What is PHP?

HYPERTEXT PREPROCESSOR (formerly, *Personal Home Page*).

Created by ***Rasmus Lerdorf*** in the year ***1995***.

It is a server-side scripting language.

It is a powerful tool for making dynamic and interactive Web pages. For this, the PHP code is embedded into the HTML source document and interpreted by a web server with a PHP processor module, which generates the web page document.

It supports many databases (MySQL, Oracle, Sybase, etc.).

It is a widely-used & free (open source).

PHP runs on different platforms (Windows, Linux, Unix, etc.).

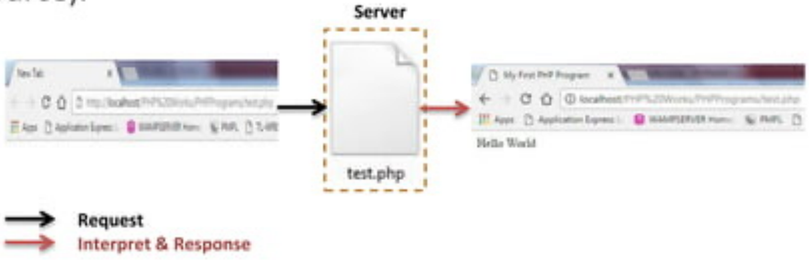
Latest version: PHP5

HTML + CSS + JavaScript + PHP = Dynamic HTML (DHTML)

Why Hypertext Preprocessor?

Machine readable text.

Libraries are already compiled. When any user request for any PHP page in the address bar of the browser, that request is first sent to the server, the server then interprets the PHP file, and returns back the response in the HTML form (that's why the PHP code is invisible when we view the page source).



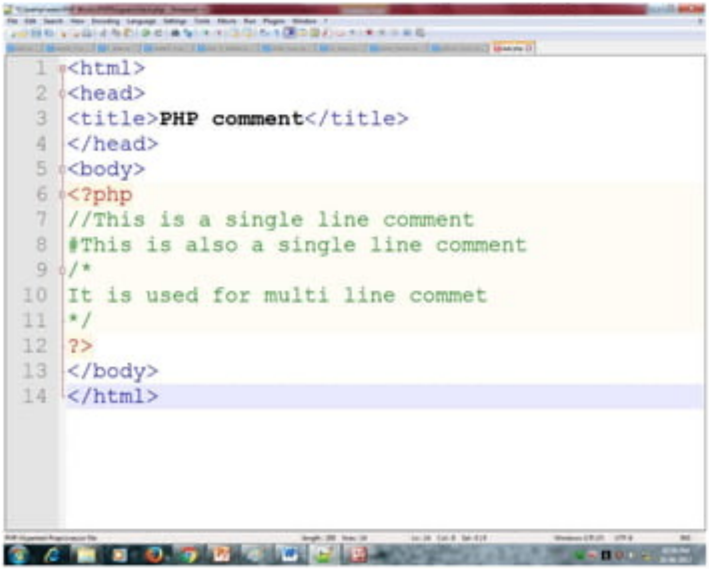
Example

```
<html>

<head>

<title>PHP code within HTML code</title>

</head>

<body>

<?php

echo "Hello World";

?>

</body>

</html>
```

Example (2)

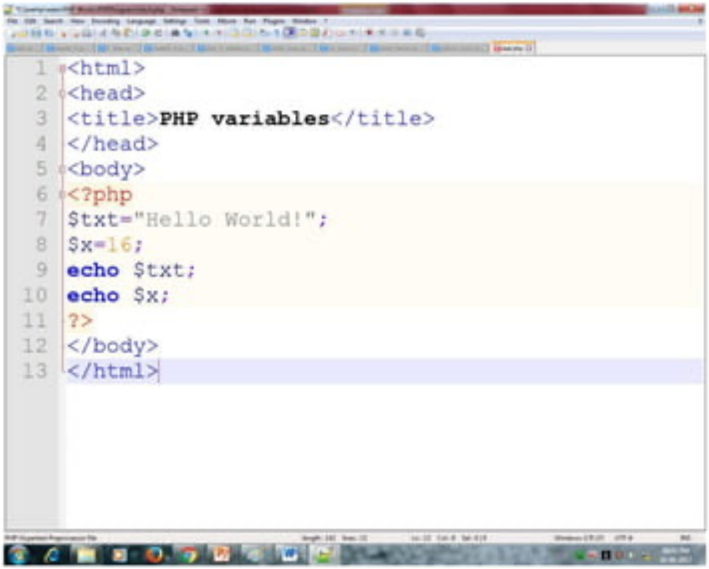
```
<html>

<head>

<title>HTML code within PHP code</title>

</head>

<body>

<?php

echo "<h1>Hello World</h1>";

?>

</body>

</html>
```

PHP Comment Lines



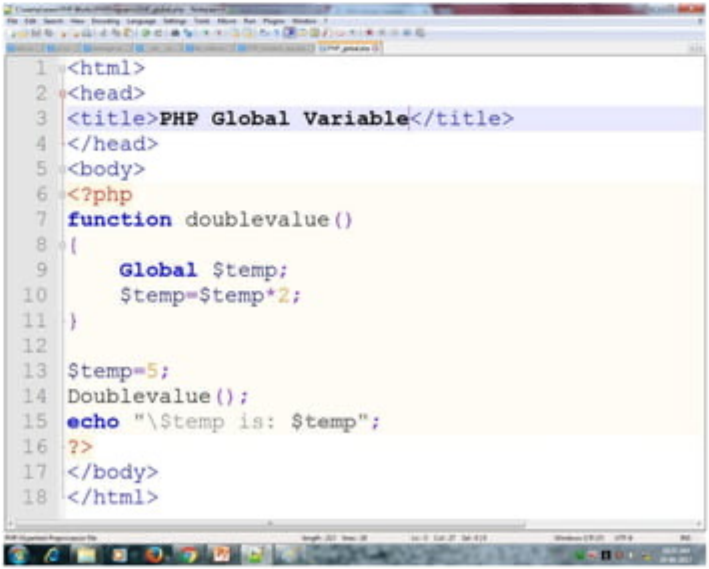
```
<html>
<head>
<title>PHP comment</title>
</head>
<body>
<?php
//This is a single line comment
#This is also a single line comment
/*
It is used for multi line commet
*/
?>
</body>
</html>
```

PHP Variables



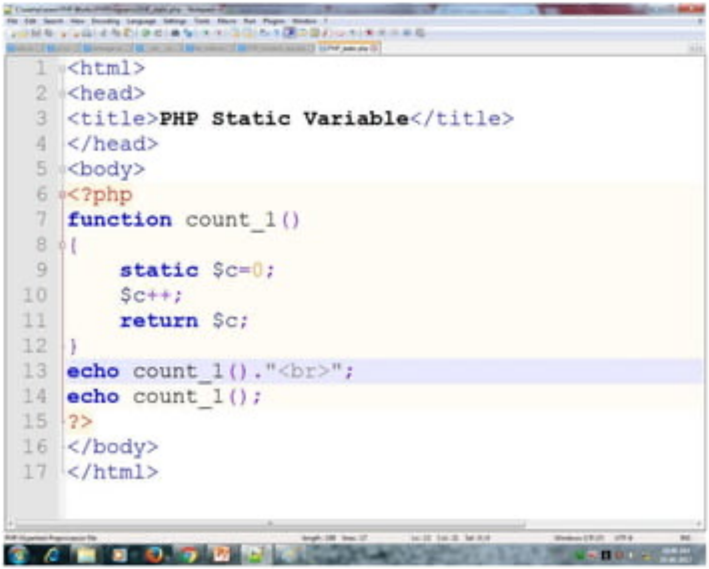
```
<html>
<head>
<title>PHP variables</title>
</head>
<body>
<?php
$txt="Hello World!";
$x=16;
echo $txt;
echo $x;
?>
</body>
</html>
```

PHP Global Variable



```
<html>
<head>
<title>PHP Global Variable</title>
</head>
<body>
<?php
function doublevalue()
{
    Global $temp;
    $temp=$temp*2;
}

$temp=5;
Doublevalue();
echo "\$temp is: $temp";
?>
</body>
</html>
```

PHP Static Variable



```
<html>
<head>
<title>PHP Static Variable</title>
</head>
<body>
<?php
function count_1()
{
     static $c=0;
     $c++;
     return $c;
}
echo count_1()."<br>";
echo count_1();
?>
</body>
</html>
```

echo VS *print*

echo Statement	*print* Statement
Has no return value.	Has a return value of 1.
Can take multiple parameters.	Can take one argument.
echo is marginally faster than *print*.	*print* is slower than *echo*.

`<html>`
`<head>`
`<title>PHP echo statement</title>`
`</head>`
`<body>`
`<?php`
`        echo "Hello World! <br />";`
`?>`
`</body>`
`</html>`
	`<html>`
	`<head>`
	`<title>PHP print statement</title>`
	`</head>`
	`<body>`
	`<?php`
	`        print "Hello World! <br />";`
	`?>`
	`</body>`
	`</html>`

PHP String Functions

Function	Description	Example
ltrim()	Removes whitespace or other characters from the left side of a string.	`<?php` `$str = " Hello World!";` `echo "Without ltrim:".$str;` `echo "<br />";` `echo "With ltrim:".ltrim($str);` `?>` Without ltrim: Hello World! With ltrim:Hello World!
rtrim()	Removes whitespace or other characters from the right side of a string.	`<?php` `$str = "Hello World!    ";` `echo "Without rtrim: " . $str;` `echo "<br />";` `echo "With rtrim: " . rtrim($str);` `?>` Without rtrim: Hello World! With rtrim: Hello World!

