

# 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

---

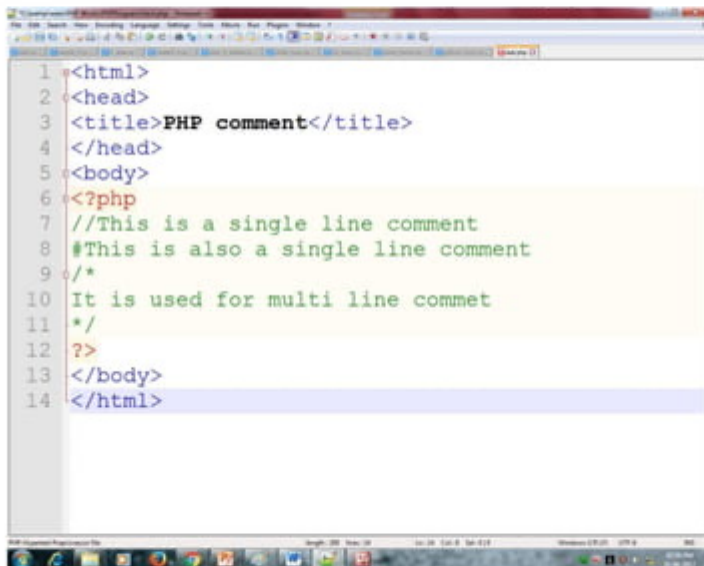
```
<html>
<head>
<title>PHP code within HTML code</title>
</head>
<body>
<?php
echo "Hello World";
?>
</body>
</html>
```

## Example (2)

---

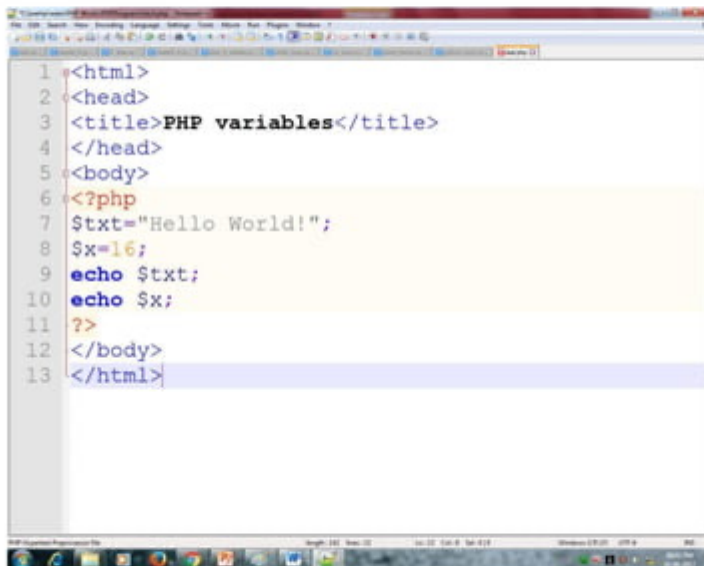
```
<html>
<head>
<title>HTML code within PHP code</title>
</head>
<body>
<?php
echo "<h1>Hello World</h1>";
?>
</body>
</html>
```

# PHP Comment Lines

A screenshot of a web browser window displaying a PHP file. The browser's address bar shows a local file path. The page content is a mix of HTML and PHP code. The PHP section, starting with <?php, contains three types of comments: a single-line comment with //, another single-line comment with #, and a multi-line comment with /\*. The multi-line comment spans three lines. The browser's status bar at the bottom shows the file path, page title, and system clock.

