PHP: **Web Applications**

CMPE 2550

Harsimranjot Aulakh

Introduction

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

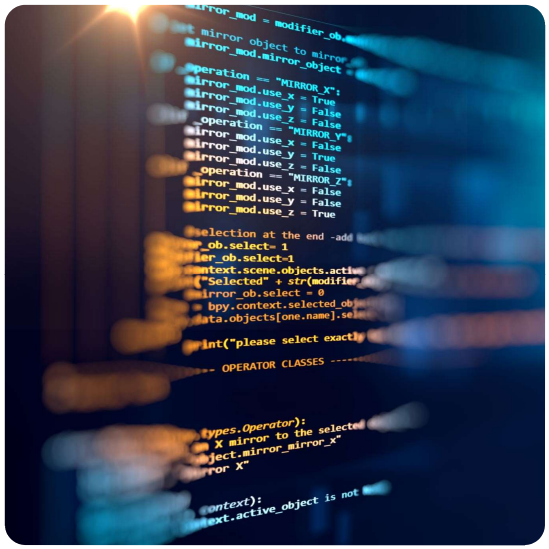
The most recent version of PHP is PHP 8.3, which was released on December 21, 2023

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"

PHP actions

- 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.

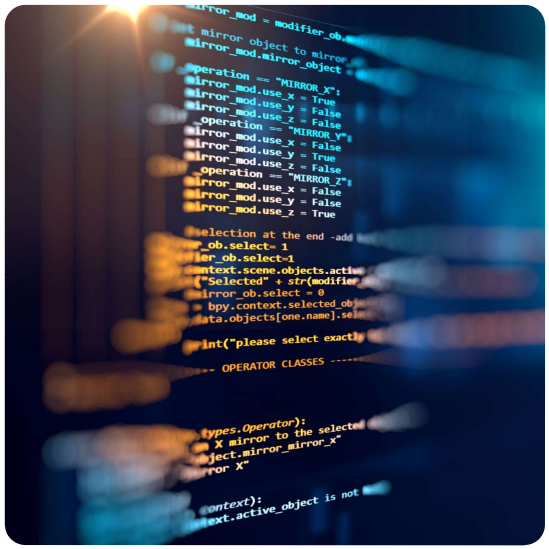


```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

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

</body>
```

Basic PHP Syntax

```
<?php

// PHP code goes here

?>
```

- The default file extension for PHP files is ".php"
- PHP statements end with a semicolon(;).
- PHP keywords are not case sensitive.
- Variable names are case sensitive.

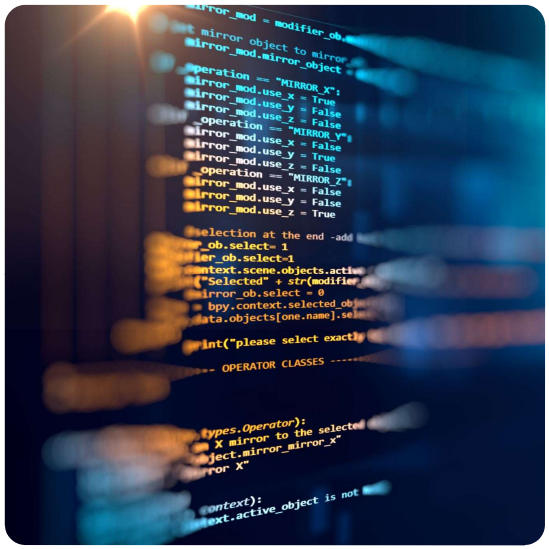


```php
<!DOCTYPE html>
<html>
<body>

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

</body>
</html>
```

Basic PHP

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

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

</body>
</html>
```

- PHP keywords are not case sensitive.



Basic PHP

```
<!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>
```

Variable names are case sensitive.

PHP Comments

Comments single line // or #

Comments multiline /* ... */

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

<?php

// This is a single-line comment

# This is also a single-line comment

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

</body>
</html>
```

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

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

```php
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

PHP Variables

- Starts with $ symbol.
- Name must start with letter or underscore(_).
- Can only contain letters or numbers.
- Are case-sensitive.
- Can be output using echo or print.
- PHP does not require data type to be specified.

PHP Variables Scope

- Variables with global scope can only be accessed outside a function