PHP String Functions (2)

Function	Description	Example
trim()	Removes whitespace or other characters from the both side of a string.	`<?php` `$str = " Hello World! ";` `echo "Without trim: " . $str;` `echo "<br />";` `echo "With trim: " . trim($str);` `?>`
		Without trim: Hello World! **With trim: Hello World!**
str_pad()	Pads a string to a new length.	`<?php` `$str = "Hello World";` `echo str_pad($str,20,".")." ";` `echo str_pad($str,20,".",STR_PAD_LEFT)." ";` `echo str_pad($str,20,".:",STR_PAD_BOTH);` `?>`
		Hello World......... **.........Hello World** **.:.:Hello World.:.:.**

PHP String Functions (3)

Function	Description	Example
strrev()	Reverses a string.	`<?php echo strrev("Hello World!"); ?>`
		!dlroW olleH
strchr()	Finds the first occurrence of a string inside another string and returns the rest of the string.	`<?php echo strchr("Hello world!","world")." "; echo strchr("Hello world!",111); ?>`
		world! o world!

PHP String Functions (4)

Function	Description	Example
strcmp()	Compares two strings (case-sensitive), & returns, 0 (if the two strings are equal), <0 (if string1 is less than string2), >0 (if string1 is greater than string2).	`<?php echo strcmp("Hello world!","Hello world!"); ?>` 0
strncmp()	String comparison of the first n characters (case-sensitive.	`<?php echo strncmp("Hello world!","Hello earth!",6); ?>` 0

PHP String Functions (5)

Function	Description	Example
strlen()	Returns the length of a string.	`<?php echo strlen("Hello world!"); ?>` 12
strpos()	Returns the position of the first occurrence of a string inside another string (case-insensitive).	`<?php echo strpos("Hello world!","wo"); ?>` 6

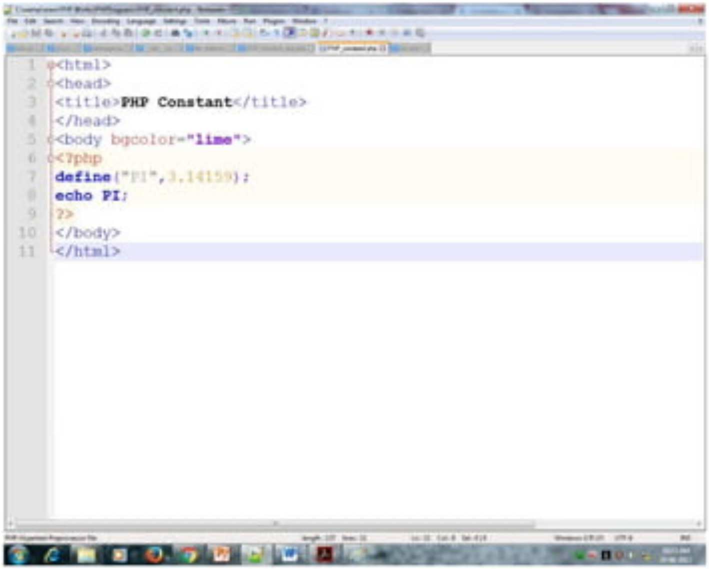
PHP String Functions (6)

Function	Description	Example
wordwrap()	Wraps a string to a given number of characters.	`<?php $str = "An example on a long word is: Supercalifragulistic"; echo wordwrap($str,15," ",TRUE); ?>`
		An example on a long word is: Supercalifragul istic

PHP String Functions (7)

Function	Description	Example
printf()	Outputs a formatted string.	`<?php` `printf("PI=%f",3.14159);` `echo " " ;` `printf("%.2f",3.14159);` `echo " " ;` `printf("%.10f",3.14159);` `echo " " ;` `printf("%10.2f",3.14159);` `echo " " ;` `printf("%9s","halfofthestring"` `);` `echo " " ;` `printf("%b %d %f` `%s",123,123,123,"test");` `?>`
		PI=3.141590 3.14 3.1415900000 3.14 halfofthestring 1111011 123 123.000000 test
……		

PHP Constants

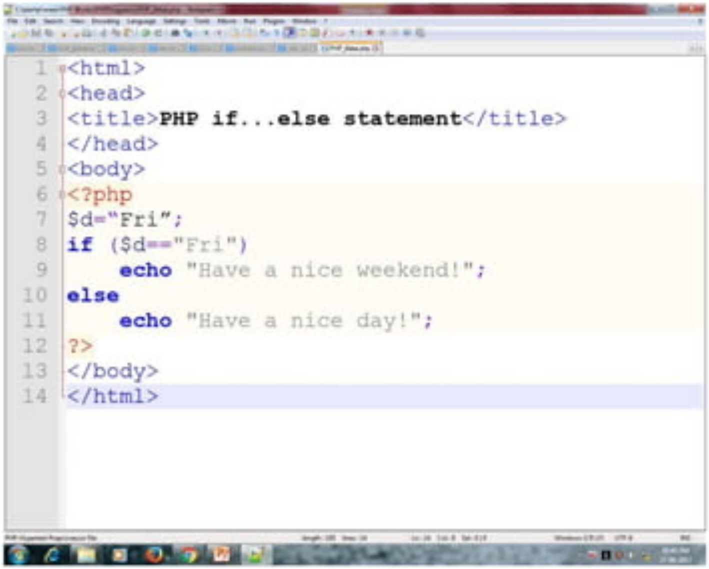


```
<html>
<head>
<title>PHP Constant</title>
</head>
<body bgcolor="lime">
<?php
define("PI",3.14159);
echo PI;
?>
</body>
</html>
```

PHP Operators

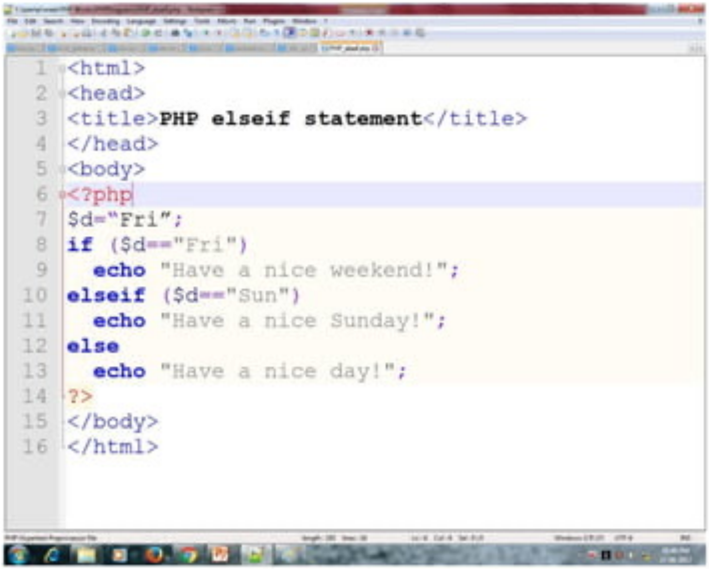
Category	Operator
Arithmetic operators	+, -, *, /, %, **
Assignment operators	=, +=, -=, *=, /=, %=
Comparison operators	==, ===, !=, <>, !==, >, <, >=, <=
Increment/Decrement operators	++$x, $x++, --$x, $x--
Logical operators	and, or, xor, &&, ||, !
String operators	., .=
Array operators	+, ==, ===, !=, <>, !==

PHP *if...else* Statement



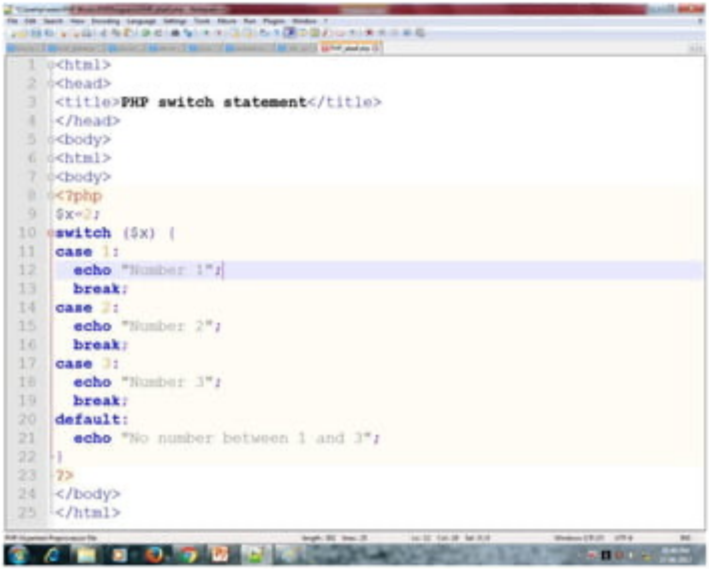
```
<html>
<head>
<title>PHP if...else statement</title>
</head>
<body>
<?php
$d="Fri";
if ($d=="Fri")
     echo "Have a nice weekend!";
else
     echo "Have a nice day!";
?>
</body>
</html>
```

