d" => "yellow");
```

```
print_r($x + $y); // union of
$x and $y
?>
```

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

```
o/p-
Array ( [a] => red [b] =>
green [c] => blue [d] =>
yellow )
```

PHP Conditional Assignment Operators

Operator	Name	Example	Result
?:	Ternary	<pre>\$x = expr1 ? expr2 : ex pr3</pre>	<p>Returns the value of \$x.</p> <p>The value of \$x is expr2 if expr1 = TRUE.</p> <p>The value of \$x is expr3 if expr1 = FALSE</p>
??	Null coalescing	<pre>\$x = expr1 ?? expr2</pre>	<p>Returns the value of \$x.</p> <p>The value of \$x is expr1 if expr1 exists, and is not NULL.</p> <p>If expr1 does not exist, or is NULL, the value of \$x is expr2.</p>

```
<!DOCTYPE html>
<html><body>
<?php
    // variable $user is the value of
    $_GET['user']
    // and 'anonymous' if it does not exist
    echo $user = $_GET["user"] ??
    "anonymous";
    echo("<br>");

    // variable $color is "red" if $color does
    not exist or is null
    echo $color = $color ?? "red";
?>
</body></html>
```

O/p-
anonymous
red

PHP Conditional Statements

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:

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

O/P-

Have a good day!

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:

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

O/P- Have a good day!

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:

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

O/P- Have a good day!

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;  
}
```

```
<!DOCTYPE html><html><body> <?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!";
```

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

O/P- Your favorite color is red!

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:

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<?php
```

```
$x = 1;
```

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

```
?>
```

```
</body>
```

```
</html>
```

O/p- The number is: 1

The number is: 2

The number is: 3

The number is: 4

The number is: 5

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:

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

O/P- The number is: 1

The number is: 2

The number is: 3

The number is: 4

The number is: 5

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:

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

O/P- The number is: 0

The number is: 1

The number is: 2

The number is: 3

The number is: 4

The number is: 5

The number is: 6

The number is: 7

The number is: 8

The number is: 9

The number is: 10

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):

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

O/P- red

green

blue

yellow

PHP 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**:

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

O/P- The number is: 0

The number is: 1

The number is: 2

The number is: 3

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**:

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

O/P- The number is: 0

The number is: 1

The number is: 2

The number is: 3

The number is: 5

The number is: 6

The number is: 7

The number is: 8

The number is: 9

PHP 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.

```
<!DOCTYPE html>  
<html>  
<body>  
<?php  
function writeMsg() {  
    echo "Hello world!";  
}  
writeMsg();  
>  
</body>  
</html>  
O/P-  
Hello world!
```

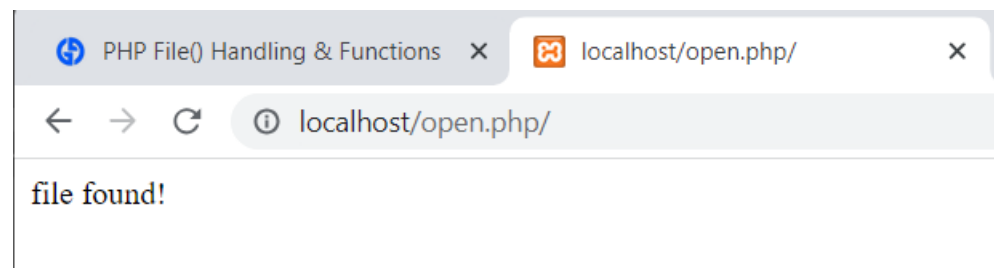
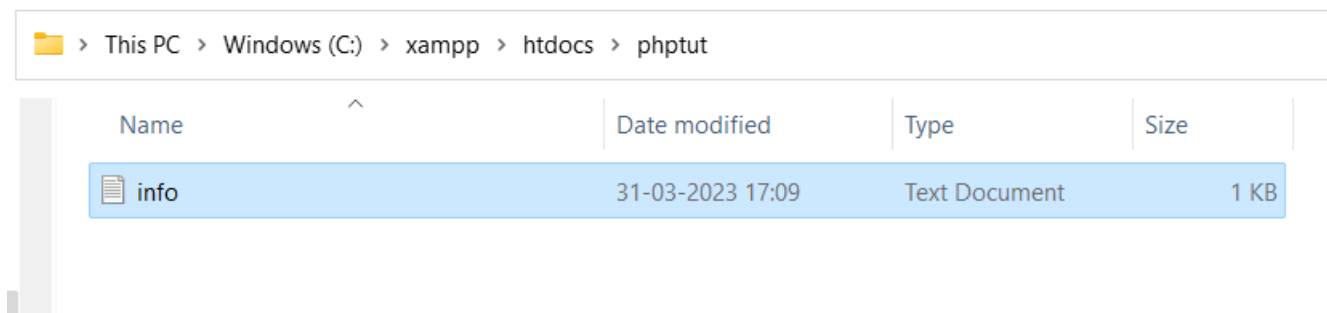
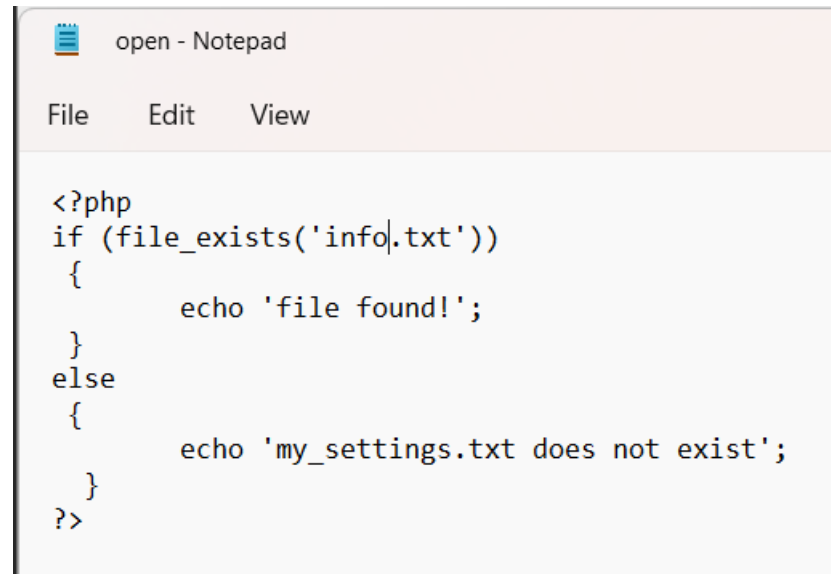
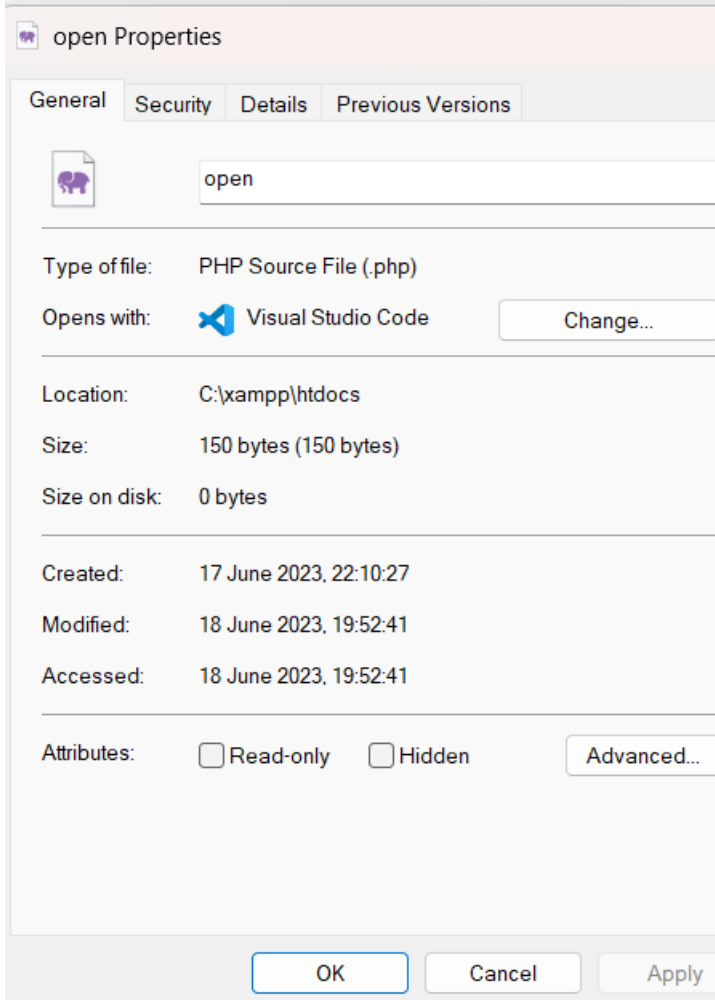
Passing Arguments by Reference

In PHP, arguments are usually passed by value, which means that a copy of the value is used in the function and the variable that was passed into the function cannot be changed.

When a function argument is passed by reference, changes to the argument also change the variable that was passed in. To turn a function argument into a reference, the **&** operator is used:

```
<!DOCTYPE html>
<html>
<body>
<?php
function add_five(&$value) {
    $value += 5;
}
$num = 2;
add_five($num);
echo $num;
?>
</body>
</html>
O/P-7
```

- PHP file_exists() Function
- This function is used to determine whether a file exists or not.
- It comes in handy when we want to know if a file exists or not before processing it.
- You can also use this function when creating a new file and you want to ensure that the file does not already exist on the server.
- The file_exists function has the following syntax.
- <?php
- file_exists(\$filename);
- ?>



• PHP File Handling

- PHP File System allows us to create file, read file line by line, read file character by character, write file, append file, delete file and close file.

PHP Open File - fopen()

The PHP fopen() function is used to open a file.

Syntax

•resource fopen (string \$filename , string \$mode [, bool \$use_include_path = false [, resource \$context]])

Example

PHP Open File Mode

