CIT – WEBSITE DEVELOPMENT AND DESIGNING

ALERT ! Mistakes are expected. If you find any, kindly inform me so that I can correct it.

PHP

The **PHP Hypertext Preprocessor (PHP)** is a server scripting language, and a powerful tool for making dynamic and interactive Web pages. It can **create, open, read, write, delete, and close files** on the server. It can **add, delete, and modify data** in your database. It can be used to **control user-access.**

PHP files can contain text, HTML, CSS, JavaScript, and PHP code.

GETTING STARTED

To start using PHP, you need to install a web server on your own PC, and then install PHP and MySQL.

Before moving towards the remote web server, install a local host webserver.

Localhost is a local webserver that has all the configurations of remote webserver. The websites made on local servers are only visible to you. Other users cannot access it.

When your website is completely ready, then you can deploy it to remote webserver and then your website can be visited by any user.

- Here we will install XAMPP software i.e., local web server.
- Now open the directory (C: ,D: or E:) where you have installed your XAMPP software.
- Click on the folder having name **xampp.**
- Click on **htdocs** folder.
- Now this is the place where all your PHP files must be placed.

Now all your PHP files must have an extension of **.php** and placed in **htdocs** folder.

Don't forget to start apache and MySQL in your XAMPP software.

SYNTAX

All the php code is written inside the following tag:

```
<?php

     //PHP code;

?>
```

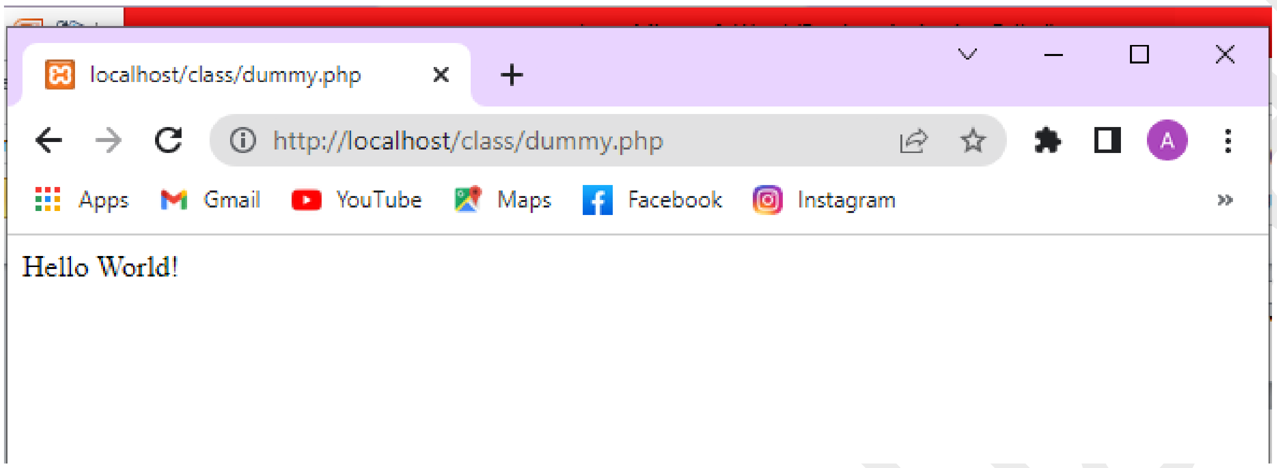
To check the output, open browser and write localhost/foldername/filename in the search bar.

Example

echo is used to output data on screen.

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

Output



Hello World!

COMMENTS IN PHP

// symbol is used to write **single line comment**.

# symbol is also used to write **single line comment**.

/* text or code */ is used to add **multi-line comment**. Any text or code written between these symbols will be considered as comment.

VARIABLES

In PHP, variable is declared using $ sign followed by variable name.

For example, `$name, $x, $age`

Rules for naming variables are same as that of JavaScript.

DATA TYPES

Integer:

Integer data type contains all positive and negative whole numbers.