PHP *elseif* Statement



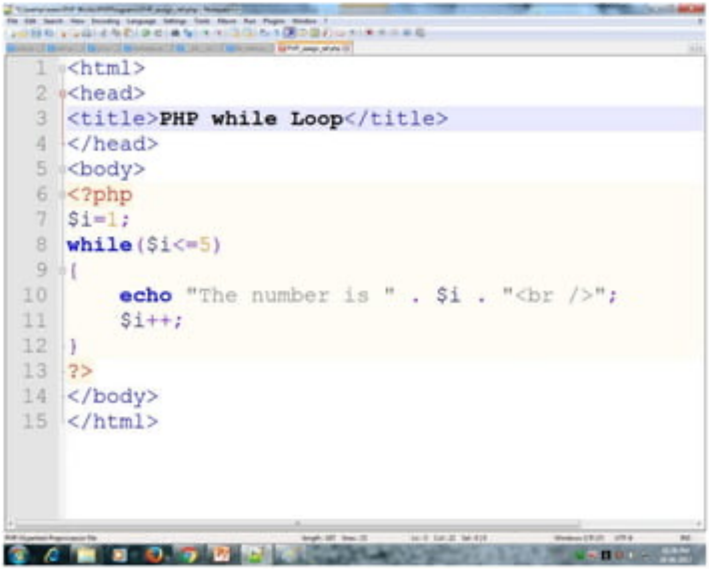
```
<html>
<head>
<title>PHP elseif statement</title>
</head>
<body>
<?php
$d="Fri";
if ($d=="Fri")
  echo "Have a nice weekend!";
elseif ($d=="Sun")
  echo "Have a nice Sunday!";
else
  echo "Have a nice day!";
?>
</body>
</html>
```

PHP *switch* Statement



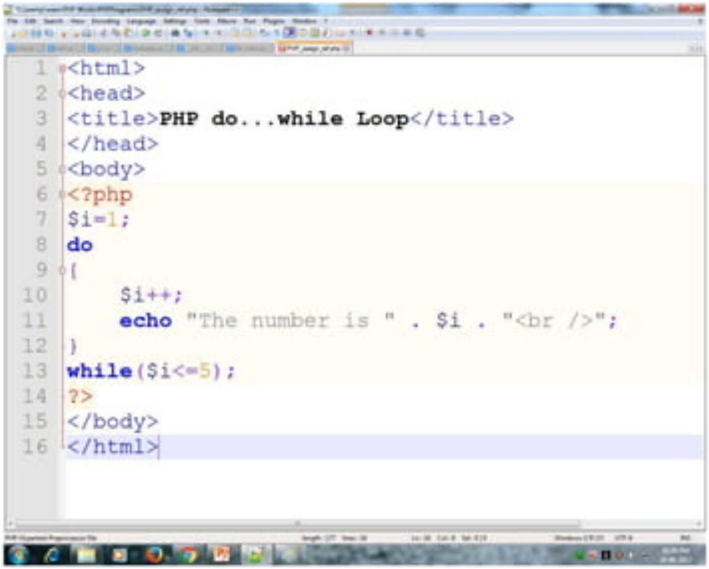
```
<html>
<head>
<title>PHP switch statement</title>
</head>
<body>
<html>
<body>
<?php
$x=2;
switch ($x) {
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>
</body>
</html>
```

PHP *while* Loop



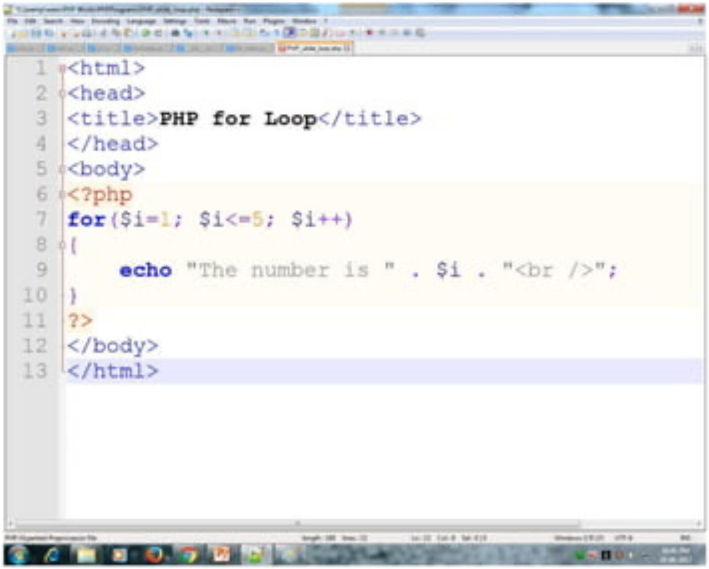
```
<html>
<head>
<title>PHP while Loop</title>
</head>
<body>
<?php
$i=1;
while($i<=5)
{
    echo "The number is " . $i . "<br />";
    $i++;
}
?>
</body>
</html>
```

PHP *do...while* Loop



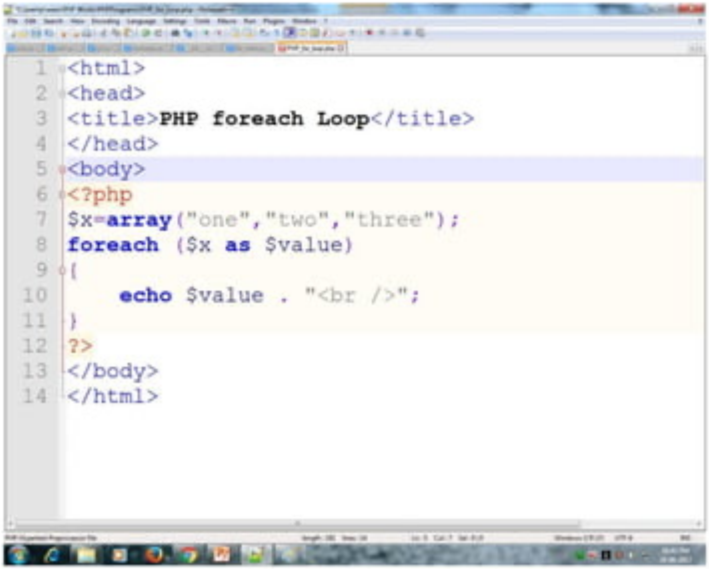
```
<html>
<head>
<title>PHP do...while Loop</title>
</head>
<body>
<?php
$i=1;
do
{
    $i++;
    echo "The number is " . $i . "<br />";
}
while($i<=5);
?>
</body>
</html>
```

PHP *for* Loop



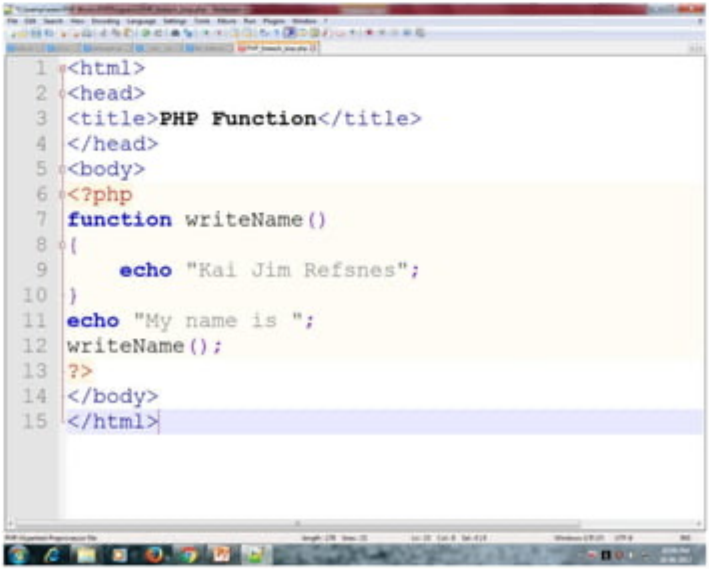
```
<html>
<head>
<title>PHP for Loop</title>
</head>
<body>
<?php
for($i=1; $i<=5; $i++)
{
    echo "The number is " . $i . "<br />";
}
?>
</body>
</html>
```

PHP *foreach* Loop



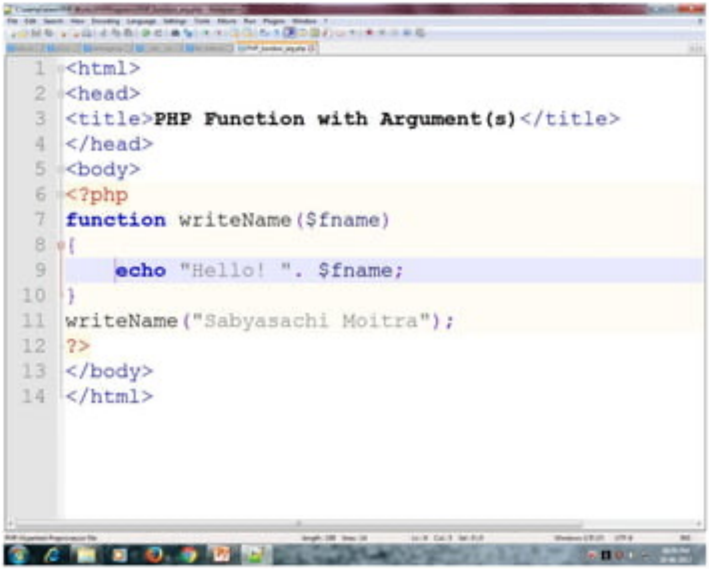
```
<html>
<head>
<title>PHP foreach Loop</title>
</head>
<body>
<?php
$x=array("one","two","three");
foreach ($x as $value)
{
    echo $value . "<br />";
}
?>
</body>
</html>
```