```
<?php  
fopen($file_name,$mode,$use_include_path,$context); ?>
```

HERE,

- “fopen” is the PHP open file function
- “\$file_name” is the name of the file to be opened
- “\$mode” is the mode in which the file should be opened, the table below shows the modes

Mode	Description
R	Opens file in read-only mode. It places the file pointer at the beginning of the file.
r+	Opens file in read-write mode. It places the file pointer at the beginning of the file.
w	Opens file in write-only mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
w+	Opens file in read-write mode. It places the file pointer to the beginning of the file and truncates the file to zero length. If file is not found, it creates a new file.
A	Opens file in write-only mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
a+	Opens file in read-write mode. It places the file pointer to the end of the file. If file is not found, it creates a new file.
X	Creates and opens file in write-only mode. It places the file pointer at the beginning of the file. If file is found, fopen() function returns FALSE.
x+	It is same as x but it creates and opens file in read-write mode.
C	Opens file in write-only mode. If the file does not exist, it is created. If it exists, it is neither truncated (as opposed to 'w'), nor the call to this function fails (as is the case with 'x'). The file pointer is positioned on the beginning of the file
c+	It is same as c but it opens file in read-write mode.

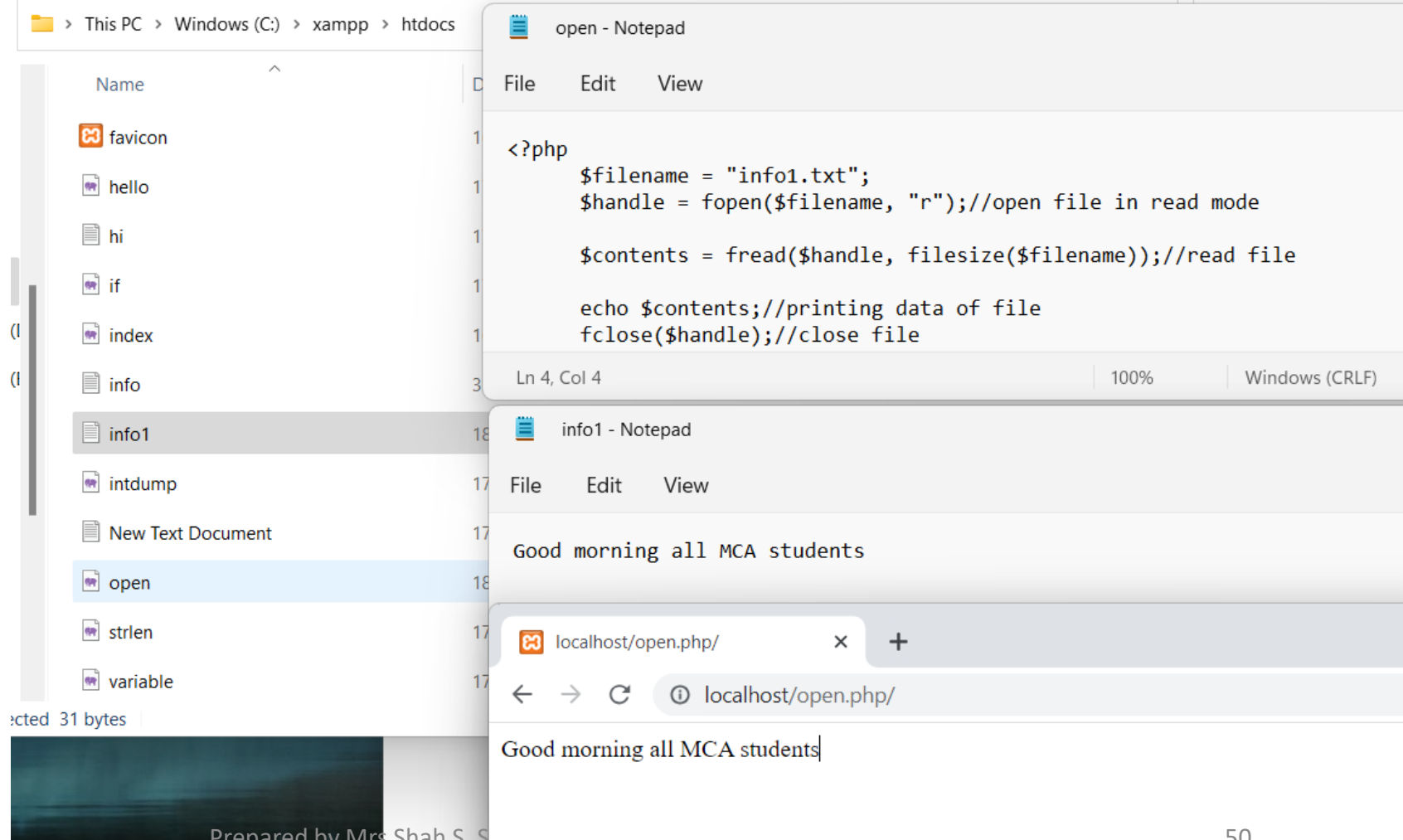
PHP Read File - fread()

The PHP fread() function is used to read the content of the file. It accepts two arguments: resource and file size.

Syntax

string fread (resource \$handle , int \$length)

Create txt file in c:/xampp/htdocs

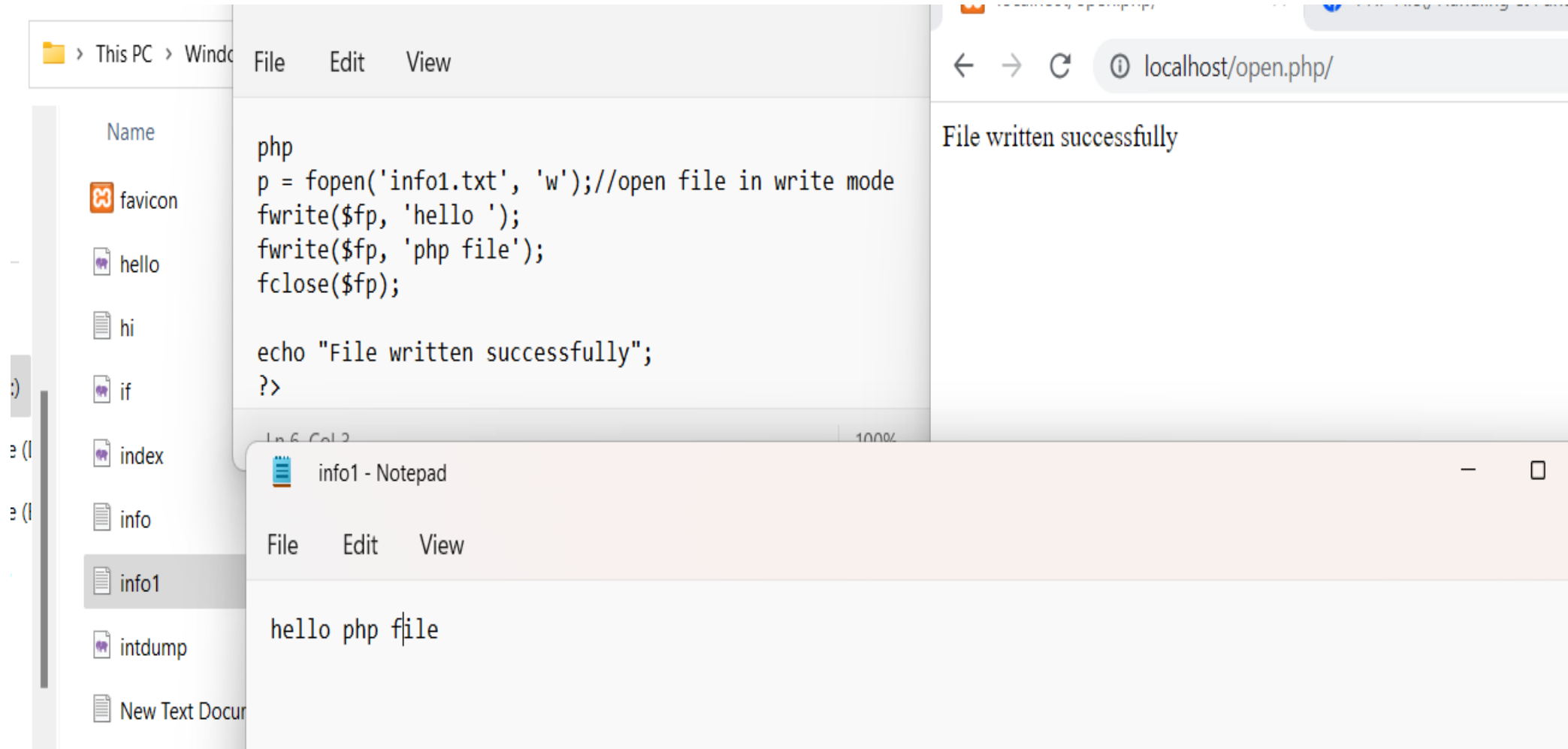


PHP Write File - fwrite()

The PHP fwrite() function is used to write content of the string into file.

Syntax

int fwrite (resource \$handle , string \$string [, int \$length])

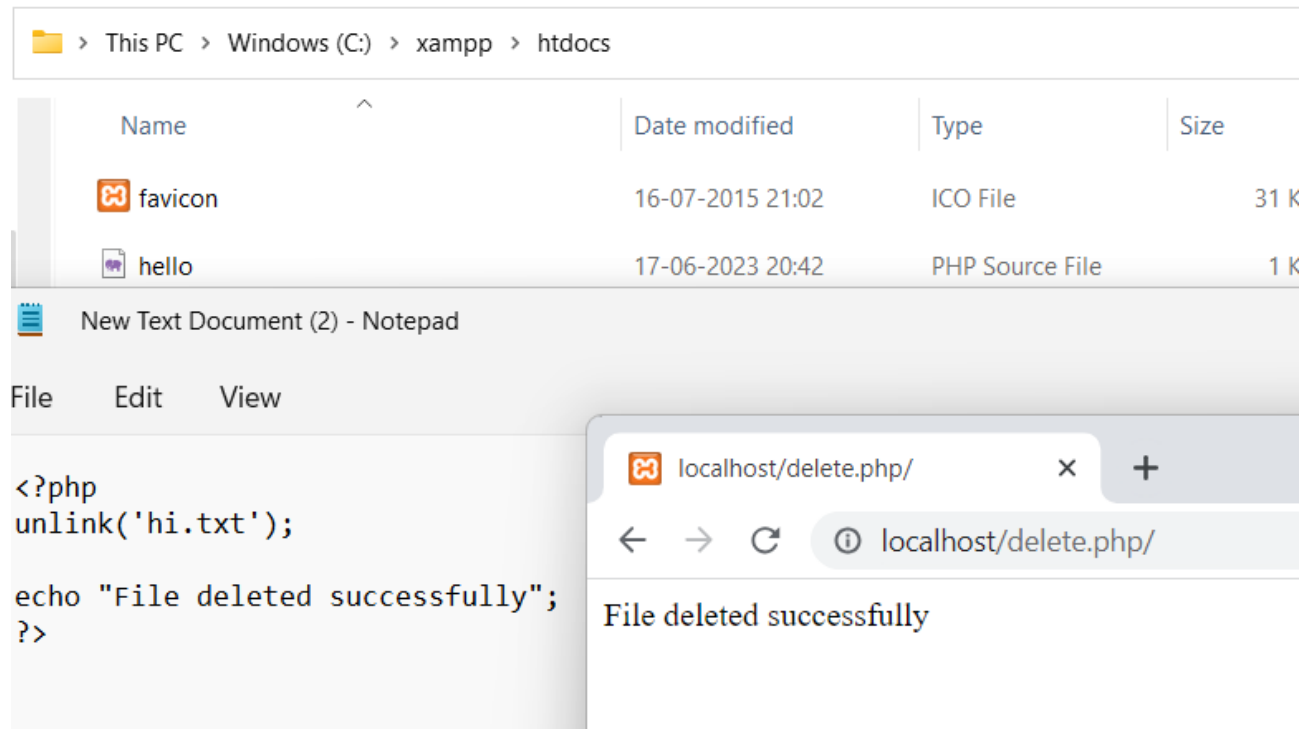


PHP Delete File - unlink()

The PHP unlink() function is used to delete file.

Syntax

bool unlink (string \$filename [, resource \$context])



PHP Copy file

PHP copy() Function

The copy() function is used to copy a file.

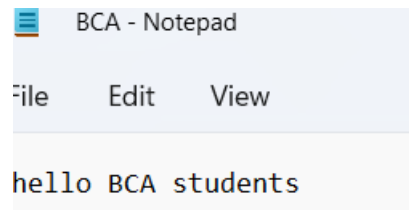
copy() function syntax