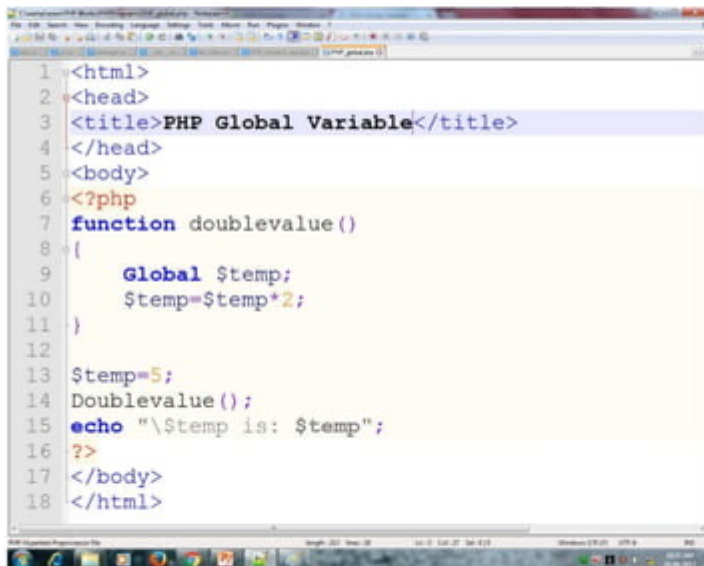
```
1 <html>
2 <head>
3 <title>PHP comment</title>
4 </head>
5 <body>
6 <?php
7 //This is a single line comment
8 #This is also a single line comment
9 /*
10 It is used for multi line comment
11 */
12 ?>
13 </body>
14 </html>
```

# PHP Variables

A screenshot of a web browser window showing the output of a PHP script. The browser's address bar shows a local file path. The page content displays "Hello World!" followed by the number 16. The background of the browser window is a light blue and white striped pattern. The taskbar at the bottom shows various application icons and the system clock indicating 10:10 on 10/10/2012.

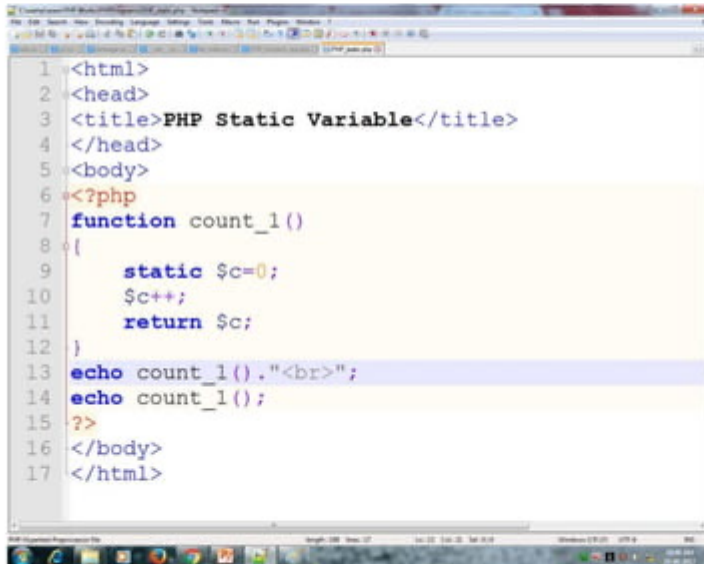
```
1 <html>
2 <head>
3   <title>PHP variables</title>
4 </head>
5 <body>
6 <?php
7   $txt="Hello World!";
8   $x=16;
9   echo $txt;
10  echo $x;
11 ?>
12 </body>
13 </html>
```

# PHP Global Variable

A screenshot of a web browser window showing a PHP script. The browser's address bar displays a local file path. The page content is the raw PHP code, which includes an HTML header with the title 'PHP Global Variable', a PHP block containing a function 'doublevalue()' that doubles a global variable '\$temp', and an 'echo' statement that prints the value of '\$temp'. The code is line-numbered from 1 to 18. The browser's taskbar at the bottom shows various application icons and the system clock indicating 10:07 AM on 10/10/2013.

```
1 <html>
2 <head>
3 <title>PHP Global Variable</title>
4 </head>
5 <body>
6 <?php
7 function doublevalue()
8 {
9     Global $temp;
10    $temp=$temp*2;
11 }
12
13 $temp=5;
14 Doublevalue();
15 echo "\$temp is: $temp";
16 ?>
17 </body>
18 </html>
```