PHP Function



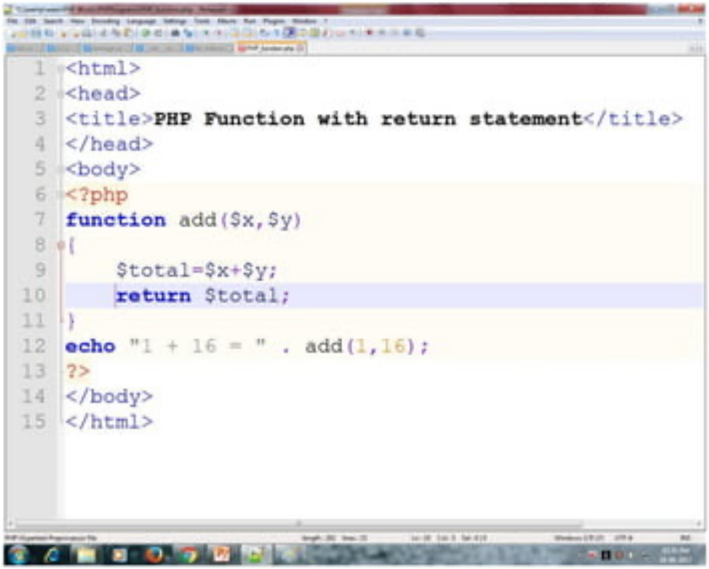
```
<html>
<head>
<title>PHP Function</title>
</head>
<body>
<?php
function writeName()
{
     echo "Kai Jim Refsnes";
}
echo "My name is ";
writeName();
?>
</body>
</html>
```

PHP Function with Argument(s)



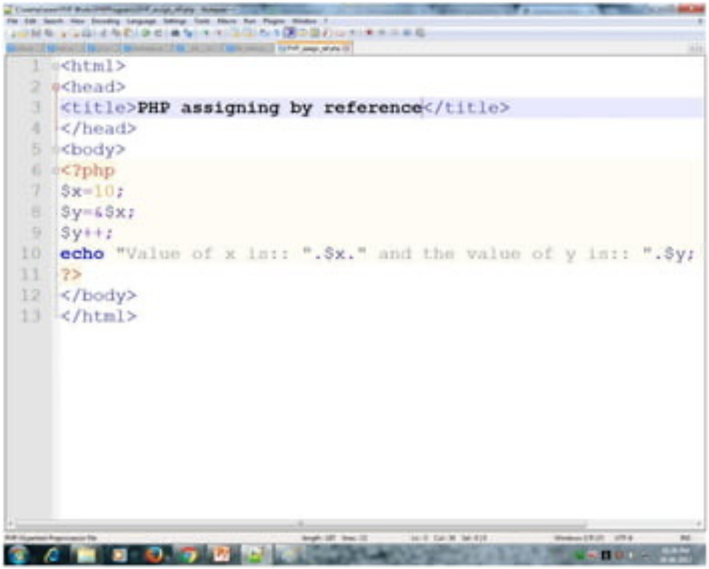
```
<html>
<head>
<title>PHP Function with Argument(s)</title>
</head>
<body>
<?php
function writeName($fname)
{
    echo "Hello! ". $fname;
}
writeName("Sabyasachi Moitra");
?>
</body>
</html>
```

PHP Function with *return* Statement



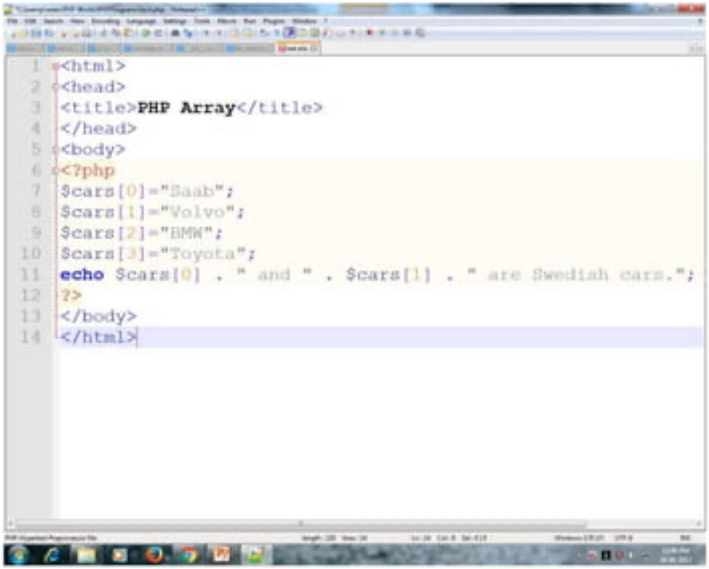
```
<html>
<head>
<title>PHP Function with return statement</title>
</head>
<body>
<?php
function add($x,$y)
{
	$total=$x+$y;
	return $total;
}
echo "1 + 16 = " . add(1,16);
?>
</body>
</html>
```

PHP Assigning by Reference



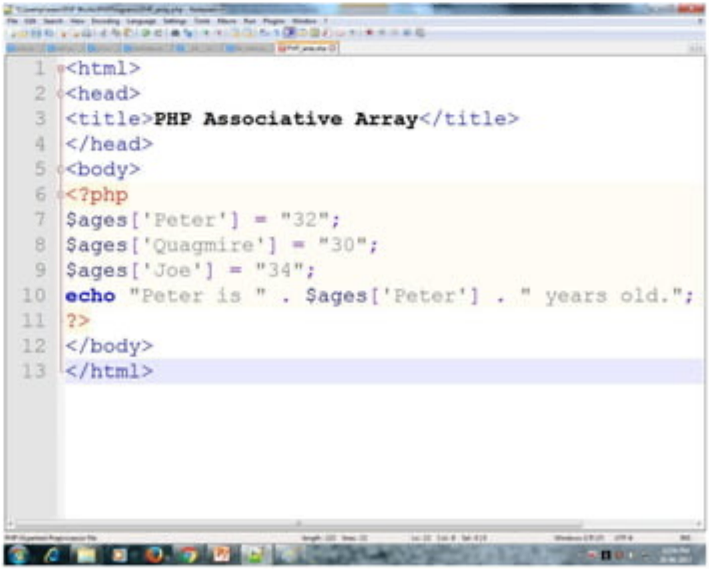
```
<html>
<head>
<title>PHP assigning by reference</title>
</head>
<body>
<?php
$x=10;
$y=&$x;
$y++;
echo "Value of x is:: ".$x." and the value of y is:: ".$y;
?>
</body>
</html>
```

PHP Array



```
<html>
<head>
<title>PHP Array</title>
</head>
<body>
<?php
$cars[0]="Saab";
$cars[1]="Volvo";
$cars[2]="BMW";
$cars[3]="Toyota";
echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
?>
</body>
</html>
```

PHP Associative Array



```
<html>
<head>
<title>PHP Associative Array</title>
</head>
<body>
<?php
$ages['Peter'] = "32";
$ages['Quagmire'] = "30";
$ages['Joe'] = "34";
echo "Peter is " . $ages['Peter'] . " years old.";
?>
</body>
</html>
```

POST vs GET

POST Method	GET Method
Data is not displayed in the URL. Thus, POST is a little safer than GET. `/test/demo_form.php`	Data is visible to everyone in the URL. Thus, less secure compared to POST. `/test/demo_form.php`**`?name1=value1&name2=value2`**

$_POST Variable

$_POST variable is used when POST method is applied in designing.

```
<html>
<head>
<title>PHP $_POST variable</title>
</head>
<body bgcolor="lime">
<form action="PHP_post.php" method="post">
Name: <input type="text" name="name"><br>
<input type="submit" name="submit"
value="Submit">
</form>
Welcome <?php if(isset($_POST['submit'])){echo
$_POST["name"];} ?>!
</body>
</html>
```

$_GET Variable

$_GET variable is used when GET method is applied in designing.

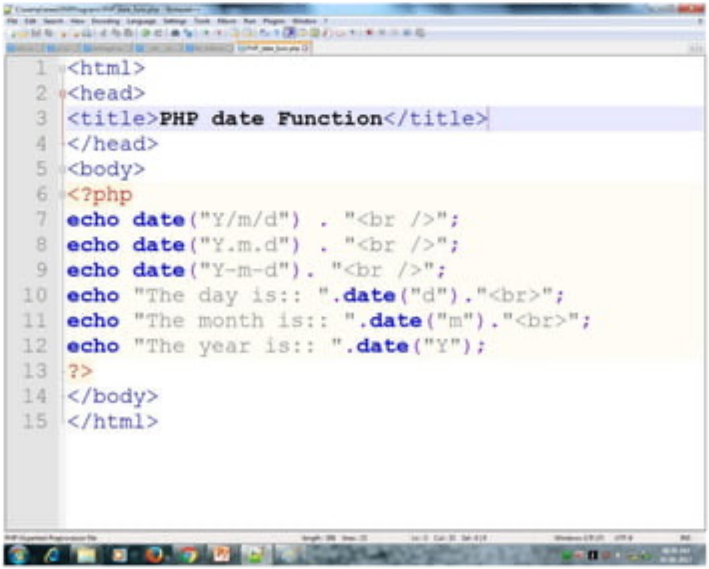
```
<html>
<head>
<title>PHP $_GET variable</title>
</head>
<body bgcolor="lime">
<form action="PHP_get.php" method="get">
Name: <input type="text" name="name"><br>
<input type="submit" name="submit"
value="Submit">
</form>
Welcome <?php if(isset($_GET['submit'])){echo
$_GET["name"];} ?>!
</body>
</html>
```

$_REQUEST Variable

$_REQUEST variable is used when POST/GET method is applied in designing.