```
<?php
```

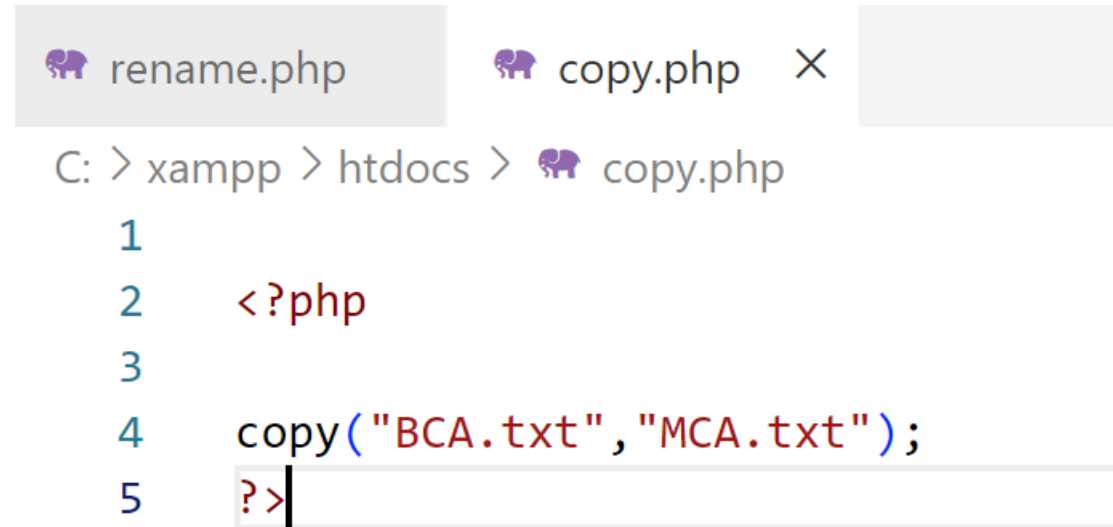
```
copy("source filename", "destination  
filename");
```

```
?>
```

Filename BCA must required in htdocs folder with content hello
BCA stydents.

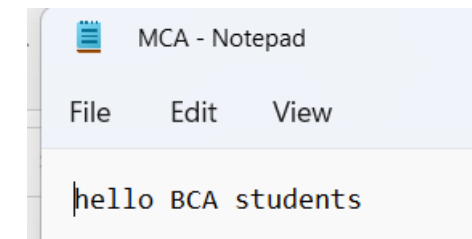


```
BCA - Notepad  
File Edit View  
hello BCA students
```



```
rename.php copy.php X  
C: > xampp > htdocs > copy.php  
1  
2 <?php  
3  
4 copy("BCA.txt", "MCA.txt");  
5 ?>
```

Output



```
MCA - Notepad  
File Edit View  
hello BCA students
```

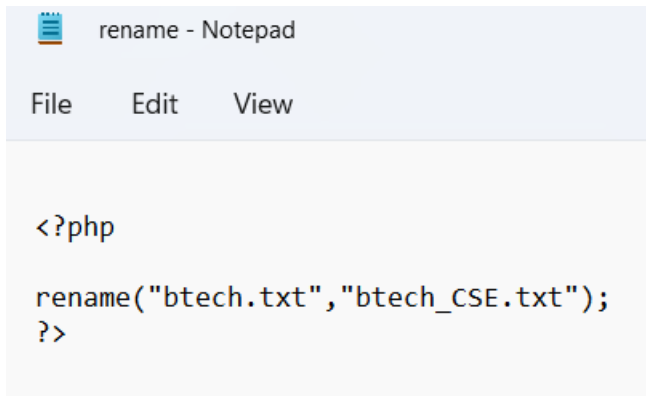
PHP rename() Function

The rename() function is used to rename a file name.

rename() function syntax

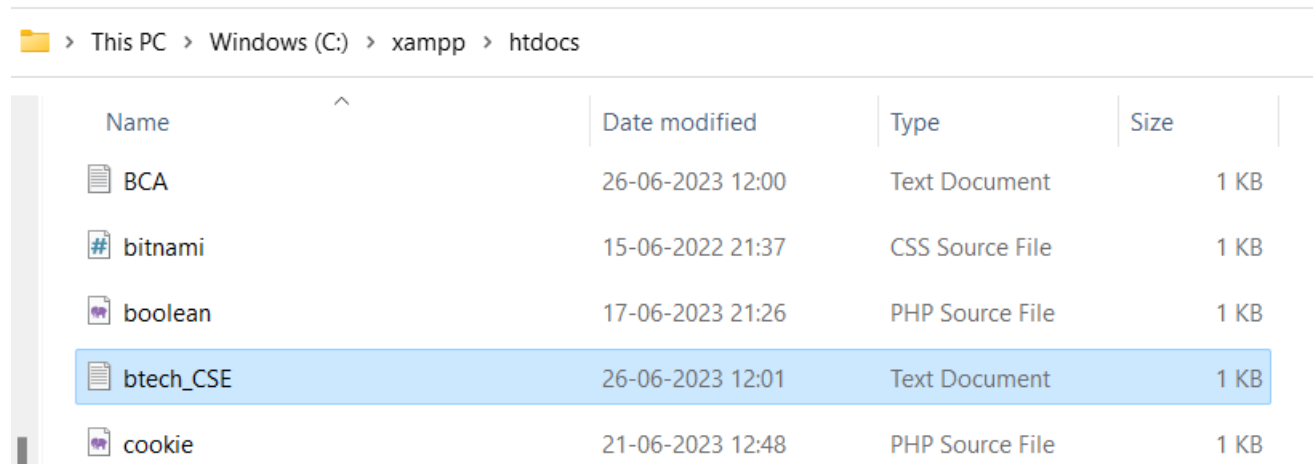
```
<?php  
rename("old filename","new filename");  
?>
```

Filename btech must
required in htdocs folder





```
rename - Notepad  
  
File Edit View  
  
<?php  
rename("btech.txt","btech_CSE.txt");  
?>
```

Output



This PC > Windows (C:) > xampp > htdocs				
Name	Date modified	Type	Size	
BCA	26-06-2023 12:00	Text Document	1 KB	
bitnami	15-06-2022 21:37	CSS Source File	1 KB	
boolean	17-06-2023 21:26	PHP Source File	1 KB	
btech_CSE	26-06-2023 12:01	Text Document	1 KB	
cookie	21-06-2023 12:48	PHP Source File	1 KB	

PHP File Upload

 rename.php  copy.php X

C: > xampp > htdocs >  copy.php

1

2