# PHP Static Variable

A screenshot of a web browser window showing a PHP script. The browser's address bar shows a local file path. The page title is "PHP Static Variable". The code defines a function `count_1()` that uses a static variable `$c` to count the number of times the function is called. The function increments `$c` and returns its value. The script calls the function twice, and the output shows the count increasing from 1 to 2.

```
1 <html>
2 <head>
3 <title>PHP Static Variable</title>
4 </head>
5 <body>
6 <?php
7 function count_1()
8 {
9     static $c=0;
10    $c++;
11    return $c;
12 }
13 echo count_1()."<br>";
14 echo count_1();
15 ?>
16 </body>
17 </html>
```



# echo VS print

<i>echo</i> Statement	<i>print</i> Statement
Has no return value.	Has a return value of 1.
Can take multiple parameters.	Can take one argument.
<i>echo</i> is marginally faster than <i>print</i> .	<i>print</i> is slower than <i>echo</i> .
<pre>&lt;html&gt; &lt;head&gt; &lt;title&gt;PHP echo statement&lt;/title&gt; &lt;/head&gt; &lt;body&gt; &lt;?php     echo "Hello World! &lt;br /&gt;"; ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>	<pre>&lt;html&gt; &lt;head&gt; &lt;title&gt;PHP print statement&lt;/title&gt; &lt;/head&gt; &lt;body&gt; &lt;?php     print "Hello World! &lt;br /&gt;"; ?&gt; &lt;/body&gt; &lt;/html&gt;</pre>

# PHP String Functions

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

## PHP String Functions (2)

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

## PHP String Functions (3)

Function	Description	Example
<b>strrev()</b>	Reverses a string.	<pre>&lt;?php echo strrev("Hello World!"); ?&gt;</pre> ldlrOw olleH
<b>strchr()</b>	Finds the first occurrence of a string inside another string and returns the rest of the string.	<pre>&lt;?php echo strchr("Hello world!", "world"). "&lt;br&gt;"; echo strchr("Hello world!", 111); ?&gt;</pre> world! o world!

# PHP String Functions (4)

Function	Description	Example
<b>strcmp()</b>	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).	<pre>&lt;?php echo strcmp("Hello world!", "Hello world!"); ?&gt;</pre> 0
<b>strncmp()</b>	String comparison of the first n characters (case-sensitive).	<pre>&lt;?php echo strncmp("Hello world!", "Hello earth!", 6); ?&gt;</pre> 0

# PHP String Functions (5)

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

# PHP String Functions (6)

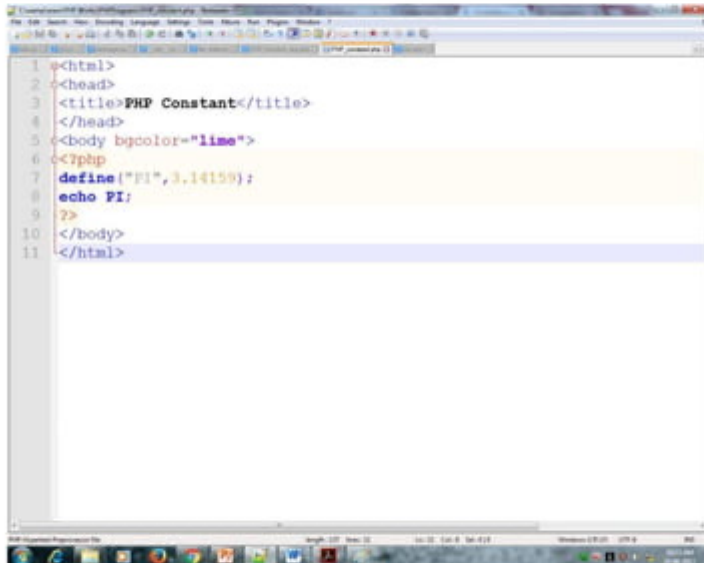
Function	Description	Example
<b>wordwrap()</b>	Wraps a string to a given number of characters.	<pre>&lt;?php \$str = "An example on a long word is: Supercalifragulistic"; echo wordwrap(\$str,15,"&lt;br&gt;",TRUE); ?&gt;</pre> An example on a long word is: Supercalifragul istic