For Example: 25, -456 etc.

```
$age=25;
```

Float:

Float data type is a number with decimal point. It can either be positive or negative.

For Example: 25.89, -456.509 etc.

```
$marks=89.5;
```

String:

A string is a combination of characters. In PHP any combination of characters written between quotation marks (either single or double) is considered as string.

For Example: "25", ' Ali ', "ahmad52" etc.

```
$Student_Name="Ali";
```

Array:

An array is a collection of similar data elements. In PHP arrays are written with keyword **`array`** followed by array elements written in small brackets `()`. Elements in array are separated by comma.

For Example:

```
$Student_Names = array ("Ahmad", "Ali", "Aslam");

$Marks= array (52.5, 82.0, 78.5, 90.75);
```

Boolean:

Boolean represents either of two values i.e. **true** or **false**.

For Example:

```
$is_raining=true;

$is_raining=false;
```

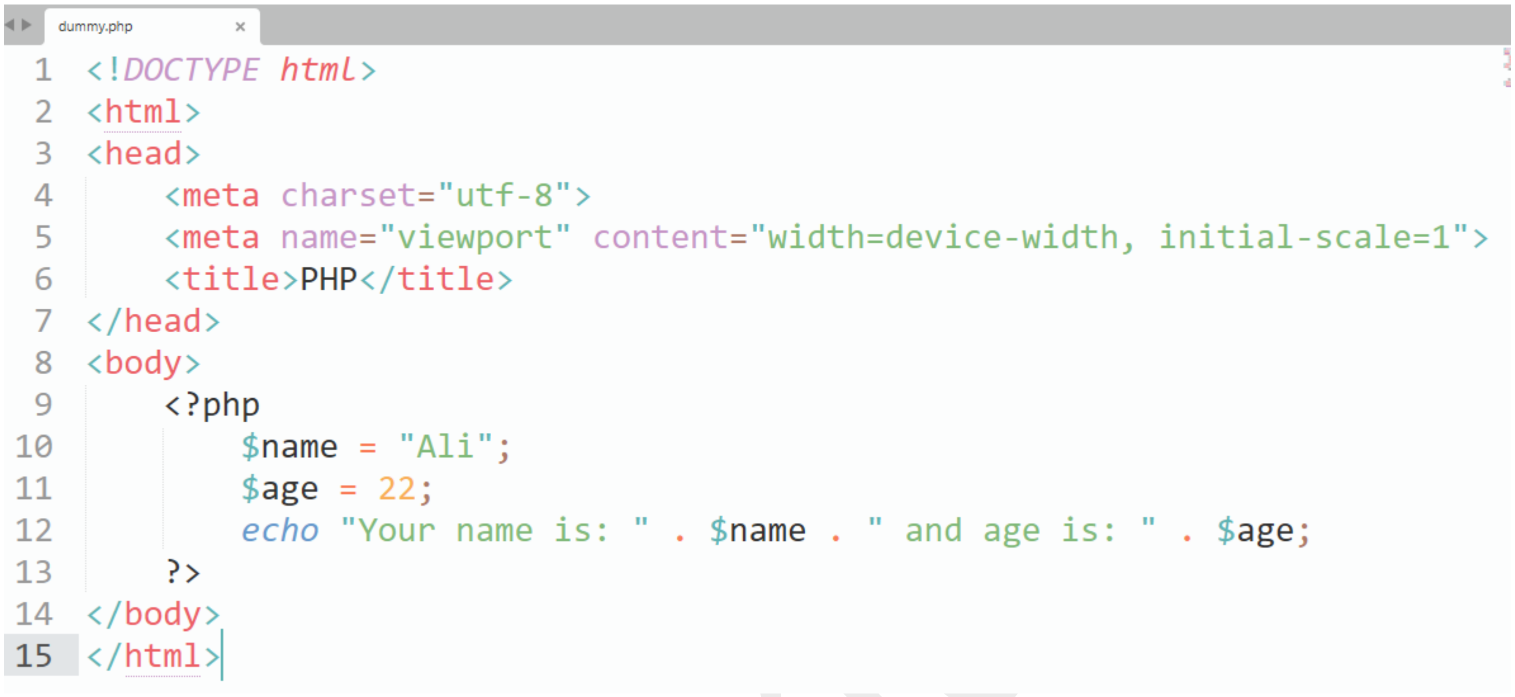
The values of variables can be displayed by using **`echo`**, **`var_dump($variable_name)`** or **`print_r($variable_name).`**

The **`var_dump($variable_name)`** displays variable data along with its data type.