```
<?php
```

3

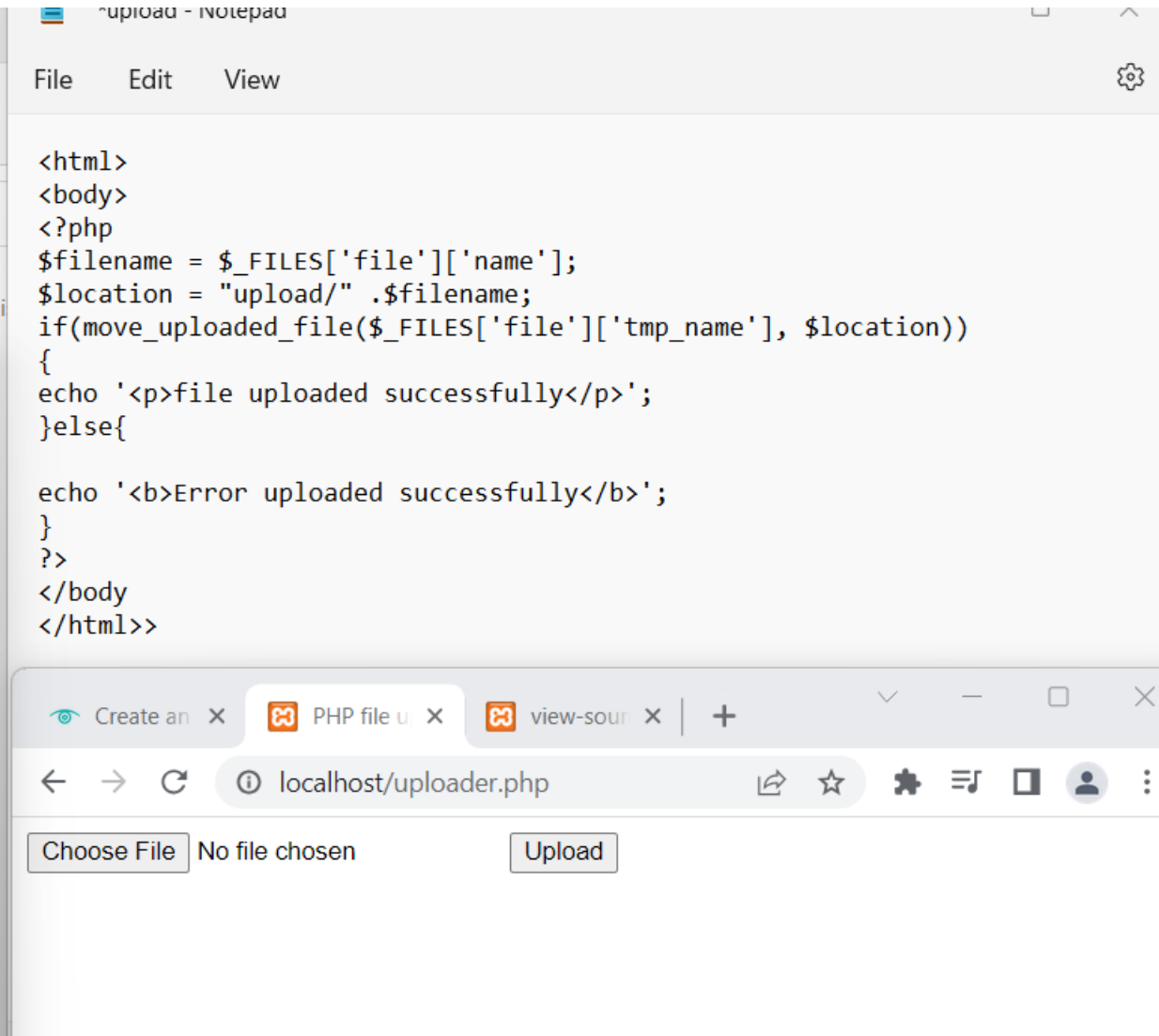
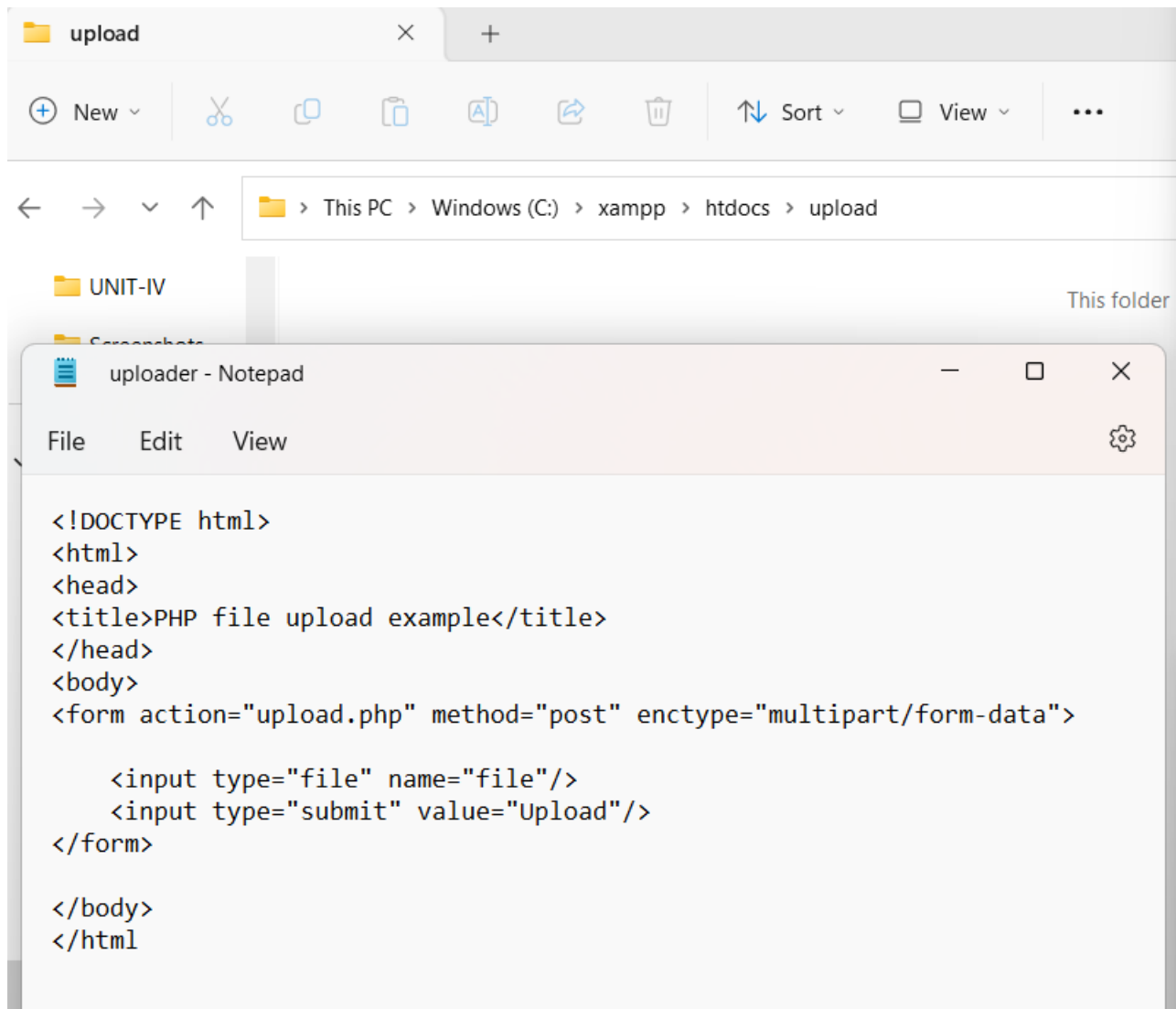
4

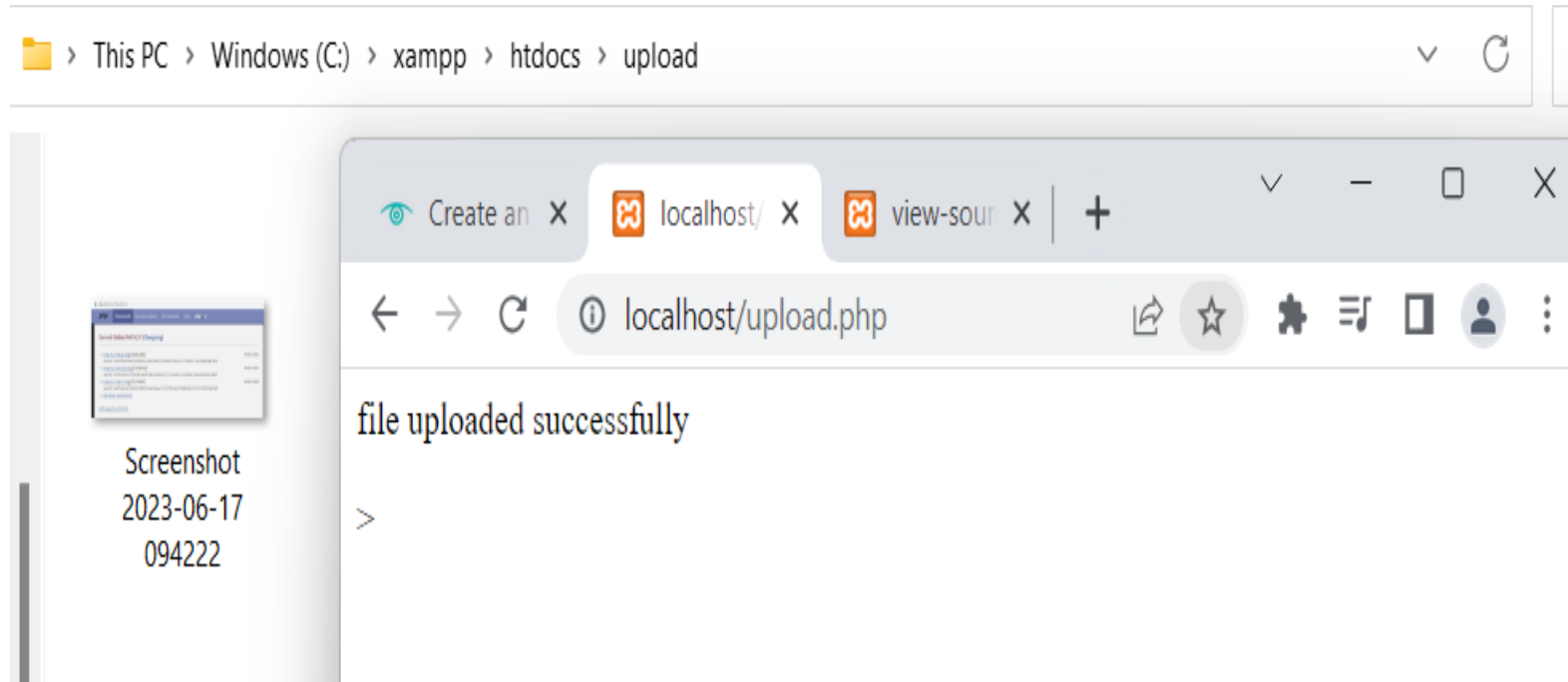
```
copy("BCA.txt", "MCA.txt");
```

5

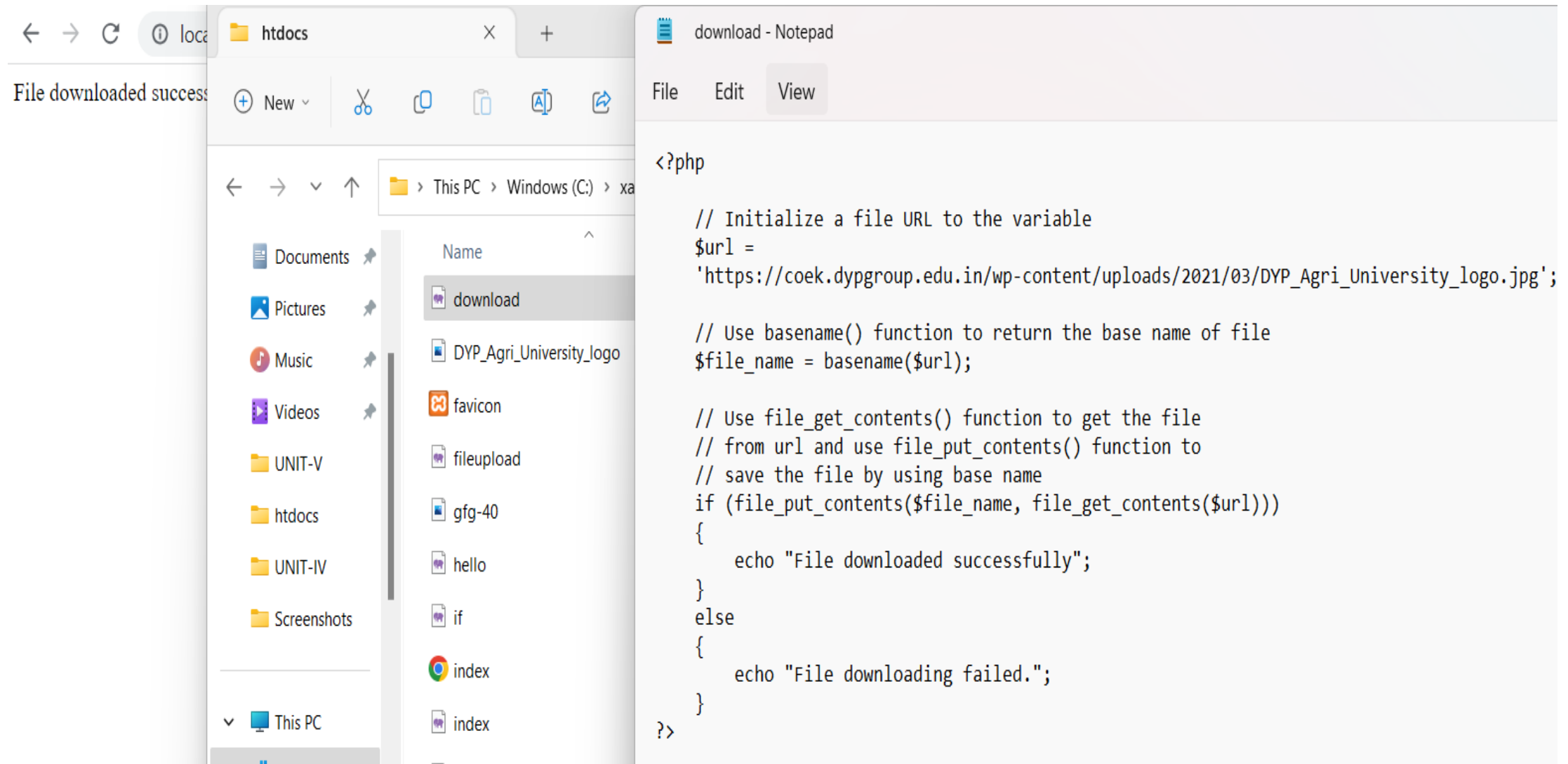
```
?>
```

PHP File Upload

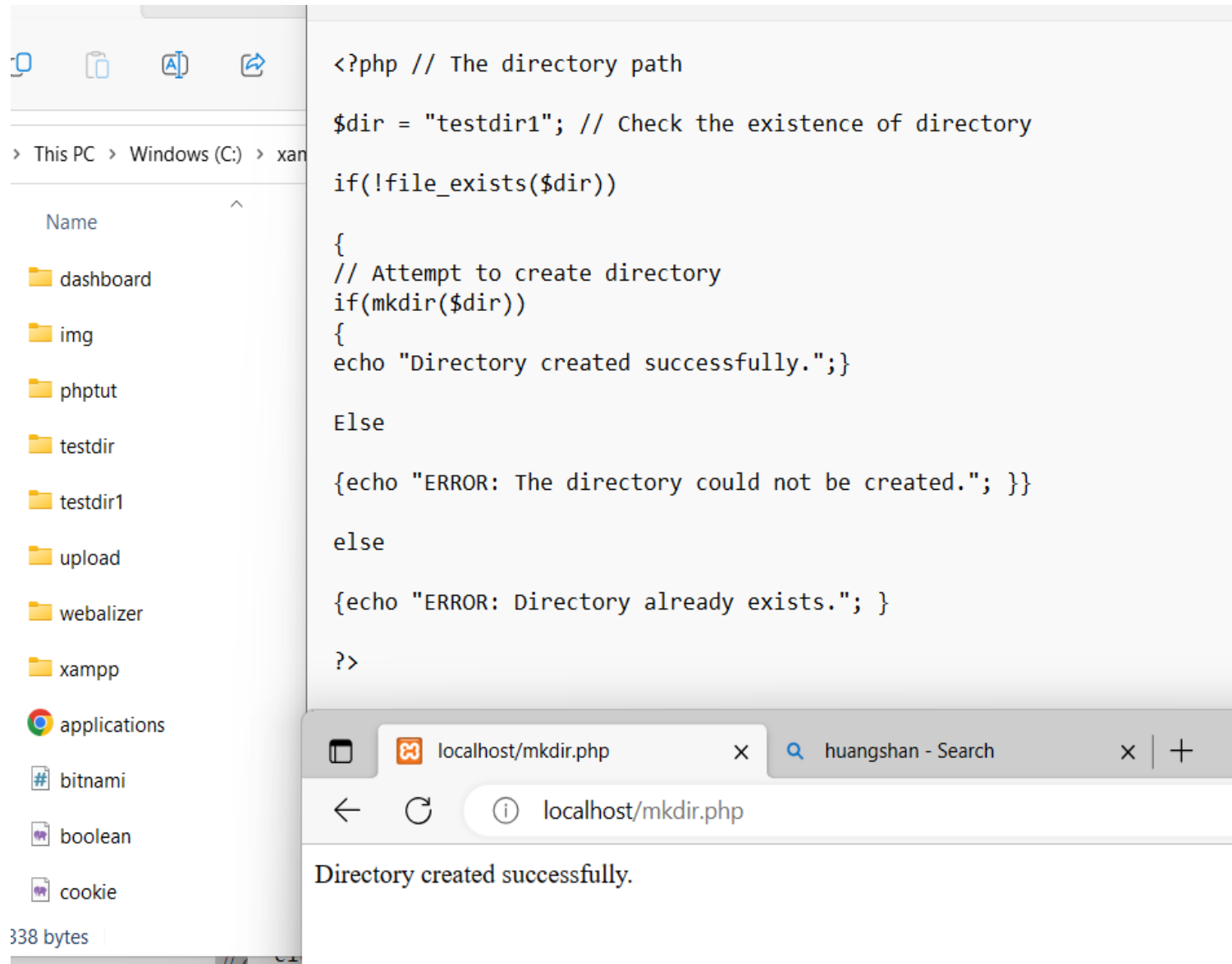




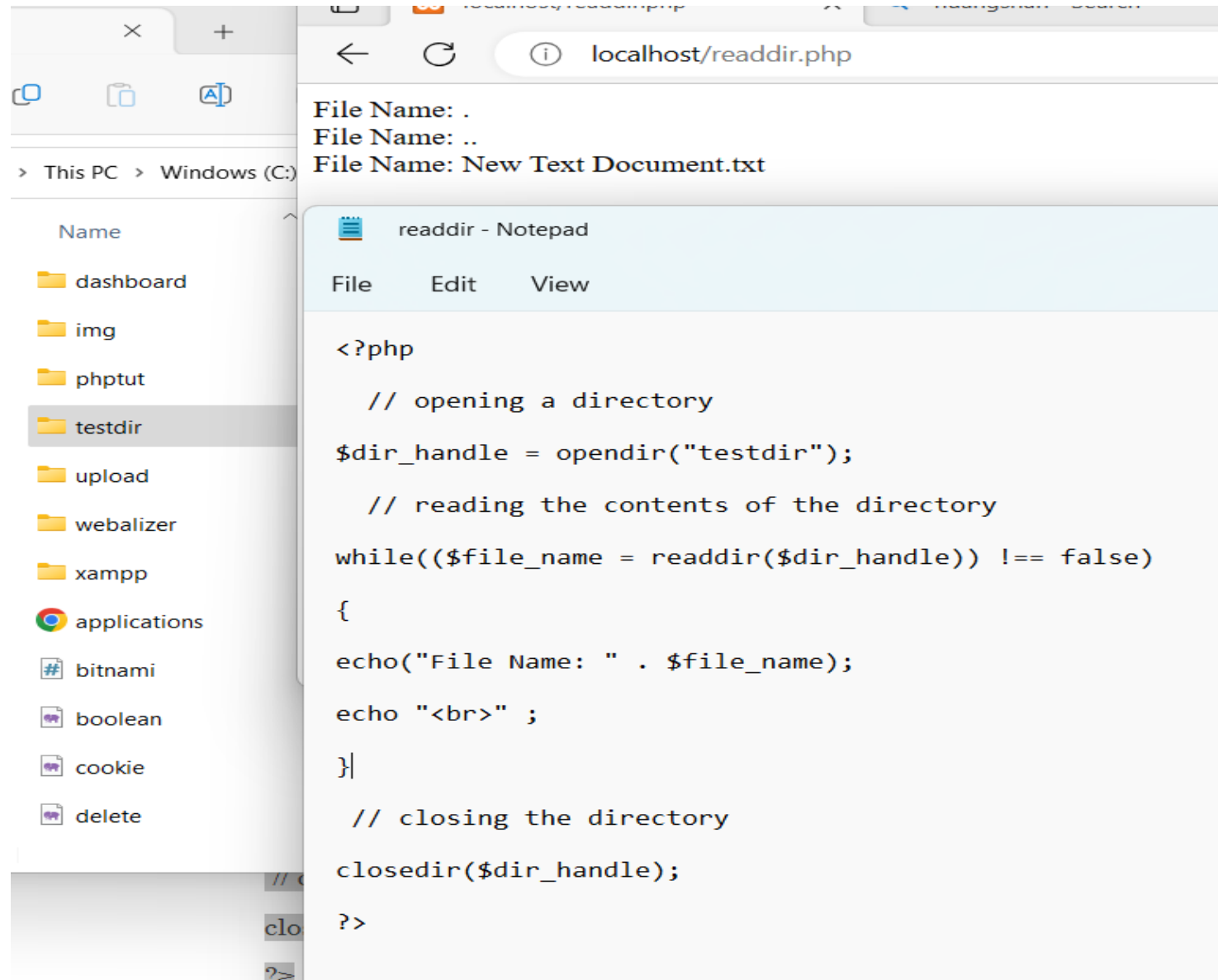
PHP Download File



mkdir



readdir



Delete dir.

The image shows a web application interface with two main components: a file explorer on the left and a code editor on the right.

File Explorer: The address bar shows the path `> This PC > Windows (C:) > xampp >`. The file list includes folders like `dashboard`, `img`, `phptut`, `testdir`, `testdir2` (highlighted), `upload`, `webalizer`, `xampp`, `applications`, `bitnami`, `boolean`, and `cookie`.

Code Editor: The editor contains PHP code for deleting a directory:

```
<?php
$directory_handle = opendir("testdir1");
while($directory_item = readdir($directory_handle)) {
    @unlink("testdir1/".$directory_item);
}
closedir($directory_handle);
$fp= rmdir("testdir1");
if(!file_exists($fp))
{
    echo("directory remove");
}
?>
```

Browser: The browser tab is titled `localhost/deletedir.php`. The address bar shows `localhost/deletedir.php`. The page content displays the output of the script: `directory remove`.

Read from one dir n Write to another dir

The image shows a web application running on localhost. A Notepad window displays the following PHP code:

```
<?php
mkdir("testdir3");

/*Creating files into php_directory_functions_manual*/

$file_pointer1 = fopen("testdir3/mkdir.txt","x");
$file_pointer2 = fopen("testdir3/rmdir.txt","x");
fclose($file_pointer1);
fclose($file_pointer2);

$directory_handle = opendir("testdir");
while($directory_item = readdir($directory_handle)) {
echo $directory_item . "<br>";
}
closedir($directory_handle);
?>
```

A file explorer window shows the contents of the 'testdir3' directory:

Name	Date modified	Type
mkdir	21-06-2023 14:45	Text Document
rmdir	21-06-2023 14:45	Text Document

A browser window shows the output of the script at localhost/readwritedir.php:

```
..
a.txt
New Text Document.txt
```

Prepared by Mrs Shah S. S.

What is a query string?

Imagine you have a URL:

`http://www.mywebsite.com/page.php?id=5&name=php`

Then the query string is:

`?id=5&name=php`

In this case, the query consists of two parts: a key `id` with value `5`, and a key `name` with value `php`.

You can access the value of the query string keys using this code:

PHP

```
$id = $_GET['id'];
```

The parameters from a URL string can be retrieved in PHP using `parse_url()` and `parse_str()` functions.

Note: Page URL and the parameters are separated by the `?` character.

parse_url() Function: The `parse_url()` function is used to return the components of a URL by parsing it. It parse an URL and return an associative array which contains its various components.

Syntax:

```
parse_url( $url, $component = -1 )
```

parse_str() Function: The `parse_str()` function is used to parse a query string into variables. The string passed to this function for parsing is in the format of a query string passed via a URL.

Syntax:

```
parse_str( $string, $array )
```

- Below examples uses parse_url() and parse_str() function to get the parameters from URL string.
- **Example 1:**

```
<?php

// Initialize URL to the variable
$url = 'https://www.geeksforgeeks.org?name=Tony';

// Use parse_url() function to parse the URL
// and return an associative array which
// contains its various components
$url_components = parse_url($url);