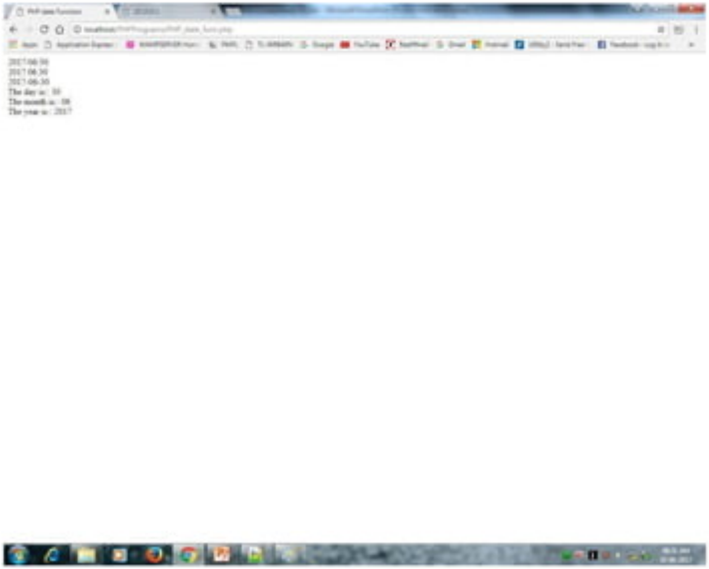
```
<html>
<head>
<title>PHP $_REQUEST variable</title>
</head>
<body bgcolor="lime">
<form action="PHP_request.php" method="get">
Name: <input type="text" name="name"><br>
<input type="submit" name="submit"
value="Submit">
</form>
Welcome <?php
if(isset($_REQUEST['submit'])){echo
$_REQUEST["name"];} ?>!
</body>
</html>
```

PHP *date()* Function



```
<html>
<head>
<title>PHP date Function</title>
</head>
<body>
<?php
echo date("Y/m/d") . "<br />";
echo date("Y.m.d") . "<br />";
echo date("Y-m-d"). "<br />";
echo "The day is:: ".date("d")."<br>";
echo "The month is:: ".date("m")."<br>";
echo "The year is:: ".date("Y");
?>
</body>
</html>
```

Output



```
2017/06/30
2017.06.30
2017-06-30
The day is : 30
The month is : 06
The year is : 2017
```

Some Date Format

W: Week number of year, weeks starting on Monday.

F: A full textual representation of a month, such as January or March.

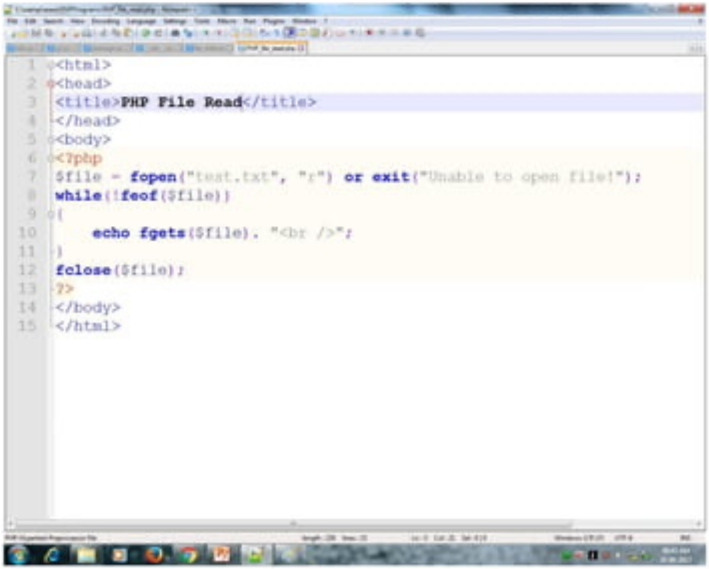
M: A short textual representation of a month, three letters.

y: A two digit representation of a year.

include() VS require()

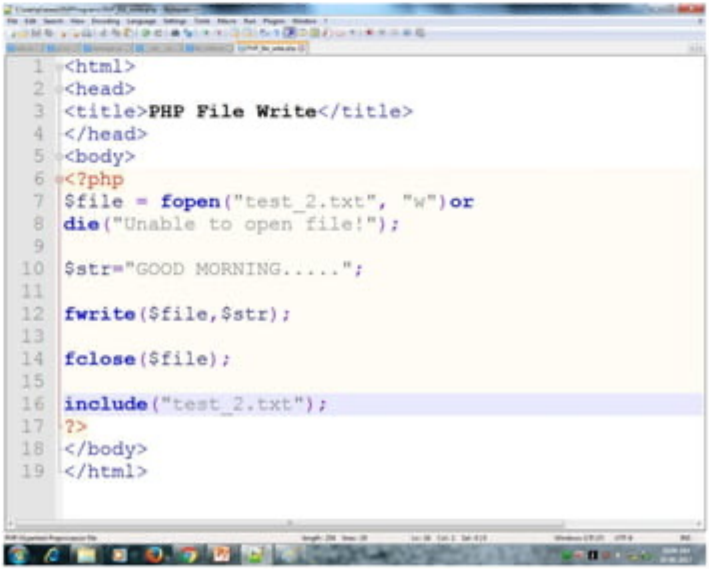
include()	require()
The include() function takes all the content in a specified file and includes it in the current file.	Like include() function, require() function takes all the content in a specified file and includes it in the current file.
If an error occurs, the include() function generates a warning, but the script will continue execution.	If an error occurs, the require() generates a fatal error, and the script will stop.
`<?php` `echo "Different Date Formats: ";` `include("PHP_date_func.php");` `?>`	`<?php` `echo "Different Date Formats: ";` `require("PHP_date_func.php");` `?>`

PHP File Read



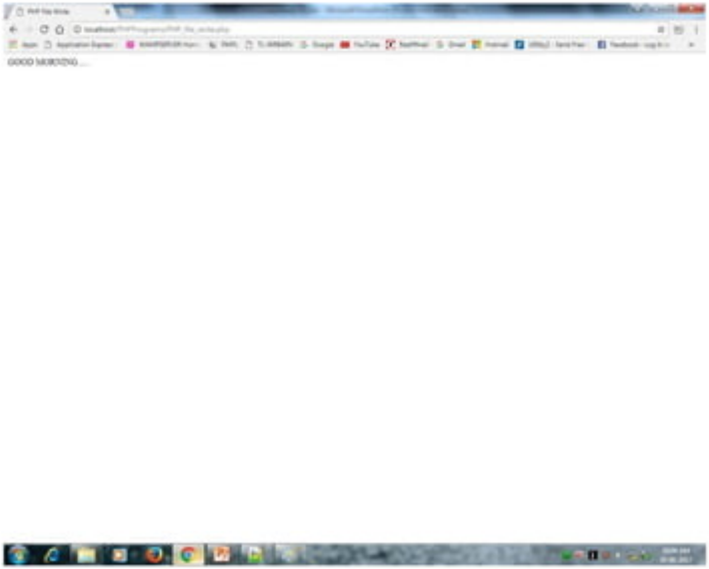
```
<html>
<head>
<title>PHP File Read</title>
</head>
<body>
<?php
$file = fopen("test.txt", "r") or exit("Unable to open file!");
while(!feof($file))
{
    echo fgets($file). "<br />";
}
fclose($file);
?>
</body>
</html>
```

PHP File Write



```
<html>
<head>
<title>PHP File Write</title>
</head>
<body>
<?php
$file = fopen("test_2.txt", "w")or
die("Unable to open file!");

$str="GOOD MORNING.....";

fwrite($file,$str);

fclose($file);

include("test_2.txt");
?>
</body>
</html>
```

Output



```
GOOD MORNING....
```

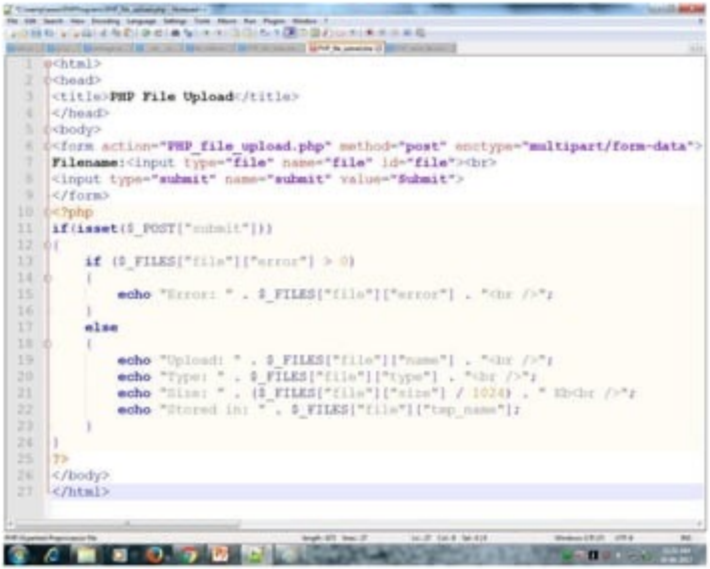
PHP File Opening Modes

Mode	Description
r	Read only. Starts at the beginning of the file. Error if the file doesn't exist.
r+	Read/Write. Starts at the beginning of the file.
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist.
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist.