CONCATENATION

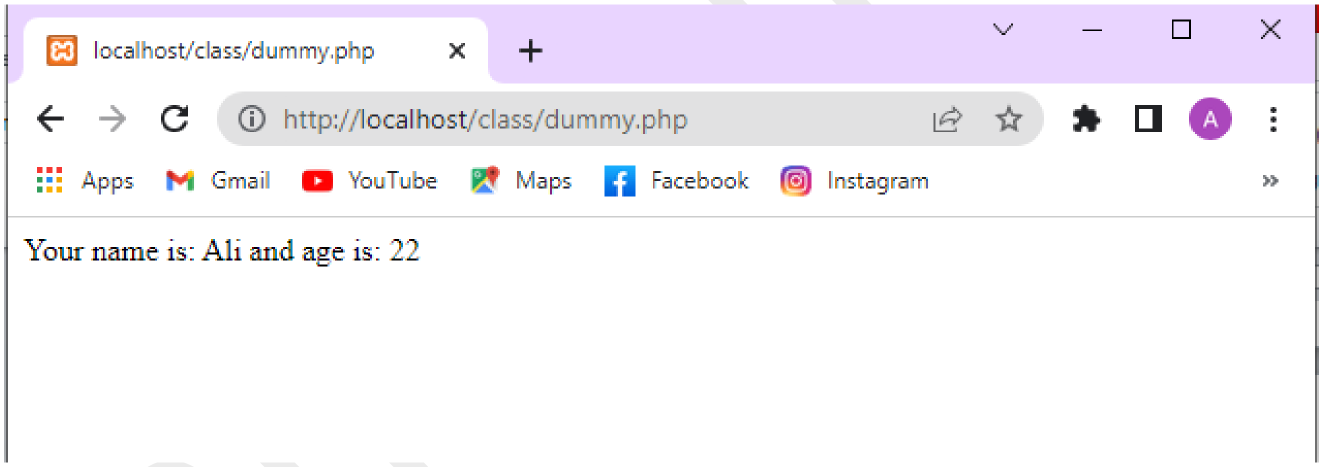
The dot (.) is used to concatenate multiple data items.

Example:



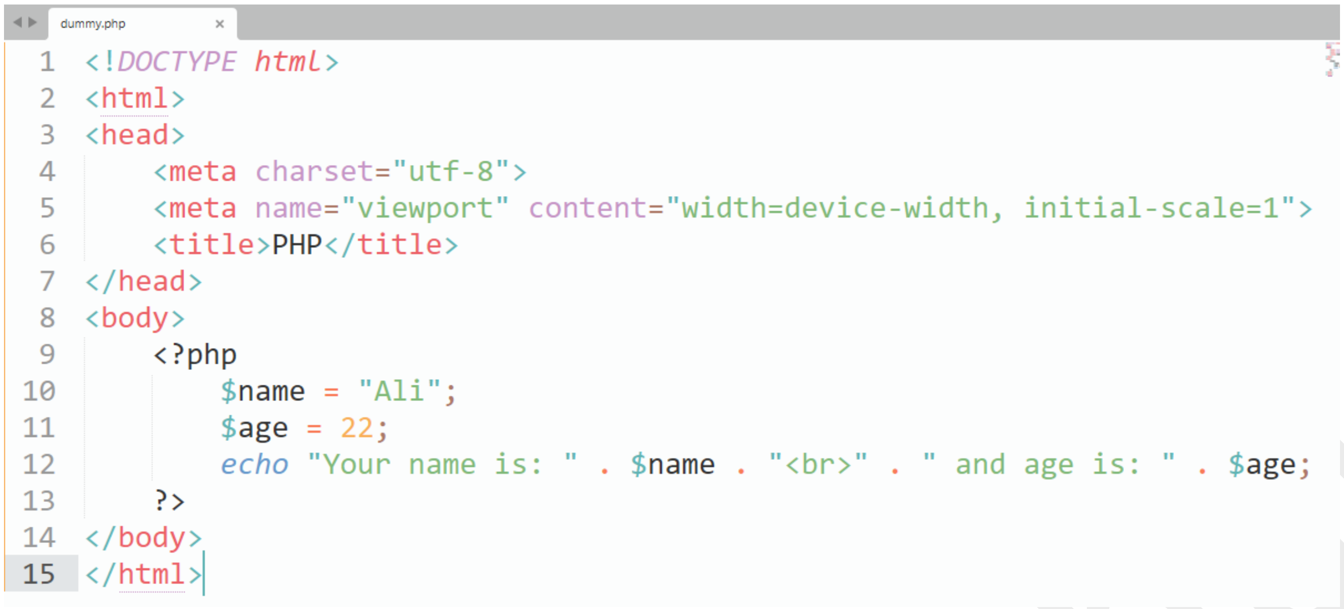
```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>PHP</title>
</head>
<body>
    <?php
        $name = "Ali";
        $age = 22;
        echo "Your name is: " . $name . " and age is: " . $age;
    ?>
</body>
</html>
```

Output:



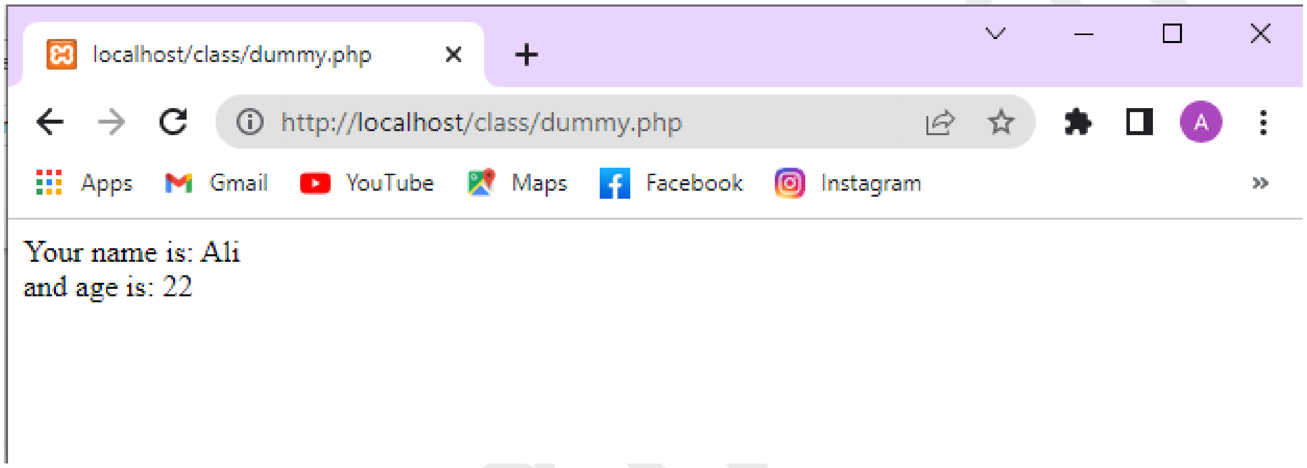
```
Your name is: Ali and age is: 22
```

You can also use HTML tags in PHP as follows:



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>PHP</title>
</head>
<body>
    <?php
        $name = "Ali";
        $age = 22;
        echo "Your name is: " . $name . "<br>" . " and age is: " . $age;
    ?>
</body>
</html>
```

Output:



```
Your name is: Ali
and age is: 22
```

The syntax and purpose of **Operators**, **conditional statements**, **Functions** and **Loops** are same as that of JavaScript.

But there is another loop in PHP as follows:

Foreach Loop

Loops through a block of code for each element in an array.

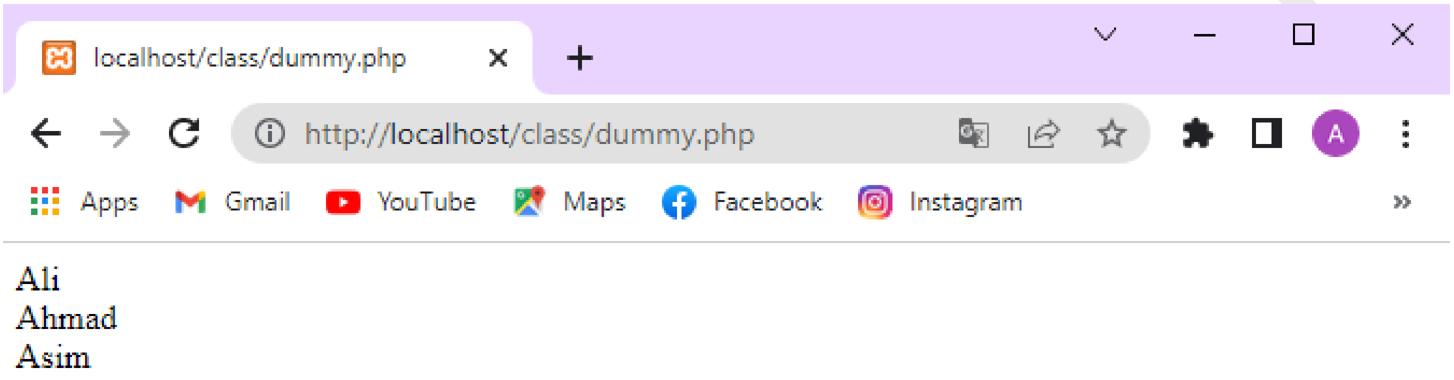
Syntax:

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

Example:

```
<?php
    $Student_Names = array("Ali", "Ahmad", "Asim");
    foreach ($Student_Names as $value) {
        echo "$value <br>";
    }
?>
```

Output:


localhost/class/dummy.php

http://localhost/class/dummy.php