// Use parse_str() function to parse the
// string passed via URL
parse_str($url_components['query'], $params);

// Display result
echo ' Hi '.$params['name'];

?>
```

Output: Hi Tony


```
<?php

// Initialize URL to the variable
$url =
'https://www.geeksforgeeks.org/register?name=Amit&email=amit1998@gmail.com';

// Use parse_url() function to parse the URL
// and return an associative array which
// contains its various components
$url_components = parse_url($url);

// Use parse_str() function to parse the
// string passed via URL
parse_str($url_components['query'], $params);

// Display result
echo ' Hi '.$params['name'].' your emailID is
'.$params['email'];

?>
```

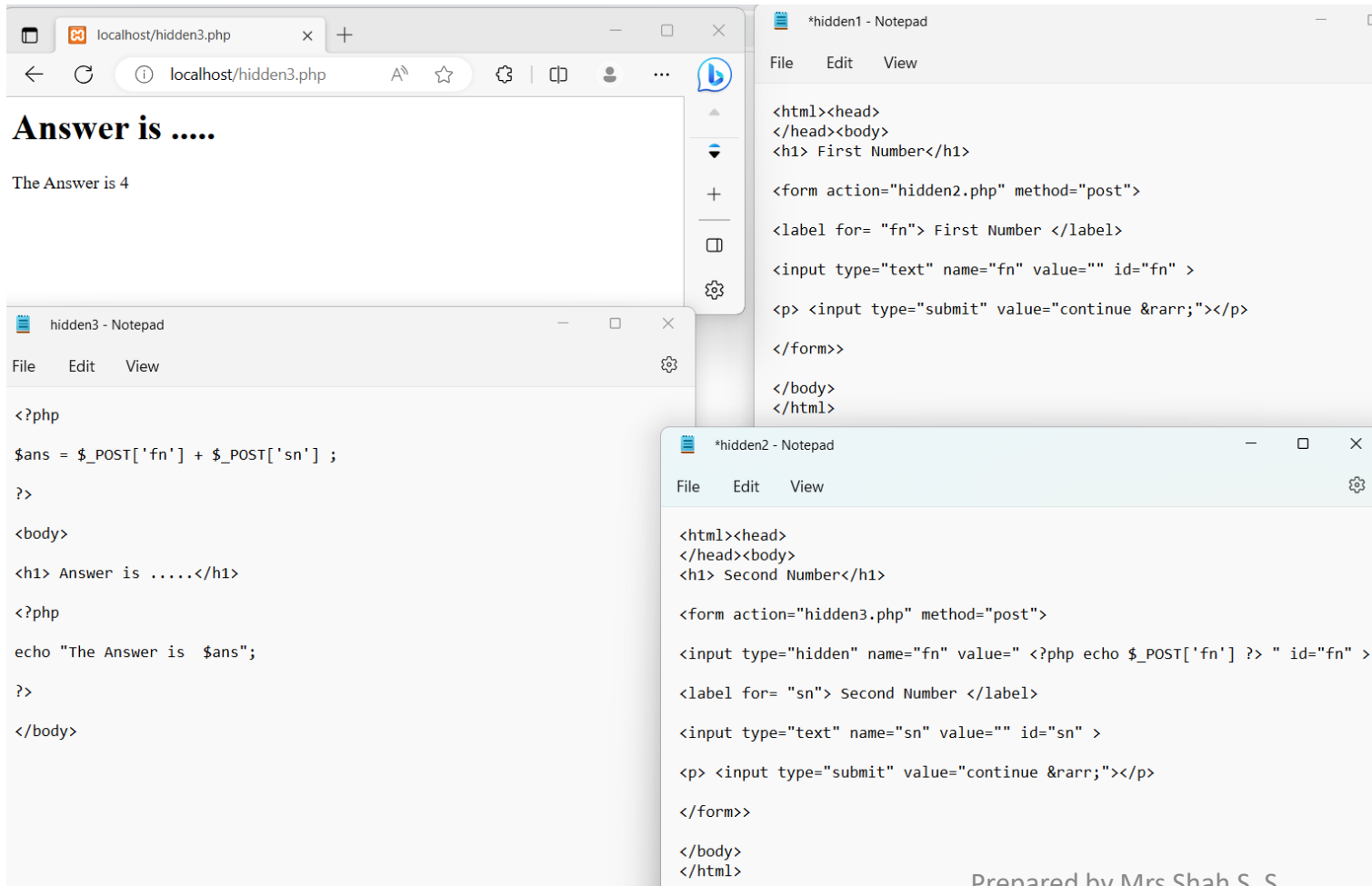
Output: Hi Amit your emailID is amit1998@gmail.com

Hidden field

Using Hidden Fields to Save State in PHP

- Use a hidden field to keep track of this value.
- A hidden field behaves the same as a text field, except that the user cannot see it unless he views the HTML source of the document that contains it.

Saving State with a Hidden Field



The screenshot shows a web browser window at localhost/hidden3.php displaying "Answer is" and "The Answer is 4". Below it are three Notepad windows showing the PHP code for three files: hidden1.php, hidden2.php, and hidden3.php.

```
*hidden1 - Notepad
<html><head>
</head><body>
<h1> First Number</h1>

<form action="hidden2.php" method="post">

<label for= "fn"> First Number </label>

<input type="text" name="fn" value="" id="fn" >

<p> <input type="submit" value="continue &arr;"></p>

</form>>

</body>
</html>
```

```
*hidden2 - Notepad
<html><head>
</head><body>
<h1> Second Number</h1>

<form action="hidden3.php" method="post">

<input type="hidden" name="fn" value=" <?php echo $_POST['fn'] ?> " id="fn" >

<label for= "sn"> Second Number </label>

<input type="text" name="sn" value="" id="sn" >

<p> <input type="submit" value="continue &arr;"></p>

</form>>

</body>
</html>
```

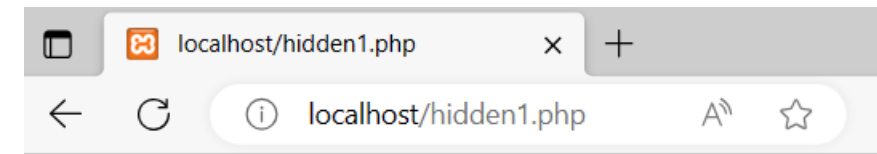
```
hidden3 - Notepad
<?php
$ans = $_POST['fn'] + $_POST['sn'] ;
?>

<body>

<h1> Answer is .....</h1>

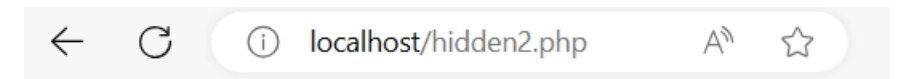
<?php
echo "The Answer is  $ans";
?>

</body>
```



First Number

First Number



Second Number

Second Number

PHP Cookies

- **What are Cookies?**
- A cookie is a small file with the maximum size of 4KB that the web server stores on the client computer. They are typically used to keeping track of information such as a username that the site can retrieve to personalize the page when the user visits the website next time. A cookie can only be read from the domain that it has been issued from. Cookies are usually set in an HTTP header but JavaScript can also set a cookie directly on a browser.
- **Setting Cookie In PHP:**

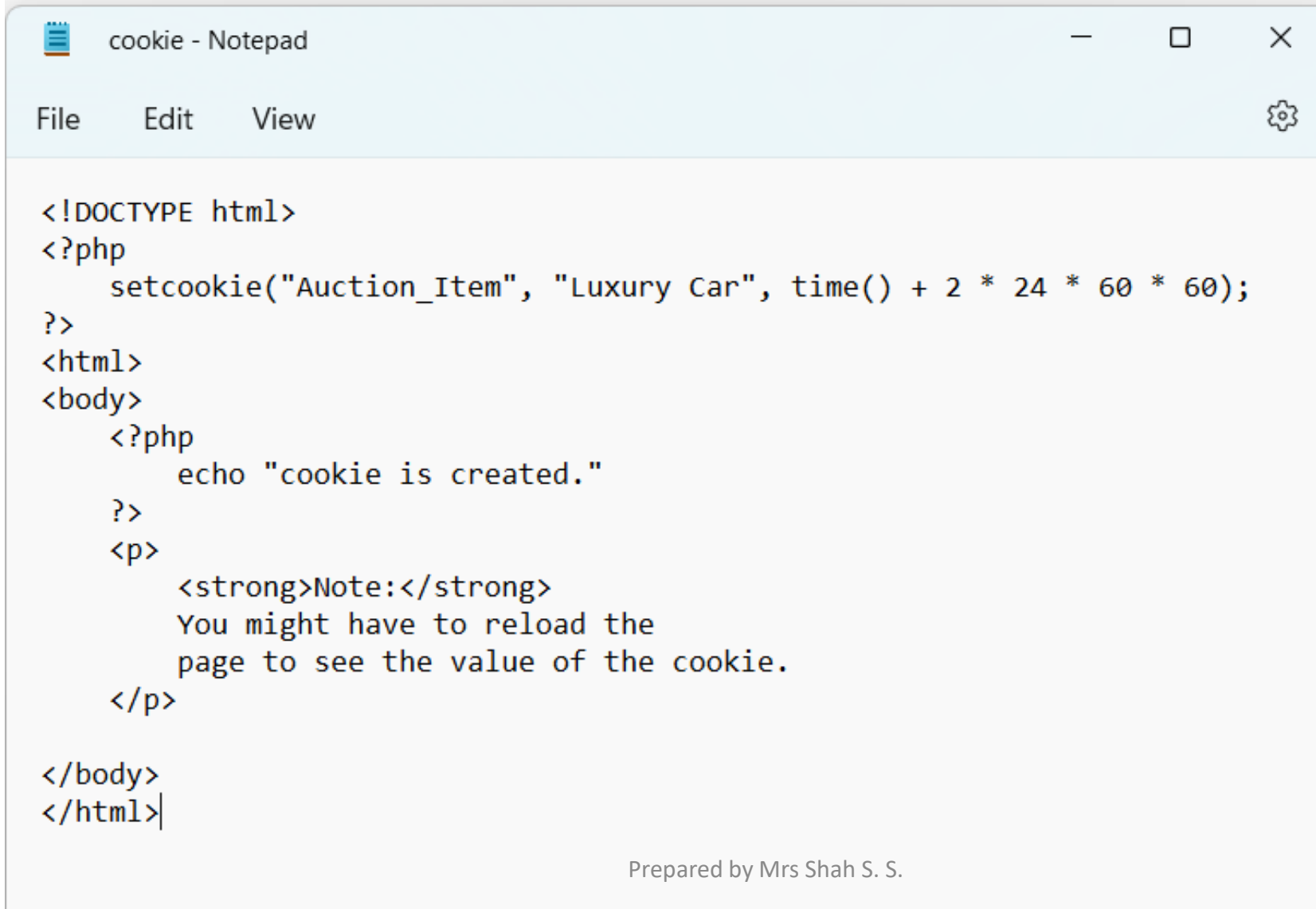
Syntax :