# PHP String Functions (7)

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



# PHP Constants

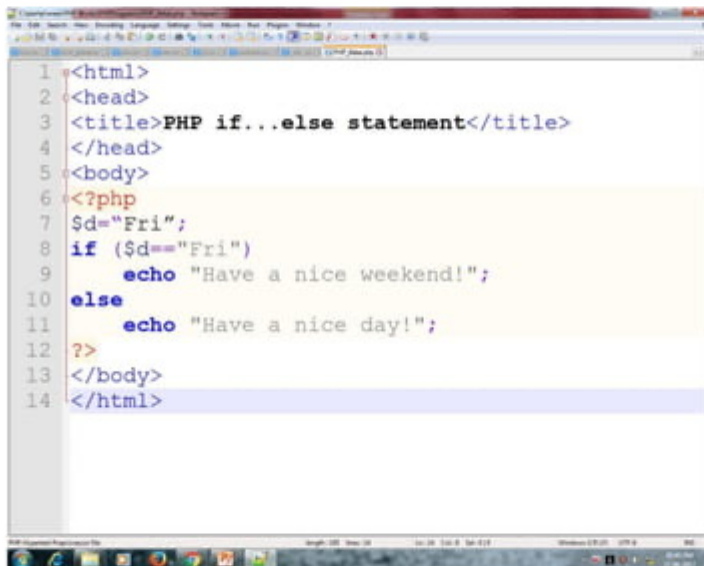
A screenshot of a web browser window showing a PHP page. The page title is "PHP Constant". The body has a light green background. The PHP code defines a constant "PI" with the value "3.14159" and echoes it. The browser's address bar shows the file path "C:\xampp\htdocs\php\php\_constants.php".

```
1 <html>
2 <head>
3 <title>PHP Constant</title>
4 </head>
5 <body bgcolor="lime">
6 <?php
7     define("PI", 3.14159);
8     echo PI;
9 >
10 </body>
11 </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

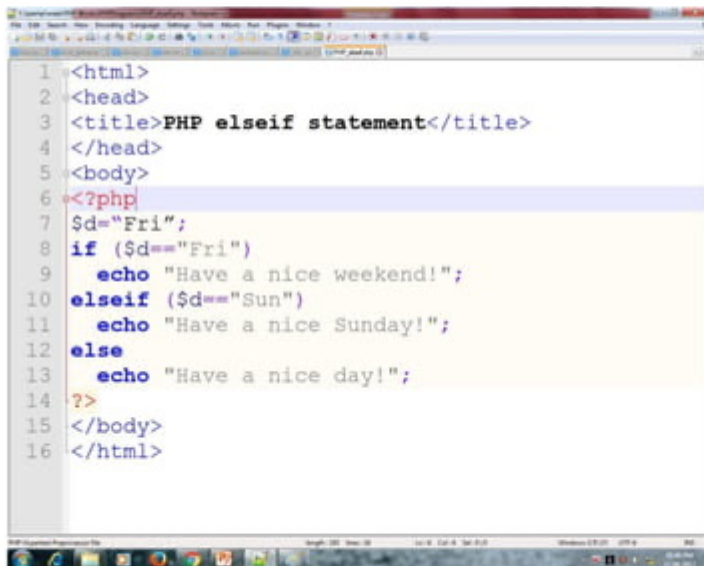


The screenshot shows a Windows Notepad++ window with a PHP file named 'index.php'. The code is as follows:

```
1 <html>
2 <head>
3   <title>PHP if...else statement</title>
4 </head>
5 <body>
6 <?php
7   $d="Fri";
8   if ($d=="Fri")
9     echo "Have a nice weekend!";
10  else
11    echo "Have a nice day!";
12 ?>
13 </body>
14 </html>
```

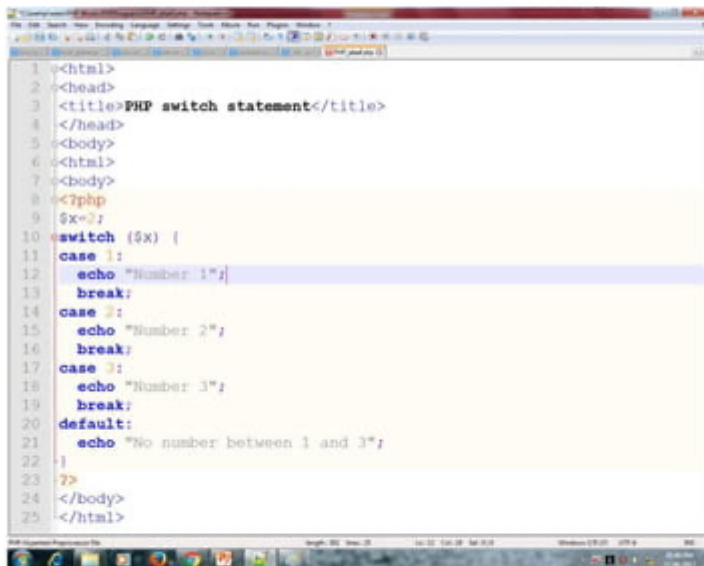
The code is displayed with syntax highlighting: HTML tags are in blue, PHP tags are in red, and the PHP code is in black. The 'if...else' block is highlighted in yellow. The window title bar shows 'index.php - Notepad++' and the status bar at the bottom shows 'Ln 18, Col 9, Line 18'.

# PHP *elseif* Statement

A screenshot of a Notepad++ text editor window. The title bar reads "Notepad++ - C:\Program Files\Notepad++\notepad++.exe". The menu bar includes File, Edit, Search, Run, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and Help. The toolbar contains icons for file operations and editing. The text area shows PHP code with line numbers 1 through 16 on the left. The code is: 1 <html>, 2 <head>, 3 <title>PHP elseif statement</title>, 4 </head>, 5 <body>, 6 <?php, 7 \$d="Fri";, 8 if (\$d=="Fri"), 9 echo "Have a nice weekend!";, 10 elseif (\$d=="Sun"), 11 echo "Have a nice Sunday!";, 12 else, 13 echo "Have a nice day!";, 14 ?>, 15 </body>, 16 </html>. The code from line 6 to 14 is highlighted in yellow. The status bar at the bottom shows "UTF-8", "CRLF", "Ln: 16, Col: 5, Sel: 0,0", "Windows-1252", "UTF-8", and "80". The Windows taskbar is visible at the very bottom with various application icons and the system clock showing 12:45 PM on 12/18/2012.