```php
<?php
$x = 5; // global scope

function myTest() {
    // using x inside this function will generate an error
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

PHP Variables Scope

- The global keyword can be used to access a global within a function

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}
```

PHP Variables Scope

- PHP stores all globals in an array called $GLOBALS[index] where index is the name of the variable

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

PHP Variables Scope

- Variables can be static. when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job. In this case we can use the static keyword to make it persist.

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

<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();   // Output 0

myTest();   // Output 1

myTest();   // Output 2
?>

</body>
</html>
```

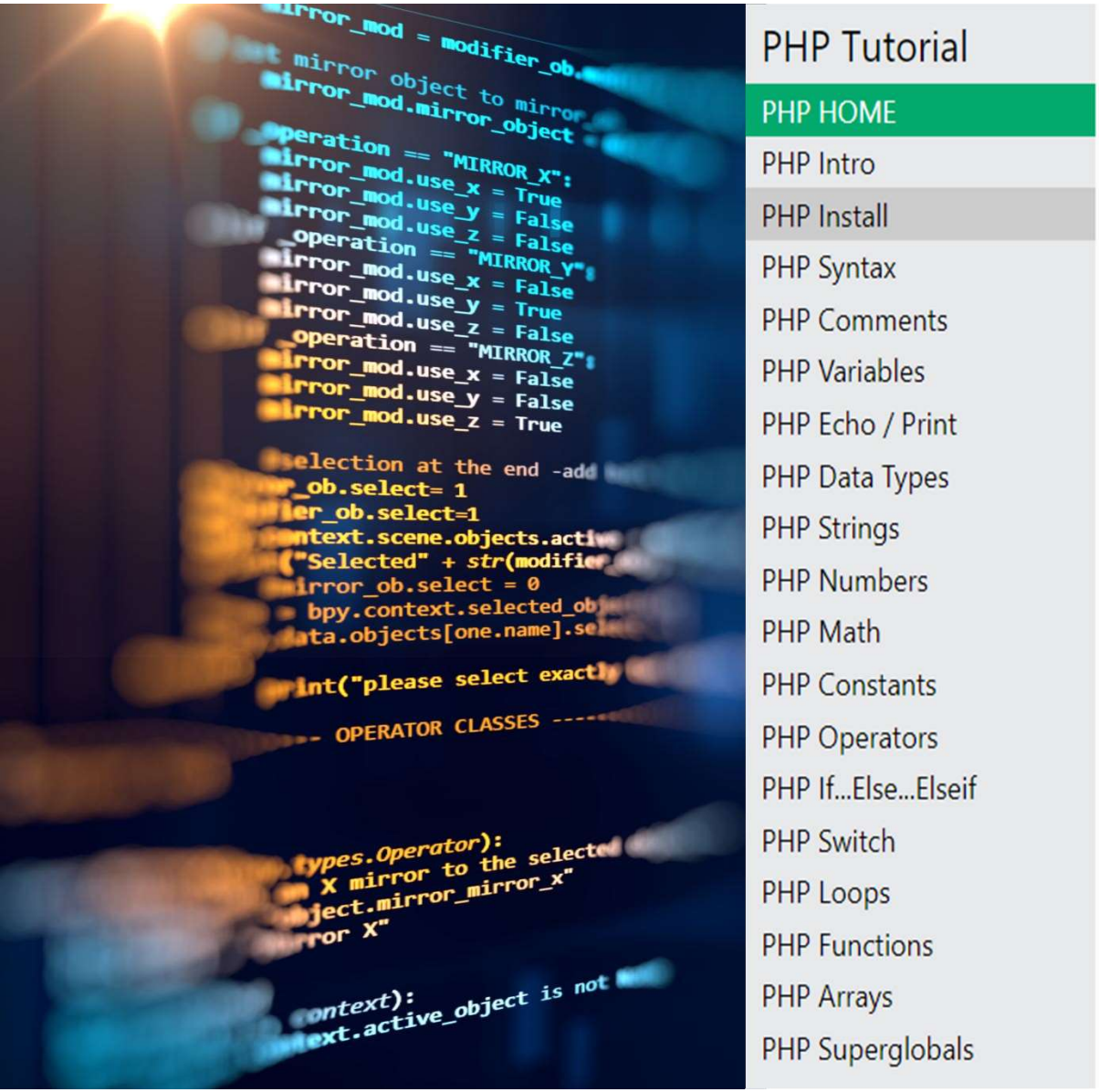
PHP Data Types