```
Ali
Ahmad
Asim
```

PHP GET AND POST METHOD

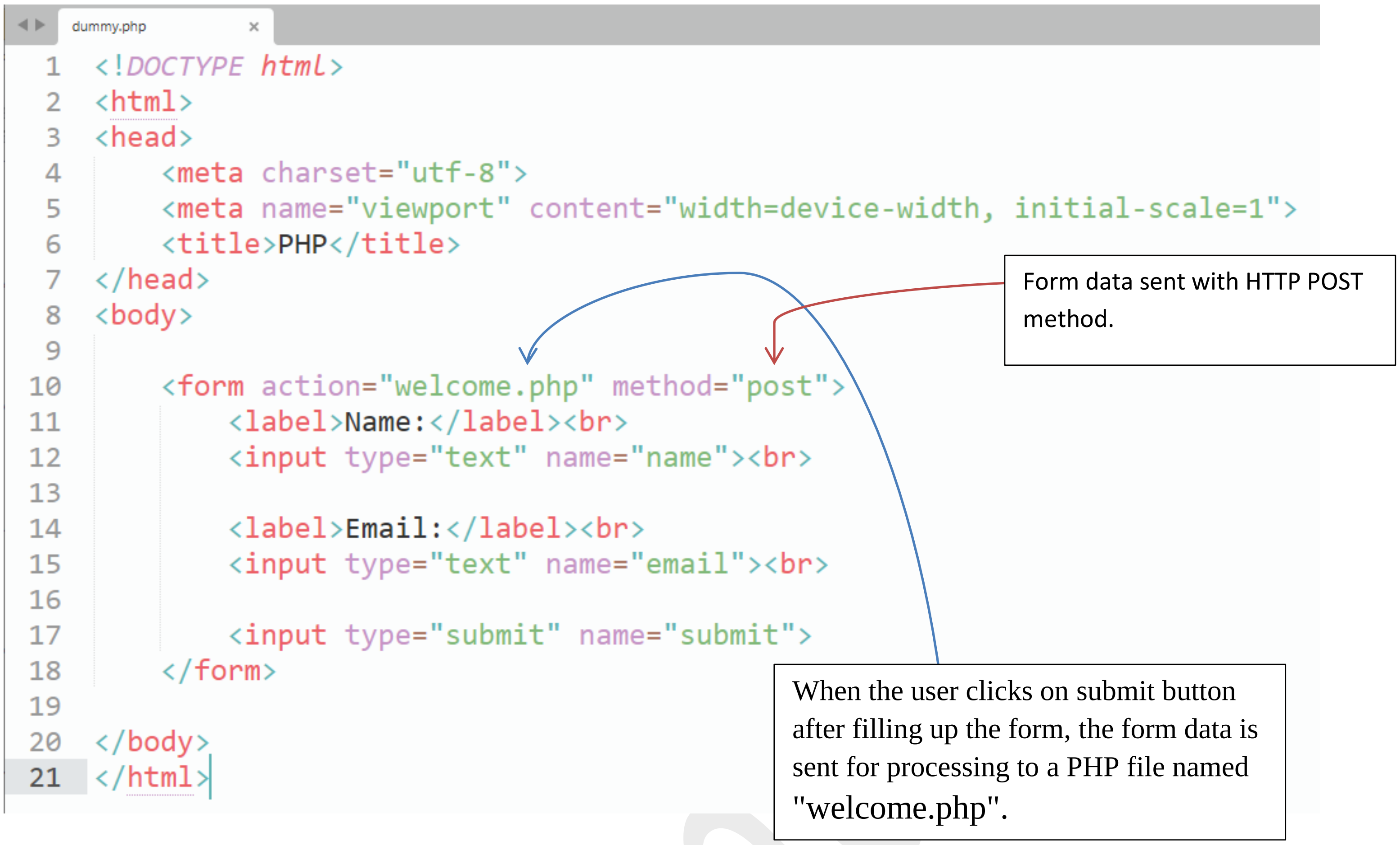
The GET and POST methods of PHP are used to collect form data.

$_GET is used to get values submitted via HTTP GET method.

$_POST is used to get values submitted via HTTP POST method.

Example:

Let's create a form in dummy.php file.

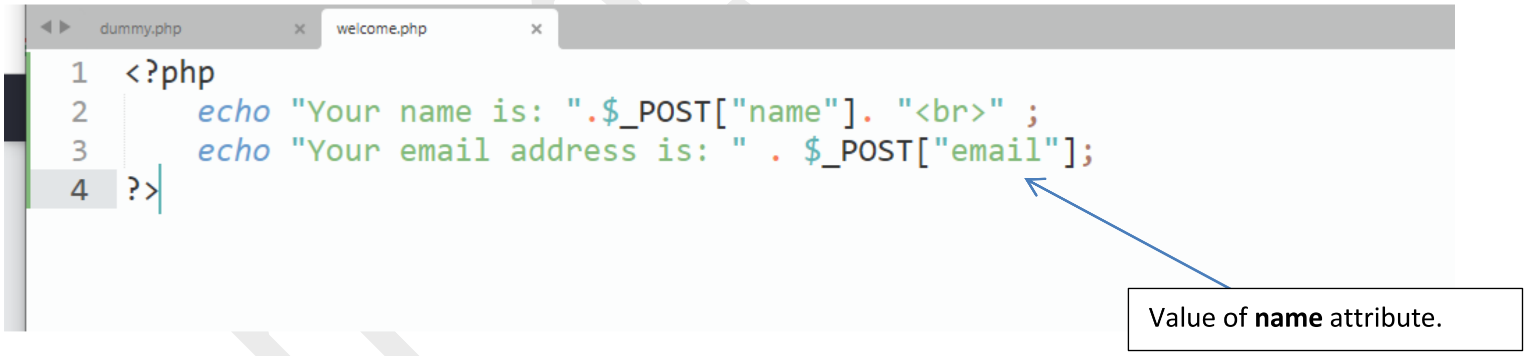

dummy.php

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>PHP</title>
</head>
<body>

    <form action="welcome.php" method="post">
        <label>Name:</label><br>
        <input type="text" name="name"><br>

        <label>Email:</label><br>
        <input type="text" name="email"><br>

        <input type="submit" name="submit">
    </form>

</body>
</html>
```

Form data sent with HTTP POST method.