```
setcookie(name, value, expire, path, domain, security);
```

- Parameters: The setcookie() function requires six arguments in general which are:
- Name: It is used to set the name of the cookie.
- Value: It is used to set the value of the cookie.
- Expire: It is used to set the expiry timestamp of the cookie after which the cookie can't be accessed.
- Path: It is used to specify the path on the server for which the cookie will be available.
- Domain: It is used to specify the domain for which the cookie is available.
- Security: It is used to indicate that the cookie should be sent only if a secure HTTPS connection exists.
- Below are some operations that can be performed on Cookies in PHP:
- Creating Cookies: Creating a cookie named Auction_Item and assigning the value Luxury Car to it. The cookie will expire after 2 days($2 \text{ days} * 24 \text{ hours} * 60 \text{ mins} * 60 \text{ seconds}$).

cookie is created.

Note: You might have to reload the page to see the value of the cookie.

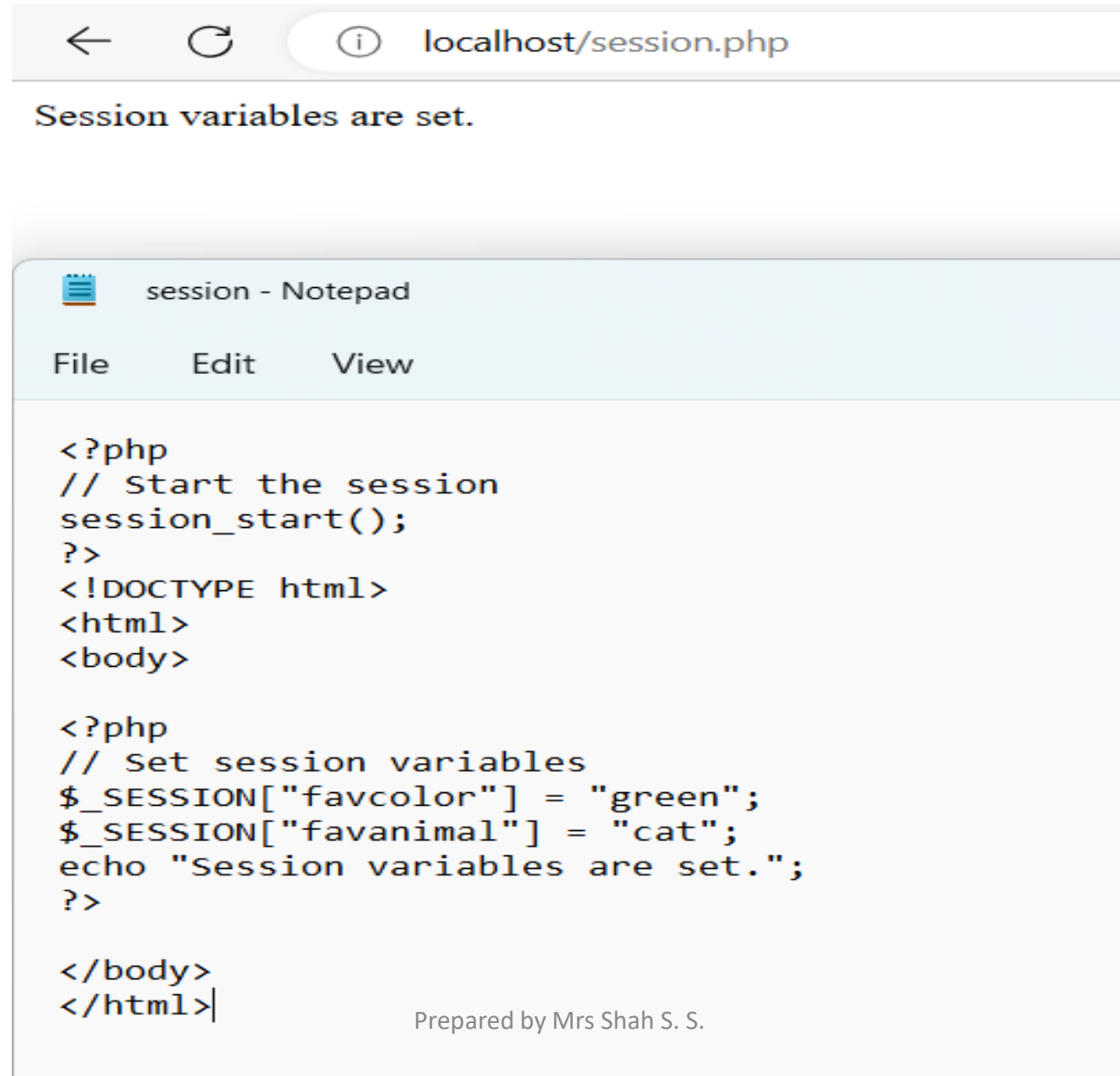


```
<!DOCTYPE html>
<?php
    setcookie("Auction_Item", "Luxury Car", time() + 2 * 24 * 60 * 60);
?>
<html>
<body>
    <?php
        echo "cookie is created."
    ?>
    <p>
        <strong>Note:</strong>
        You might have to reload the
        page to see the value of the cookie.
    </p>

</body>
</html>
```

- What is a PHP Session?
- When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.
- So; Session variables hold information about one single user, and are available to all pages in one application.

- we start a new PHP session and set some session variables



The image shows a web browser window at the top with the address bar displaying 'localhost/session.php'. The page content says 'Session variables are set.' Below the browser is a Notepad window titled 'session - Notepad' with a menu bar containing 'File', 'Edit', and 'View'. The Notepad contains the following PHP code:

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

Prepared by Mrs Shah S. S.

PHP Form Handling

The PHP superglobals `$_GET` and `$_POST` are used to collect form-data.

Superglobal-which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

Post method

```
postphp - Notepad
File Edit View

<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

```
welcome - Notepad
File Edit View

<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>

localhost/welcome.php
```

Welcome ss
Your email address is: ss@gmail.com

For execution-<http://localhost/postphp.php>

Get method

```
getphp - Notepad
File Edit View

<html>
<body>

<form action="welcome.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

```
welcome - Notepad
File Edit View

<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"]; ?>

</body>
</html>
```

← → ↺ ⓘ localhost/welcome.php?name=a&email=aa%40gmail.com

Welcome a
Your email address is: aa@gmail.com

GET vs. POST

Both GET and POST create an array (e.g. `array(key1 => value1, key2 => value2, key3 => value3, ...)`). This array holds key/value pairs, where keys are the names of the form controls and values are the input data from the user.

Both GET and POST are treated as `$_GET` and `$_POST`. These are superglobals, which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.

`$_GET` is an array of variables passed to the current script via the URL parameters.

`$_POST` is an array of variables passed to the current script via the HTTP POST method.

When to use GET?

Information sent from a form with the GET method is **visible to everyone** (all variable names and values are displayed in the URL). GET also has limits on the amount of information to send. The limitation is about 2000 characters. However, because the variables are displayed in the URL, it is possible to bookmark the page. This can be useful in some cases.

GET may be used for sending non-sensitive data.

Note: GET should NEVER be used for sending passwords or other sensitive information!

When to use POST?

Information sent from a form with the POST method is **invisible to others** (all names/values are embedded within the body of the HTTP request) and has **no limits** on the amount of information to send.

Moreover POST supports advanced functionality such as support for multi-part binary input while uploading files to server.

However, because the variables are not displayed in the URL, it is not possible to bookmark the page.

PHP Form Validation

The validation rules for the form above are as follows:

PHP Form Validation Example

*** required field**

Name: *

E-mail: *

Website:

Comment:

Gender: ☐ Female ☐ Male ☐ Other *

Your Input:

Field	Validation Rules
Name	Required. + Must only contain letters and whitespace
E-mail	Required. + Must contain a valid email address (with @ and .)
Website	Optional. If present, it must contain a valid URL
Comment	Optional. Multi-line input field (textarea)
Gender	Required. Must select one

Text Fields

The name, email, and website fields are text input elements, and the comment field is a textarea. The HTML code looks like this:

Name: `<input type="text" name="name">`

E-mail: `<input type="text" name="email">`

Website: `<input type="text" name="website">`

Comment: `<textarea name="comment" rows="5" cols="40"></textarea>`

Radio Buttons

The gender fields are radio buttons and the HTML code looks like this:

Gender:

`<input type="radio" name="gender" value="female">Female`

`<input type="radio" name="gender" value="male">Male`

`<input type="radio" name="gender" value="other">Other`

The Form Element

The HTML code of the form looks like this:

`<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">`

When the form is submitted, the form data is sent with method="post".

the `$_SERVER["PHP_SELF"]` sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form.

What is the `htmlspecialchars()` function?

The `htmlspecialchars()` function converts special characters to HTML entities. This means that it will replace HTML characters like `<` and `>` with `<` and `>`.

```

<!DOCTYPE HTML> <html><head></head><body>
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $name = test_input($_POST["name"]);
    $email = test_input($_POST["email"]);
    $website = test_input($_POST["website"]);
    $comment = test_input($_POST["comment"]);
    $gender = test_input($_POST["gender"]);
}
function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
    Name: <input type="text" name="name"> <br><br>
    E-mail: <input type="text" name="email"> <br><br>
    Website: <input type="text" name="website"> <br><br>
    Comment: <textarea name="comment" rows="5" cols="40"></textarea> <br><br>
    Gender: <input type="radio" name="gender" value="female">Female
           <input type="radio" name="gender" value="male">Male
           <input type="radio" name="gender" value="other">Other <br><br>
           <input type="submit" name="submit" value="Submit">
</form>
<?php
echo "<h2>Your Input:</h2>";
echo $name;echo "<br>";
echo $email;echo "<br>";
echo $website;echo "<br>";
echo $comment;echo "<br>";
echo $gender;?>
</body></html>

```

PHP Form Validation Example

Name: E-mail: Website: Comment: Gender: ☐ Female ☐ Male ☐ Other

Your Input:

Shah
 shah@gmail.com
 google
 MCA-I-Web Technology
 female



File Edit View

```

<!DOCTYPE HTML>
<html><head><style>.error {color: #FF0000;}</style></head>
<body>
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = test_input($_POST["name"]);
    }

    if (empty($_POST["email"])) {
        $emailErr = "Email is required";
    } else {
        $email = test_input($_POST["email"]);
    }

    if (empty($_POST["website"])) {
        $website = "";
    } else {
        $website = test_input($_POST["website"]);
    }

    if (empty($_POST["comment"])) {
        $comment = "";
    } else {
        $comment = test_input($_POST["comment"]);
    }

    if (empty($_POST["gender"])) {
        $genderErr = "Gender is required";
    } else {
        $gender = test_input($_POST["gender"]);
    }
}
}

```