string – use single or double quotes
integer – 32 bit range, signed
float – any value with a decimal
boolean – true and false
array – format… $cars = array(“Toyota”, “Mazda”, “Honda”);
object
NULL – a variable with no value assigned to it
resource – used for a database call
var_dump($x) can be used to return a string with the datatype(s) and value(s)

PHP echo or print

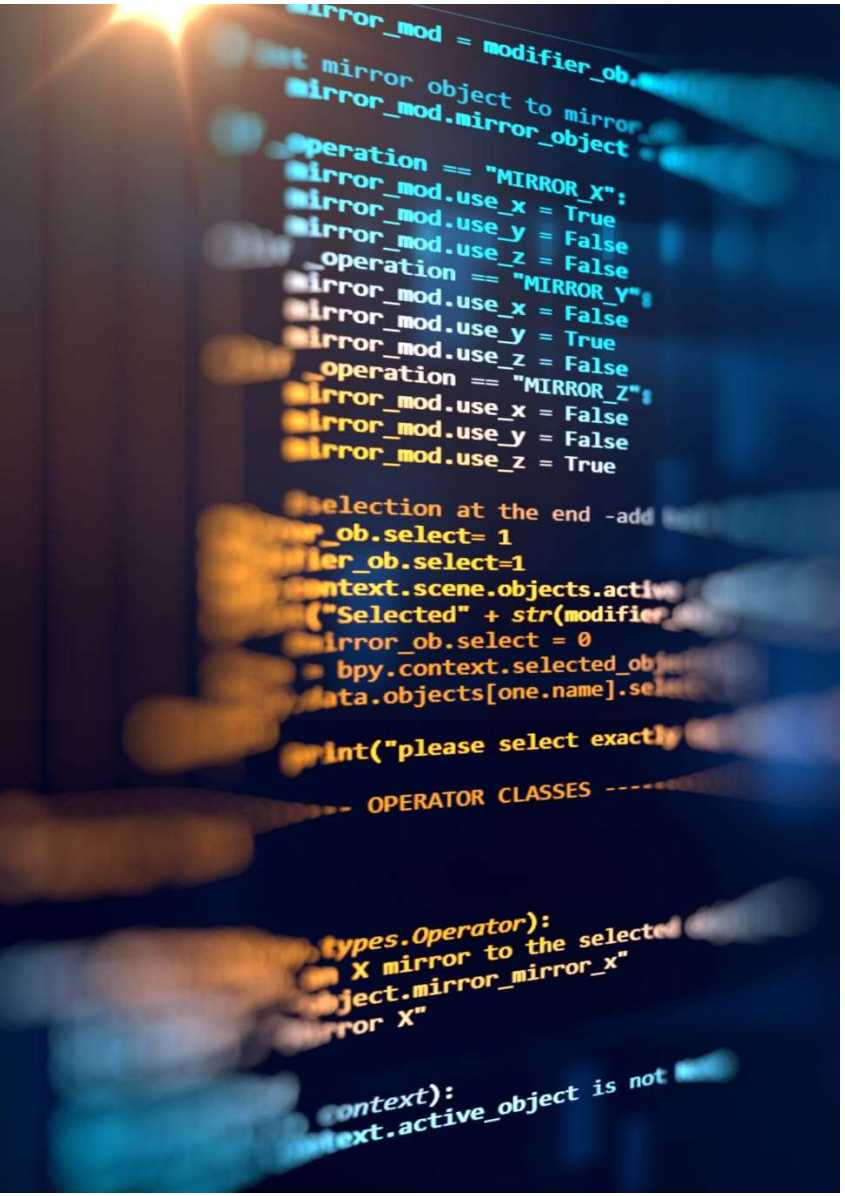
```php
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

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

- Can be used with optional ()
- echo has no return value and can take multiple parameters
- print has a return value of 1 and has only one parameter
- HTML tags can be within the string to be output



PHP Basic Stuff

- Take time and review basic stuff from w3schools



PHP Basic Stuff Demo

- Demonstrate how to use basic things with demo.

DEMO 01

PHP Basic Stuff Demo 01

- Php stuff can be included directly in html file.