```
1 <html>
2 <head>
3 <title>PHP elseif statement</title>
4 </head>
5 <body>
6 <?php
7 $d="Fri";
8 if ($d=="Fri")
9     echo "Have a nice weekend!";
10 elseif ($d=="Sun")
11     echo "Have a nice Sunday!";
12 else
13     echo "Have a nice day!";
14 ?>
15 </body>
16 </html>
```

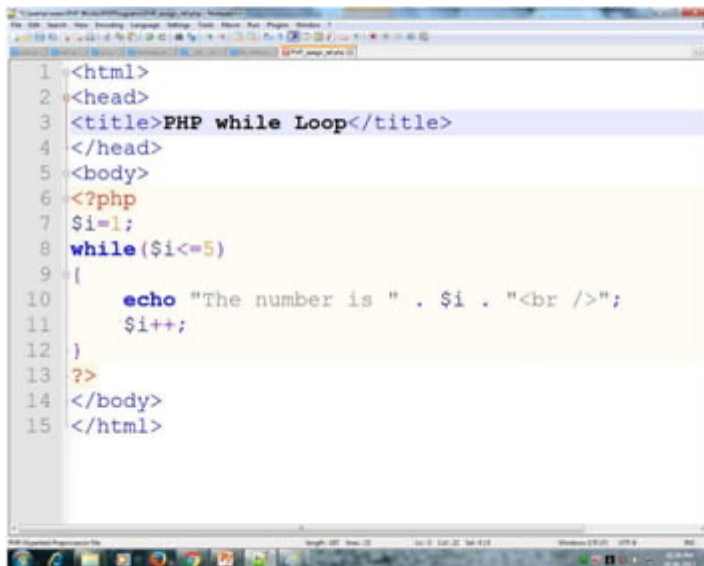
# PHP *switch* Statement



The screenshot shows a Windows Notepad++ editor window with a PHP script. The script uses a switch statement to check the value of a variable \$x. The code is as follows:

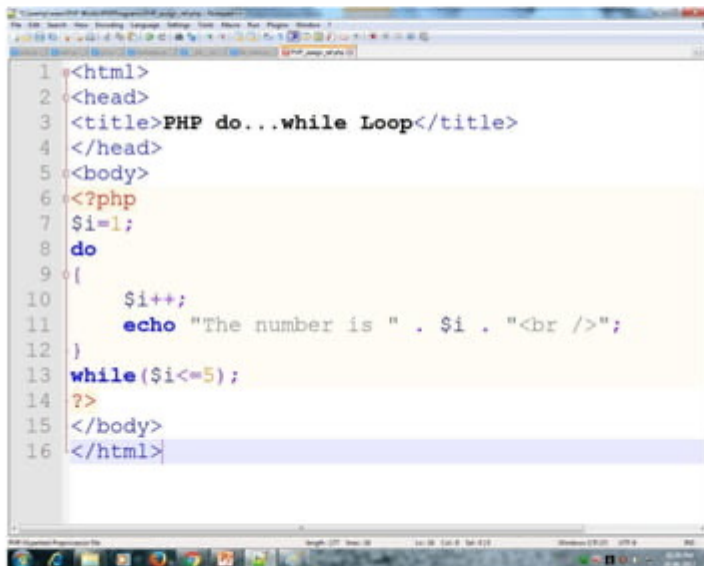
```
1 <html>
2 <head>
3 <title>PHP switch statement</title>
4 </head>
5 <body>
6 <html>
7 <body>
8 <?php
9 $x=2;
10 switch ($x) {
11 case 1:
12     echo "Number 1";
13     break;
14 case 2:
15     echo "Number 2";
16     break;
17 case 3:
18     echo "Number 3";
19     break;
20 default:
21     echo "No number between 1 and 3";
22 }
23 ?>
24 </body>
25 </html>
```

# PHP *while* Loop

A screenshot of a web browser window showing a PHP script. The browser's address bar shows a local file path. The page content is the output of the PHP code, which is not yet rendered. The code is displayed in a syntax-highlighted editor. The code defines an HTML page titled "PHP while Loop" and contains a PHP block that uses a while loop to print the numbers 1 through 5, each on a new line.

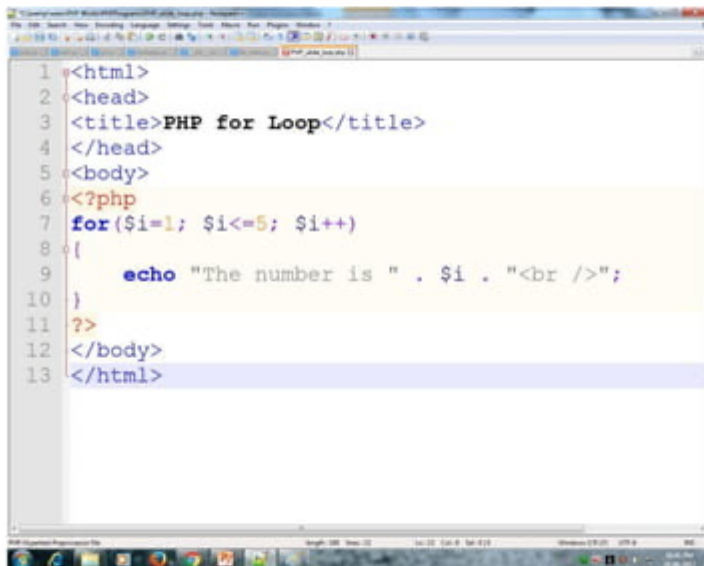
```
1 <html>
2 <head>
3 <title>PHP while Loop</title>
4 </head>
5 <body>
6 <?php
7 $i=1;
8 while ($i<=5)
9 {
10     echo "The number is " . $i . "<br />";
11     $i++;
12 }
13 ?>
14 </body>
15 </html>
```

# PHP *do...while* Loop