```

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>

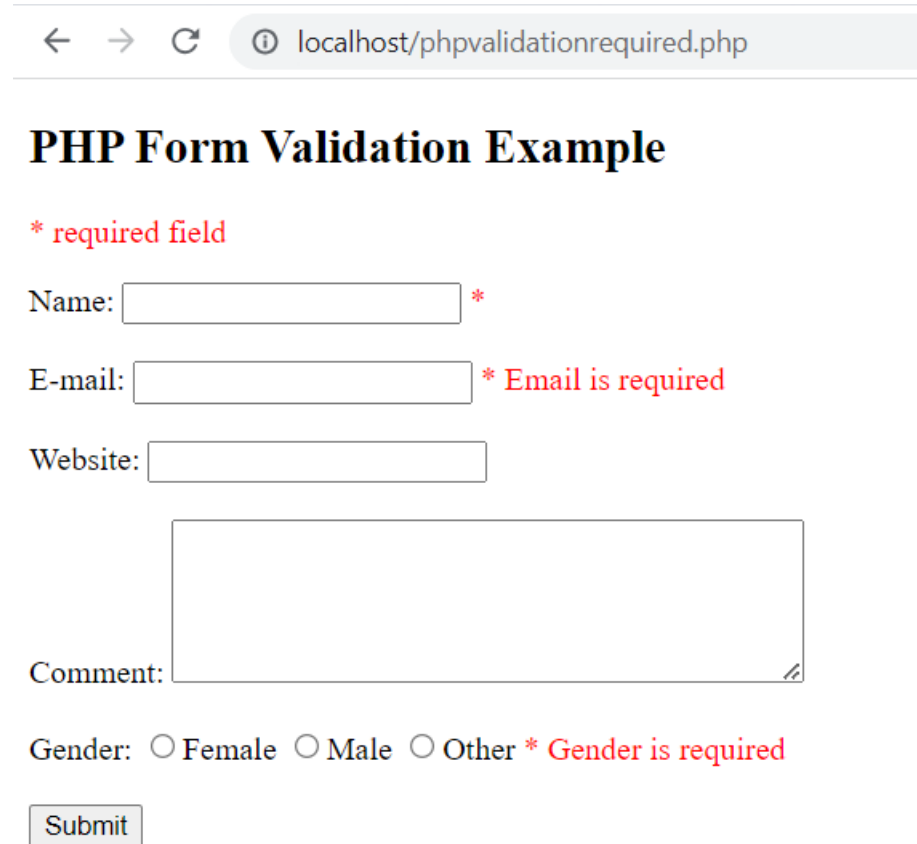
```

```

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
    Name: <input type="text" name="name">
    <span class="error">* <?php echo $nameErr;?></span>
    <br><br>
    E-mail: <input type="text" name="email">
    <span class="error">* <?php echo $emailErr;?></span>
    <br><br>
    Website: <input type="text" name="website">
    <span class="error"><?php echo $websiteErr;?></span>
    <br><br>
    Comment: <textarea name="comment" rows="5" cols="40"></textarea>
    <br><br>
    Gender:
    <input type="radio" name="gender" value="female">Female
    <input type="radio" name="gender" value="male">Male
    <input type="radio" name="gender" value="other">Other
    <span class="error">* <?php echo $genderErr;?></span>
    <br><br>
    <input type="submit" name="submit" value="Submit">
</form>
<?php
echo "<h2>Your Input:</h2>";
echo $name;echo "<br>";
echo $email;echo "<br>";
echo $website;echo "<br>";
echo $comment;echo "<br>";
echo $gender;
?></body></html>

```

some new variables: \$nameErr, \$emailErr, \$genderErr, and \$websiteErr. These error variables will hold error messages for the required fields. We have also added an **if else** statement for each \$_POST variable. This checks if the \$_POST variable is empty (with the PHP **empty()** function). If it is empty, an error message is stored in the different error variables, and if it is not empty, it sends the user input data through the **test_input()** function:



← → ↻ ⓘ localhost/phpvalidationrequired.php

PHP Form Validation Example

* required field

Name: *

E-mail: * Email is required

Website:

Comment:

Gender: ☐ Female ☐ Male ☐ Other * Gender is required

Your Input:

PHP Forms - Validate E-mail and URL

PHP - Validate Name

The code below shows a simple way to check if the name field only contains letters, dashes, apostrophes and whitespaces. If the value of the name field is not valid, then store an error message:

```
$name = test_input($_POST["name"]);  
if (!preg_match("/^[a-zA-Z-' ]*$/", $name)) {  
    $nameErr = "Only letters and white space allowed";  
}
```

The [preg_match\(\)](#) function searches a string for pattern, returning true if the pattern exists, and false otherwise.

PHP - Validate E-mail

The easiest and safest way to check whether an email address is well-formed is to use PHP's `filter_var()` function.

In the code below, if the e-mail address is not well-formed, then store an error message:

```
$email = test_input($_POST["email"]);  
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {  
    $emailErr = "Invalid email format";  
}
```


PHP - Validate URL

The code below shows a way to check if a URL address syntax is valid (this regular expression also allows dashes in the URL). If the URL address syntax is not valid, then store an error message:

```
$website = test_input($_POST["website"]);
if
(!preg_match("/\b(?:(:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?=_|!:,.;]*[-a-z0-9+&@#\/%?=_|]/i",$website)) {
    $websiteErr = "Invalid URL";
}
```

← → ↻ ⓘ localhost/phpvalidationemailurl.php

PHP Form Validation Example

* required field

Name: * Only letters and white space allowed

E-mail: * Invalid email format

Website: Invalid URL

Comment:

Gender: ☐ Female ☐ Male ☐ Other * Gender is required

Your Input:

@
aa
google

```

<!DOCTYPE HTML> <html><head><style>.error {color: #FF0000;}</style></head><body>
<?php// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = test_input($_POST["name"]);
        // check if name only contains letters and whitespace
        if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
            $nameErr = "Only letters and white space allowed";
        }
    }
    if (empty($_POST["email"])) {
        $emailErr = "Email is required";
    } else {
        $email = test_input($_POST["email"]);
        // check if e-mail address is well-formed
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $emailErr = "Invalid email format";
        }
    }
    if (empty($_POST["website"])) {
        $website = "";
    } else {
        $website = test_input($_POST["website"]);
        // check if URL address syntax is valid
        if (!preg_match("/\b(?:(:?https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?~_!|:,.;]*[-a-z0-9+&@#\/%?~_]/i",$website)) {
            $websiteErr = "Invalid URL";
        }
    }
    if (empty($_POST["comment"])) {
        $comment = "";
    } else {
        $comment = test_input($_POST["comment"]);
    }
    if (empty($_POST["gender"])) {
        $genderErr = "Gender is required";
    } else {
        $gender = test_input($_POST["gender"]);
    }
}
}

```

```

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
    Name: <input type="text" name="name">
    <span class="error">* <?php echo $nameErr;?></span>
    <br><br>
    E-mail: <input type="text" name="email">
    <span class="error">* <?php echo $emailErr;?></span>
    <br><br>
    Website: <input type="text" name="website">
    <span class="error"><?php echo $websiteErr;?></span>
    <br><br>
    Comment: <textarea name="comment" rows="5" cols="40"></textarea>
    <br><br>
    Gender:
    <input type="radio" name="gender" value="female">Female
    <input type="radio" name="gender" value="male">Male
    <input type="radio" name="gender" value="other">Other
    <span class="error">* <?php echo $genderErr;?></span>
    <br><br>
    <input type="submit" name="submit" value="Submit">
</form><?php
echo "<h2>Your Input:</h2>";
echo $name;echo "<br>";
echo $email;echo "<br>";
echo $website;echo "<br>";
echo $comment;echo "<br>";
echo $gender;
?></body></html>

```

PHP Complete Form Example

PHP - Keep The Values in The Form

To show the values in the input fields after the user hits the submit button, we add a little PHP script inside the value attribute of the following input fields: name, email, and website. In the comment textarea field, we put the script between the <textarea> and </textarea> tags. The little script outputs the value of the \$name, \$email, \$website, and \$comment variables. Then, we also need to show which radio button that was checked. For this, we must manipulate the checked attribute (not the value attribute for radio buttons):

← → ↻ ⓘ localhost/validationcomplete.php

PHP Form Validation Example

* required field

Name: *

E-mail: *

Website: Invalid URL

Comment:

Gender: ☒ Female ☐ Male ☐ Other *

Your Input:

ss
ss@gmail.com
google
web
female



File Edit View

```

<!DOCTYPE HTML>
<html>
<head>
<style>
.error {color: #FF0000;}
</style>
</head>
<body>

<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr = $websiteErr = "";
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (empty($_POST["name"])) {
        $nameErr = "Name is required";
    } else {
        $name = test_input($_POST["name"]);
        // check if name only contains letters and whitespace
        if (!preg_match("/^[a-zA-Z-' ]*$/",$name)) {
            $nameErr = "Only letters and white space allowed";
        }
    }

    if (empty($_POST["email"])) {
        $emailErr = "Email is required";
    } else {
        $email = test_input($_POST["email"]);
        // check if e-mail address is well-formed
        if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $emailErr = "Invalid email format";
        }
    }
}

```

```

if (empty($_POST["website"])) {
    $website = "";
} else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid (this regular expression also allows dashes in the URL)
    if (!preg_match("/\b(?:(:https?|ftp):\/\/|www\.)[-a-z0-9+&@#\/%?~_!|:,.;]*[-a-z0-9+&@#\/%?~_]/i",$website)) {
        $websiteErr = "Invalid URL";
    }
}

if (empty($_POST["comment"])) {
    $comment = "";
} else {
    $comment = test_input($_POST["comment"]);
}

if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
} else {
    $gender = test_input($_POST["gender"]);
}

function test_input($data) {
    $data = trim($data);
    $data = stripslashes($data);
    $data = htmlspecialchars($data);
    return $data;
}
?>

```

```

<h2>PHP Form Validation Example</h2>
<p><span class="error">* required field</span></p>
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
    Name: <input type="text" name="name" value="<?php echo $name;?>">
    <span class="error">* <?php echo $nameErr;?></span>
    <br><br>
    E-mail: <input type="text" name="email" value="<?php echo $email;?>">
    <span class="error">* <?php echo $emailErr;?></span>
    <br><br>
    Website: <input type="text" name="website" value="<?php echo $website;?>">
    <span class="error"><?php echo $websiteErr;?></span>
    <br><br>
    Comment: <textarea name="comment" rows="5" cols="40"><?php echo $comment;?></textarea>
    <br><br>
    Gender:
    <input type="radio" name="gender" <?php if (isset($gender) && $gender=="female") echo "checked";?> value="female">Female
    <input type="radio" name="gender" <?php if (isset($gender) && $gender=="male") echo "checked";?> value="male">Male
    <input type="radio" name="gender" <?php if (isset($gender) && $gender=="other") echo "checked";?> value="other">Other
    <span class="error">* <?php echo $genderErr;?></span>
    <br><br>
    <input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;
echo "<br>";
echo $email;
echo "<br>";
echo $website;
echo "<br>";
echo $comment;
echo "<br>";
echo $gender;
?>
</body>
</html>

```