```
<header>
    <h1> CMPE2550 A04 - Demo 01 PHP Introduction </h1>
</header>
<main>
    <?php
        // Printing the first line of code with PHP
        # Another way to include single line comment in the code.

        // Inline -style of adding php directly inside HTML code

        echo " This is my PHP stuff.";
        /*
            This is example of including multiline comment
        */

        echo '<div style= "color:red";>
                    <p> This is first paragraph on my web page.</p>

                </div>'
    ?>
```

Direct php in html

PHP Basic Stuff Demo 01

- Using php variables in html file.

```
<div>
    <?php
        $name="Simran";
        // One way of including variables directly inside a string
        echo "Your name is $name";
        // Alternative way of doing the same
        echo "<br/>Your name is ". $name." welcome to our page.";
        echo "<br/>Your name is ". UserData()." welcome to our page.";
    ?>
</div>
```

Declaring and initializing PHP variable. $ sign is used in front of PHP variables

PHP variables in html

Using concatenation operator

Including PHP function call. UserData() function definition is on next slide.

PHP Basic Stuff Demo 01

- Using php variables in html file.

```php
// Declaring variable to store username
$name="Simran";  // Global scope

2 references
function UserData()
{
    //global $name;

    // returning the value of name variable
    //return $name;

    // Alternative way of accessing global variables inside the function

    return $GLOBALS['name'];
}
```

The `global` keyword is used to access a global variable from within a function.

PHP also stores all global variables in an array called `$GLOBALS[index]`. The *index* holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

PHP Basic Stuff Demo 01

- Using php stuff at the top of the page.

```php
<?php
    // Declaring variable to store username
    //including PHP stuff at the top of the page. Uncomment it to test it.
    $name="Simran";

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title><?php echo "Demo 01- Intro to PHP"; ?>  </title>
</head>
<body>
    <header>
        <h1> CMPE2550 A04 - Demo 01 PHP Introduction </h1>
    </header>
    <main>
        <div>
            <?php
                // One way of including variables directly inside a string
                echo "Your name is $name";
                // Alternative way of doing the same
                echo "<br/>Your name is ". $name." welcome to our page.";
```

Accessing PHP variable declared in the top block of PHP.

PHP Basic Stuff Demo 01

PHP include files

- The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.

- Including files is very useful when you want to include the same PHP, HTML, or text on multiple pages of a website.

- It is possible to insert the content of one PHP file into another PHP file (before the server executes it), with the include or require statement.

The include and require statements are identical, except upon failure:

- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

- So, if you want the execution to go on and show users the output, even if the include file is missing, use the include statement. Otherwise, in case of FrameWork, CMS, or a complex PHP application coding, always use the require statement to include a key file to the flow of execution. This will help avoid compromising your application's security and integrity, just in-case one key file is accidentally missing.

PHP Basic Stuff Demo 01

PHP include files

Syntax

```
include 'filename';

or

require 'filename';
```

PHP Basic Stuff Demo 01

PHP include files

Example 1

Assume we have a standard footer file called "footer.php", that looks like this:

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";
?>
```

To include the footer file in a page, use the `include` statement:

Example

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

OUTPUT

Welcome to my home page!

Some text.

Some more text.

Copyright © 1999-2023 W3Schools.com

PHP Basic Stuff Demo 01

PHP include files

Example 2

Assume we have a standard menu file called "menu.php":

```
<?php
echo '<a href="/default.asp">Home</a> -
<a href="/html/default.asp">HTML Tutorial</a> -
<a href="/css/default.asp">CSS Tutorial</a> -
<a href="/js/default.asp">JavaScript Tutorial</a> -
<a href="default.asp">PHP Tutorial</a>';
?>
```

All pages in the Web site should use this menu file. Here is how it can be done (we are using a element so that the menu easily can be styled with CSS later):

Example

```
<html>
<body>

<div class="menu">
<?php include 'menu.php';?>
</div>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>

</body>
</html>
```

OUTPUT

Home - HTML Tutorial - CSS Tutorial - JavaScript Tutorial - PHP 7 Tutorial

Welcome to my home page!