PHP File Opening Modes (2)

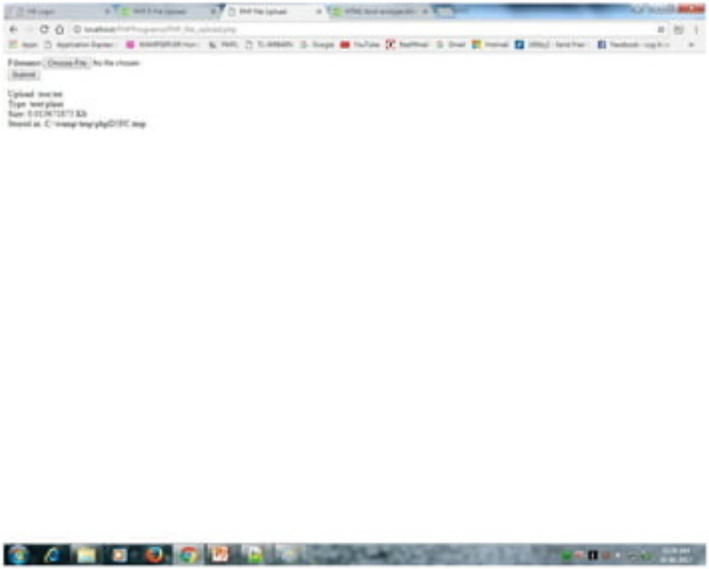
Mode	Description
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist.
a+	Read/Append. Preserves file content by writing to the end of the file.
x	Write only. Creates a new file. Returns FALSE and an error if file already exists.
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists.

PHP File Upload



```
<html>
<head>
<title>PHP File Upload</title>
</head>
<body>
<form action="PHP_file_upload.php" method="post" enctype="multipart/form-data">
Filename:<input type="file" name="file" id="file"><br>
<input type="submit" name="submit" value="Submit">
</form>
<?php
if(isset($_POST["submit"]))
{
    if ($_FILES["file"]["error"] > 0)
    {
        echo "Error: " . $_FILES["file"]["error"] . "<br />";
    }
    else
    {
        echo "Upload: " . $_FILES["file"]["name"] . "<br />";
        echo "Type: " . $_FILES["file"]["type"] . "<br />";
        echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
        echo "Stored in: " . $_FILES["file"]["tmp_name"];
    }
}
?>
</body>
</html>
```

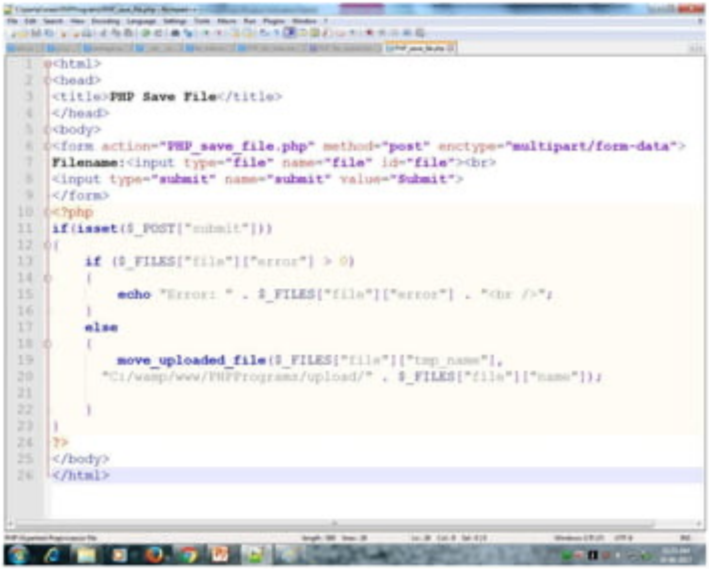
Output



```
Filename: Choose File  No file chosen
Submit

Upload: test.txt
Type: text/plain
Size: 0.013671875 Kb
Stored in: C:\wamp\tmp\phpD1FC.tmp
```

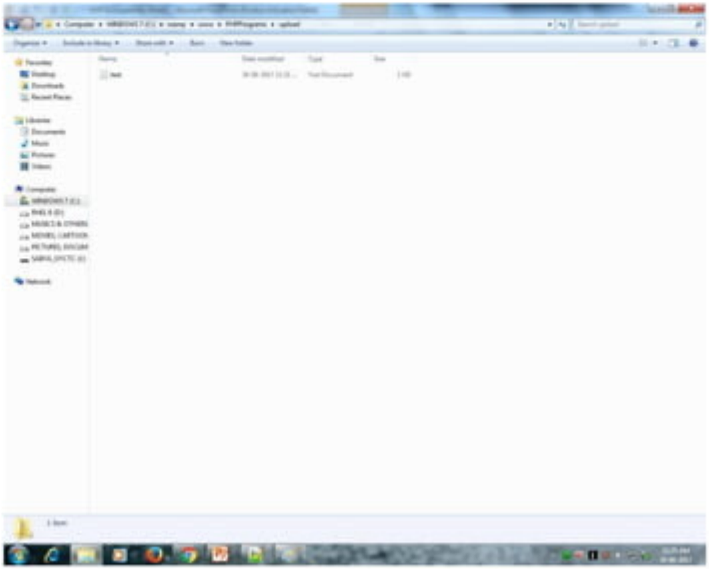
	Description
enctype="multipart/form-data"	• The *enctype* attribute specifies how the form-data should be encoded when submitting it to the server. • *multipart/form-data* value specifies that no characters are encoded.
$_FILES	An associate double dimension array that keeps all the information related to the uploaded file.
$_FILES['file']['error']	The error code associated with the uploaded file.
$_FILES['file']['name']	The actual name of the uploaded file.
$_FILES['file']['type']	The MIME type of the uploaded file.
$_FILES['file']['size']	The size in bytes of the uploaded file.
$_FILES['file']['tmp_name']	The uploaded file in the temporary directory on the web server.

PHP Save File



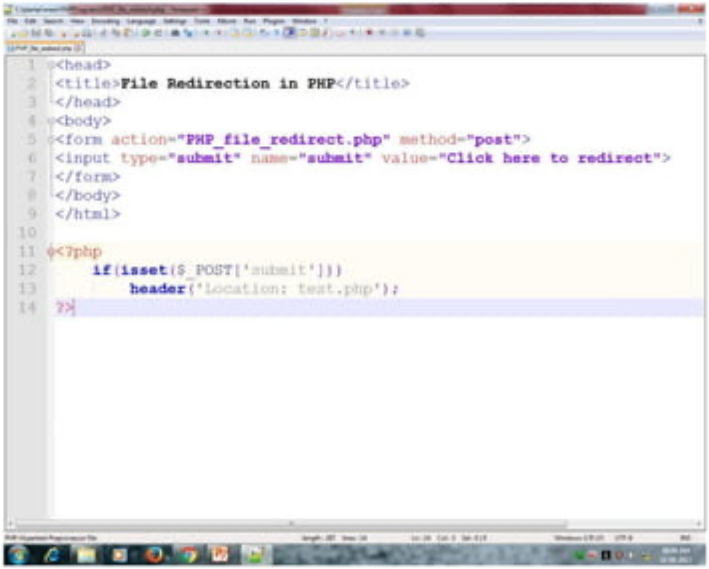
```
<html>
<head>
<title>PHP Save File</title>
</head>
<body>
<form action="PHP_save_file.php" method="post" enctype="multipart/form-data">
Filename:<input type="file" name="file" id="file"><br>
<input type="submit" name="submit" value="Submit">
</form>
<?php
if(isset($_POST["submit"]))
{
    if ($_FILES["file"]["error"] > 0)
    {
        echo "Error: " . $_FILES["file"]["error"] . "<br />";
    }
    else
    {
        move_uploaded_file($_FILES["file"]["tmp_name"],
      "C:/wamp/www/PHPPrograms/upload/" . $_FILES["file"]["name"]);

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

Output


Computer ▸ WINDOWS 7 (C:) ▸ wamp ▸ www ▸ PHPPrograms ▸ upload
Organize
Include in library
Share with
Burn
New folder
Name
Date modified
Type
Size
test
Text Document
1 KB
1 item

	Description
move_uploaded_file(file,newloc)	Moves an uploaded file to a new location.

File Redirection in PHP



```
<head>
<title>File Redirection in PHP</title>
</head>
<body>
<form action="PHP_file_redirect.php" method="post">
<input type="submit" name="submit" value="Click here to redirect">
</form>
</body>
</html>

<?php
    if(isset($_POST['submit']))
        header('Location: test.php');
?>
```

PHP Cookie

A cookie is often used to identify a user.

A cookie is a small file that the server embeds on the user's computer.

Each time the same computer requests a page with a browser, it will send the cookie too.

Some Cookie Operations

Operation	Example

Create a Cookie	`<?php`
	`setcookie("user", "Alex Porter");`
	`?>`

Retrieve a Cookie	`<?php`
	`// Print a cookie`
	`echo $_COOKIE["user"];`
	
	`// A way to view all cookies`
	`print_r($_COOKIE);`
	`?>`

Delete a Cookie	`<?php`
	`// set the expiration date to`
	`after one hour`
	`setcookie("user", "",`
	`time()+3600);`
	`?>`

PHP Session

A session is a way to store information (in variables) about one single user, to be used across multiple pages.

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.

PHP Session Start

```
<?php session_start(); ?>
<html>
<body>
</body>
</html>
```

Before you can store user information in your PHP session, you must first start up the session.

The *session_start()* function must appear before the *<html>* tag.

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

Store PHP Session Variable

The correct way to store and retrieve session variables is to use the PHP $_SESSION variable.