When the user clicks on submit button after filling up the form, the form data is sent for processing to a PHP file named "welcome.php".

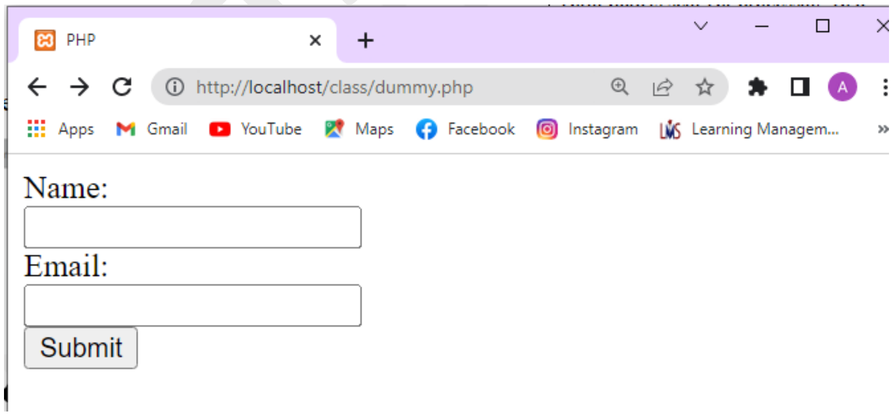
Now create a **welcome.php** file to display form data by getting values submitted via post method.


dummy.php welcome.php

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

Value of **name** attribute.

Output:

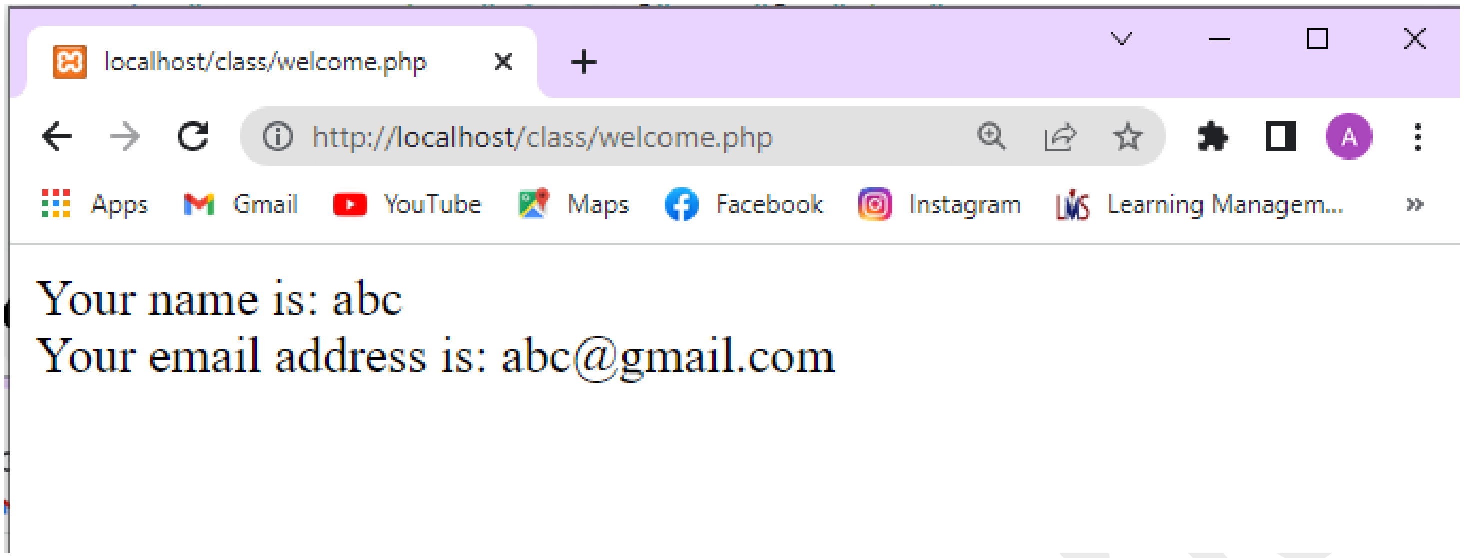

PHP
http://localhost/class/dummy.php
Apps Gmail YouTube Maps Facebook Instagram Learning Managem...

Name:

Email:

Submit

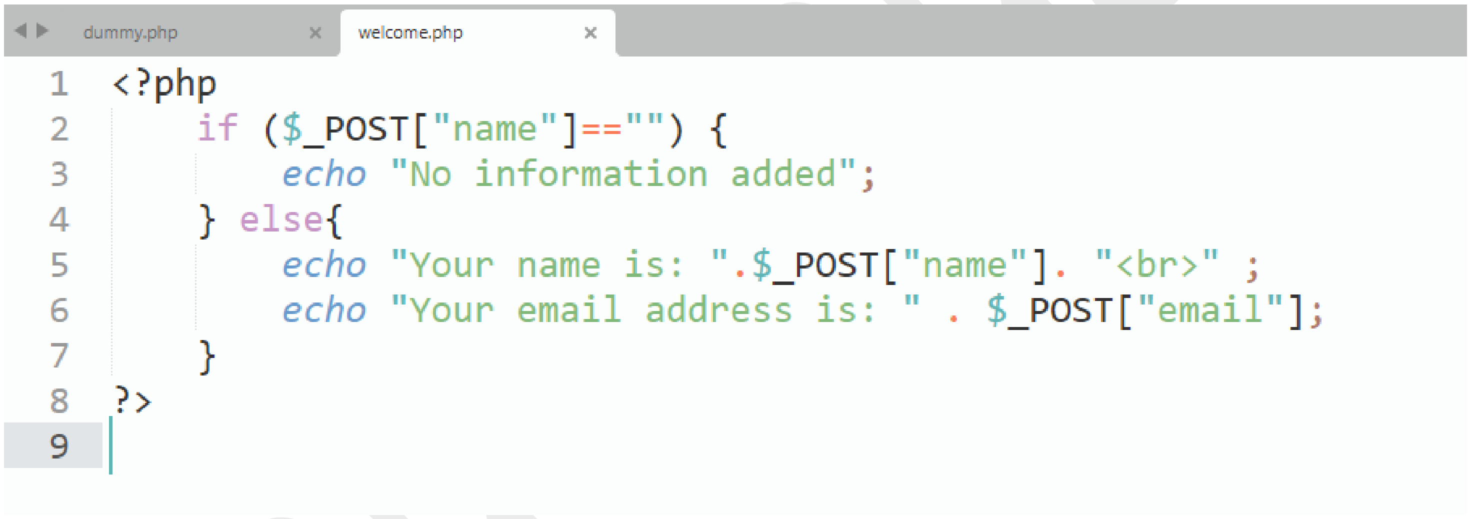
After filling up the form and clicking on submit button.


localhost/class/welcome.php
http://localhost/class/welcome.php
Apps
Gmail
YouTube
Maps
Facebook
Instagram
Learning Managem...

```
Your name is: abc
Your email address is: abc@gmail.com
```

What if the user does not fill up the form and submit form?

Let’s apply a basic validation



```php
<?php
    if ($_POST["name"]=="") {
        echo "No information added";
    } else{
        echo "Your name is: ".$_POST["name"]. "<br>" ;
        echo "Your email address is: " . $_POST["email"];
    }
?>

```

Now check the output.