Some text.

Some more text.

PHP Basic Stuff Demo 01

PHP Form processing

- The PHP superglobals $_GET and $_POST are used to collect form-data.
- the example below displays a simple HTML form with two input fields and a submit button:

```
<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>
```

- When the user fills out the form above and clicks the submit button, the form data is sent for processing to a PHP file named "welcome.php". The form data is sent with the HTTP POST method.
- To display the submitted data you could simply echo all the variables. The "welcome.php" looks like:
  It is on next slide

PHP Basic Stuff Demo 01

PHP Form processing

- The "welcome.php" looks like:

```
<html>
<body>

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

</body>
</html>
```

- The output could be something like this:

```
Welcome John
Your email address is john.doe@example.com
```

The same result could also be achieved using the HTTP GET method as well.

The code above is quite simple. However, the most important thing is missing. You need to validate form data to protect your script from malicious code.

PHP Basic Stuff Demo 01

PHP Form processing

- Think SECURITY when processing PHP forms!

- The page on previous slide does not contain any form validation, it just shows how you can send and retrieve form data.

- However, the next slides will show how to process PHP forms with security in mind! Proper validation of form data is important to protect your form from hackers and spammers!

PHP Basic Stuff Demo 01

PHP Form processing

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.

PHP Basic Stuff Demo 01

PHP Form processing

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.
- **Developers prefer POST for sending form data.**

PHP Basic Stuff Demo 01

PHP Form Validation

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

We check whether the form has been submitted using $_SERVER["REQUEST_METHOD"]. If the REQUEST_METHOD is POST, then the form has been submitted - and it should be validated. If it has not been submitted, skip the validation and display a blank form.

PHP Basic Stuff Demo 01

PHP Form processing

PHP $_REQUEST - $_REQUEST captures both Get and Post

- PHP $_REQUEST is a PHP super global variable which is used to collect data after submitting an HTML form.
- The example below shows a form with an input field and a submit button.
- When a user submits the data by clicking on "Submit", the form data is sent to the file specified in the action attribute of the <form> tag.
- In this example, we point to this file itself for processing form data. If you wish to use another PHP file to process form data, replace that with the filename of your choice. Then, we can use the super global variable $_REQUEST to collect the value of the input field:

```
<html>
<body>

<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // collect value of input field
  $name = $_REQUEST['fname'];
  if (empty($name)) {
    echo "Name is empty";
  } else {
    echo $name;
  }
}
?>

</body>
</html>
```

PHP Basic Stuff Demo 01

PHP Form Validation

isset() checks whether a variable is set, which means it is declared and is not null.

```
if(isset($_POST["name"]) && strlen($_POST["name"]) > 0 && isset($_POST['age'])
&& strlen($_POST['age'])>0)
{ // validating that both variables are set and are not empty.

   // Sanitize submitted data
   // Trim()- trim function removes white spaces from both sides of the
      string.
    //strip_tags()- strips a string from HTML, XML and PHP tags.

    $name= strip_tags( trim($_POST["name"]) );
    $age= strip_tags( trim($_POST["age"]) );

    // We are good to use submitted data
}
```

PHP Basic Stuff Demo 01

PHP Form Validation

Other validations can also be imposed depending upon the requirements.

The is_numeric() function checks whether a variable is a number or a numeric string.

This function returns true (1) if the variable is a number or a numeric string, otherwise it returns false/nothing.

```
if(is_numeric($age))
   {
        //concat age part into the output
        $output.= "Your age is $age";
   }
```

PHP Basic Stuff Demo 01

PHP Form - testing of the PHP page using hardcoded GET in URL

You can test your form by hardcoding the values in the URL

https://thor.cnt.sast.ca/~aulakhha/1231/CMPE2550A04/Demos/Demo01/Demo01forms.php?name=Simran&age=42