```
<?php
session_start();
?>
<html>
<head>
<title>Store &amp; Retrieve PHP Session
Variable</title>
</head>
<body>
<?php
$_SESSION["views"]=1;     //storing PHP session
variable
echo $_SESSION["views"];  //retrieving PHP session
variable
?>
</body>
</html>
```

PHP Session Destroy

Code	Description

`<?php`
`unset($_SESSION['views']);`
`?>`
	The *unset()* function is used to free the specified session variable.

`<?php`
`session_destroy();`
`?>`
	The *session_destroy()* function is used to completely destroy the session. *session_destroy()* will reset your session and you will lose all your stored session data.

Cookie VS Session

Cookie	Session
Stores user's information to client's side.	Stores user's information to server's side.

PHP Filters

PHP filters are used to validate and filter data coming from insecure sources (input data from a form, database query results, etc.).

PHP Filter Functions

Function	Description
filter_var()	Filters a single variable with a specified filter.
filter_var_array()	Filter several variables with the same or different filters.
filter_input()	Get one input variable and filters it.
filter_input_array()	Get several input variables and filter them with the same or different filters.

The *filter_var()* Function

	Explanation
`<?php` `$int = 123;` `if(!filter_var($int, FILTER_VALIDATE_INT))` `{` `echo("Integer is not valid");` `}` `else` `{` `echo("Integer is valid");` `}` `?>`	• The code uses the "FILTER_VALIDATE_INT" filter to filter the variable. • Since the integer is valid, the output of the code above will be: "Integer is valid". • If we try with a variable that is not an integer (like "123abc"), the output will be: "Integer is not valid".

Options and Flags

Options and flags are used to add additional filtering options to the specified filters.

Different filters have different options and flags.

Example

```
<?php
$var=300;
$int_options = array(
"options"=>array
(
"min_range"=>0,
"max_range"=>256
)
);
if(!filter_var($var,
FILTER_VALIDATE_INT, $int_options))
{
echo("Integer is not valid");
}
else
{
echo("Integer is valid");
}
?>
```

Explanation

- The *filter_var()* function uses *min_range* & *max_range* options.
- Options must be put in an associative array with the name "options". If a flag is used it does not need to be in an array.
- Since the integer is "300" it is not in the specified range, and the output of the code above will be: "Integer is not valid".

Validating VS Sanitizing

Validating Filters	Sanitizing Filters
Used to validate user input.	Used to allow or disallow specified characters in a string.
Strict format rules (like URL or E-Mail validating).	No data format rules.
Returns the expected type on success or FALSE on failure.	Always return the string.

Validate E-mail by PHP

```php
<?php
if(!filter_has_var(INPUT_GET,
"email"))
{
echo("Input type does not exist");
}
else
{
if (!filter_input(INPUT_GET, "email",
FILTER_VALIDATE_EMAIL))
{
echo "E-Mail is not valid";
}
else
{
echo "E-Mail is valid";
}
}
?>
```

Explanation

The example has an input (email) sent to it using the "GET" method:

- Check if an "email" input variable of the "GET" type exist.
- If the input variable exists, check if it is a valid e-mail address.

Sanitize URL by PHP

	Explanation

`<?php`
`if(!filter_has_var(INPUT_POST, "url"))`
`  {`
`  echo("Input type does not exist");`
`  }`
`else`
`  {`
`  $url = filter_input(INPUT_POST, "url", FILTER_SANITIZE_URL);`
`  }`
`?>`
	The example above has an input (url) sent to it using the "POST" method:
	• Check if the "url" input variable of the "POST" type exists.
	• If the input variable exists, sanitize (take away invalid characters) and store it in the $url variable.
	- *If the input variable is a string like this "http://www.W3ååSchøøools.com/", the $url variable after the sanitizing will look like this: http://www.W3Schools.com/*

PHP MySQL Database

With PHP, you can connect to and manipulate databases.

MySQL is the most popular database system used with PHP

- MySQL is a database system used on the web

- MySQL is a database system that runs on a server

- MySQL is ideal for both small and large applications

- MySQL is very fast, reliable, and easy to use

- MySQL uses standard SQL

- MySQL compiles on a number of platforms

- MySQL is free to download and use

- MySQL is developed, distributed, and supported by Oracle Corporation

- MySQL is named after co-founder Monty Widenius's daughter: My

PHP MySQL Connectivity

Type	Code

MySQLi (OOP)	`<?php`
	`$servername = "localhost";`
	`$username = "root";`
	`$password = "";`
	
	`// Create connection`
	`$conn = new mysqli($servername,`
	`$username, $password);`
	
	`// Check connection`
	`if ($conn->connect_error) {`
	`    die("Connection failed: " .`
	`$conn->connect_error);`
	`}`
	`echo "Connected successfully";`
	`?>`

PHP MySQL Connectivity (2)

Type	Code

MySQLi (POP)	`<?php`
	`$servername = "localhost";`
	`$username = "root";`
	`$password = "";`
	
	`// Create connection`
	`$conn =`
	`mysqli_connect($servername,`
	`$username, $password);`
	
	`// Check connection`
	`if (!$conn) {`
	`    die("Connection failed: " .`
	`mysqli_connect_error());`
	`}`
	`echo "Connected successfully";`
	`?>`

PHP MySQL Connectivity (3)

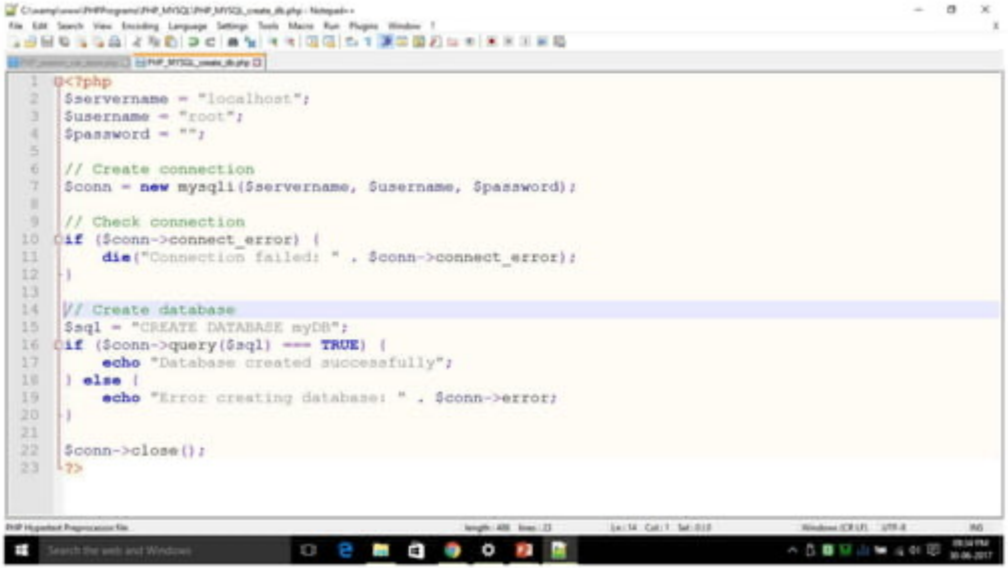
Type	Code

PDO	`<?php`
	`$servername = "localhost";`
	`$username = "root";`
	`$password = "";`
	
	`try {`
	`    $conn = new`
	`PDO("mysql:host=$servername;dbname=myDB",`
	`$username, $password);`
	`    // set the PDO error mode to`
	`exception`
	`    $conn-`
	`>setAttribute(PDO::ATTR_ERRMODE,`
	`PDO::ERRMODE_EXCEPTION);`
	`    echo "Connected successfully";`
	`    }`
	`catch(PDOException $e)`
	`    {`
	`    echo "Connection failed: " . $e-`
	`>getMessage();`
	`    }`
	`?>`

MySQLi VS PDO

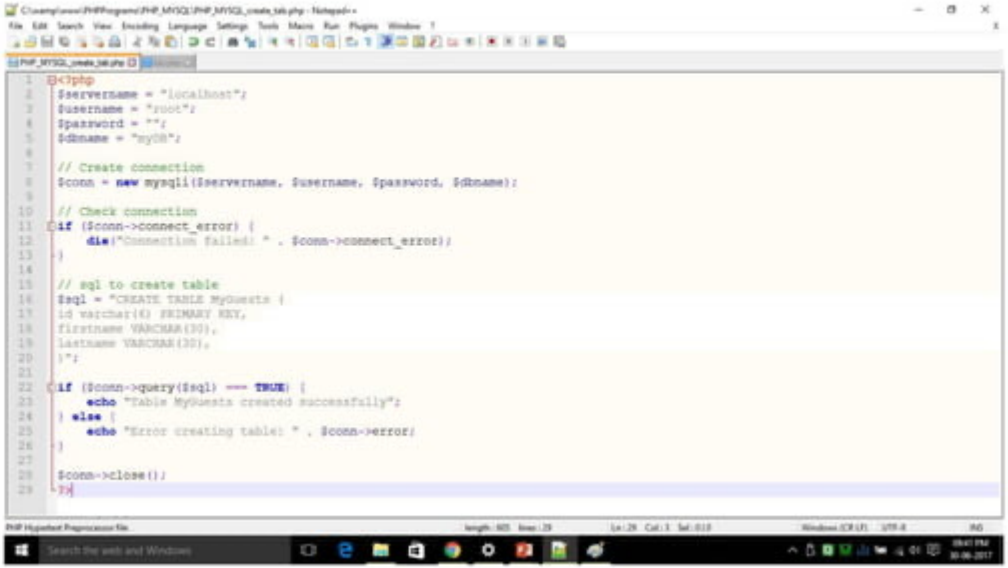
MySQLi	PDO
MySQLi will only work with MySQL databases.	PDO will work on 12 different database systems.
With MySQLi, you will need to rewrite the entire code - queries included.	If you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries.

PHP MySQL Create Database



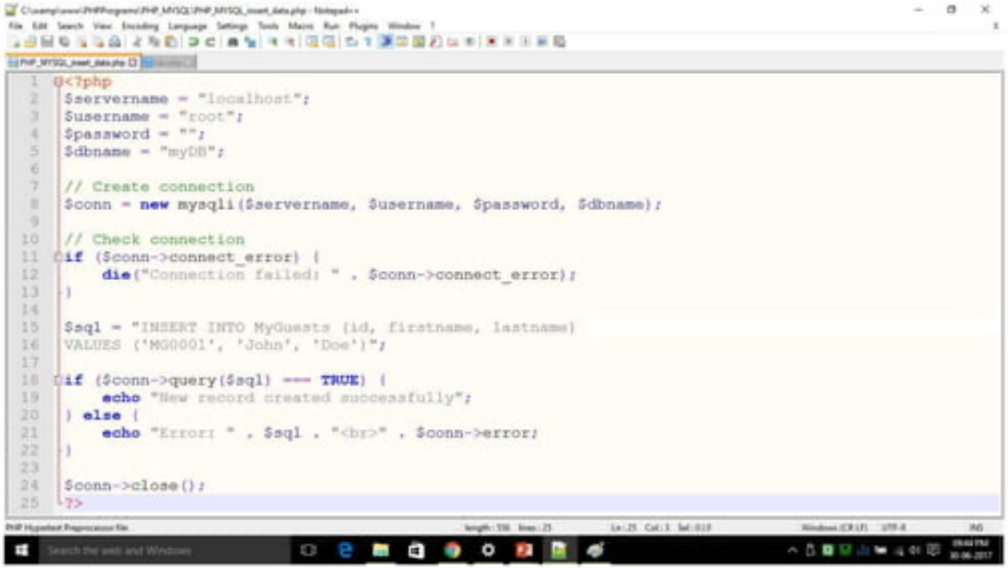
```php
<?php
$servername = "localhost";
$username = "root";
$password = "";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Create database
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " . $conn->error;
}

$conn->close();
?>
```

PHP MySQL Create Table



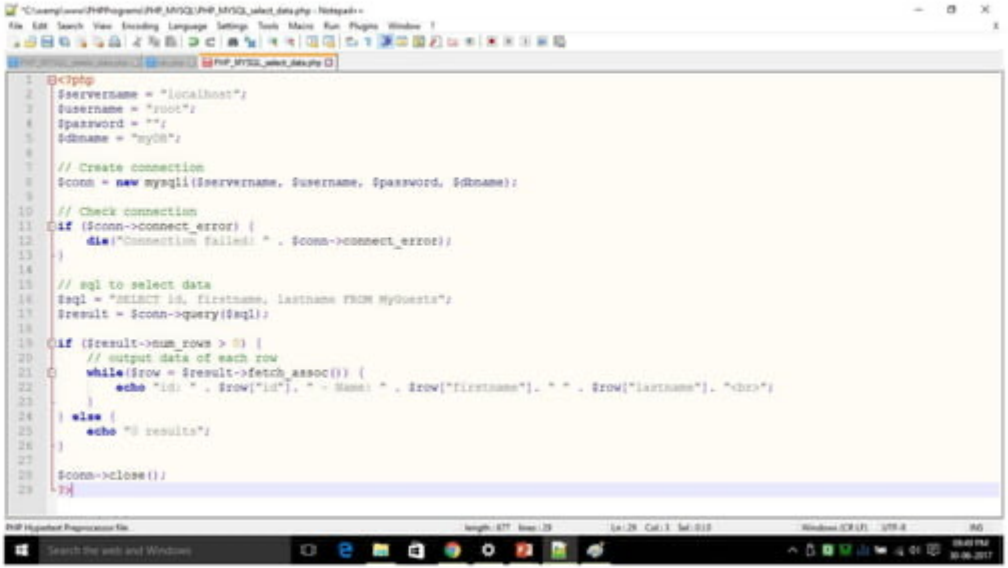
```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to create table
$sql = "CREATE TABLE MyGuests (
id varchar(4) PRIMARY KEY,
firstname VARCHAR(30),
lastname VARCHAR(30),
)";

if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created successfully";
} else {
    echo "Error creating table: " . $conn->error;
}

$conn->close();
?>
```

PHP MySQL Insert Data



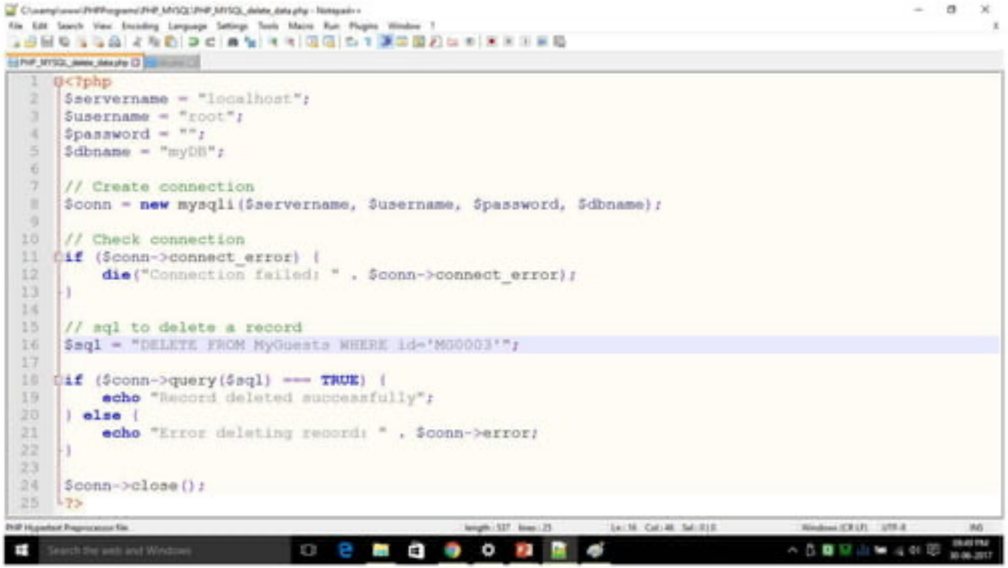
```php
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO MyGuests (id, firstname, lastname)
VALUES ('MG0001', 'John', 'Doe')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
?>
```

PHP MySQL Select Data



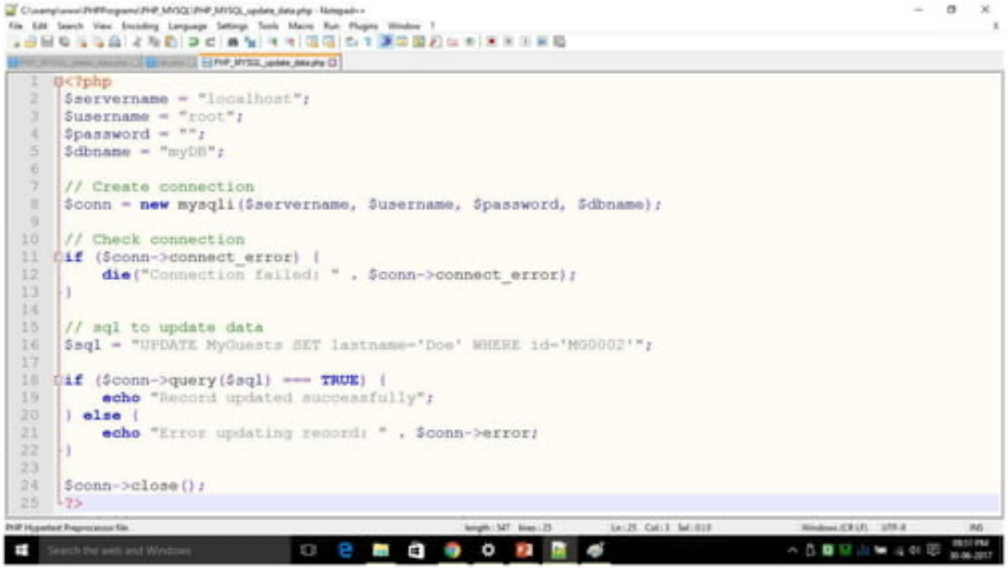
```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to select data
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

$conn->close();
?>
```

PHP MySQL Delete Data



```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id='MG0003'";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn->error;
}

$conn->close();
?>
```

PHP MySQL Update Data



```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// sql to update data
$sql = "UPDATE MyGuests SET lastname='Doe' WHERE id='MG0002'";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn->error;
}

$conn->close();
?>
```

References

Courtesy of W3Schools – PHP Tutorial. URL: http://www.w3schools.com/php/

Courtesy of TutorialsPoint – PHP Tutorial. URL: http://www.tutorialspoint.com/php/

Ivan Bayross, Web Enabled Commercial Applications Development Using HTML, JavaScript, DHTML and PHP, 4th Revised Edition, BPB Publications, 2010