```
1 <html>
2 <head>
3 <title>PHP do...while Loop</title>
4 </head>
5 <body>
6 <?php
7 $i=1;
8 do
9 {
10     $i++;
11     echo "The number is " . $i . "<br />";
12 }
13 while ($i<=5);
14 ?>
15 </body>
16 </html>
```

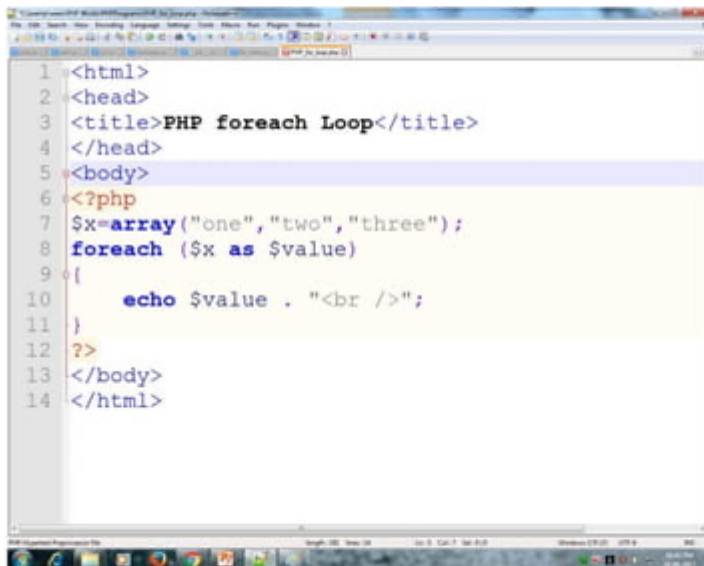
# PHP *for* Loop

A screenshot of a web browser window showing a PHP script. The browser's address bar displays a local file path. The page content is the raw PHP code, which includes HTML headers and a for loop that prints numbers 1 through 5. The code is as follows:

```
1 <html>
2 <head>
3   <title>PHP for Loop</title>
4 </head>
5 <body>
6 <?php
7   for($i=1; $i<=5; $i++)
8   {
9     echo "The number is " . $i . "<br />";
10  }
11  ?>
12 </body>
13 </html>
```



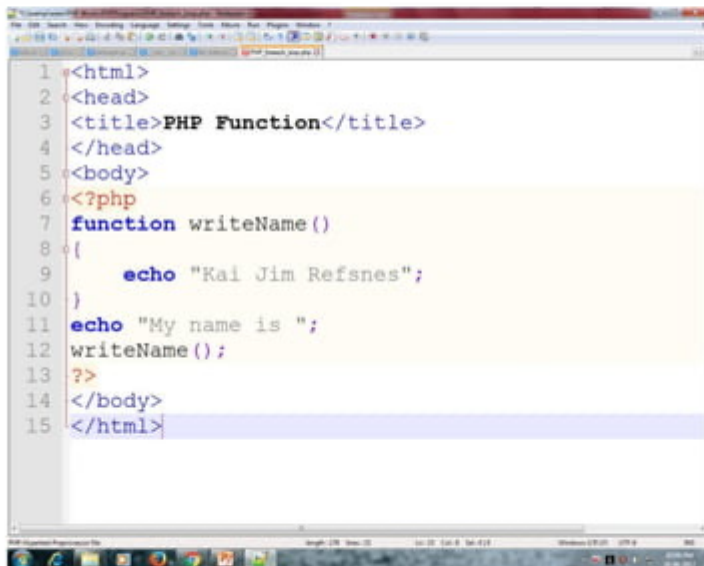
# PHP *foreach* Loop



The screenshot shows a web browser window with a single page. The page title is "PHP foreach Loop". The page content consists of three lines of text, each on a new line: "one", "two", and "three". The text is displayed in a monospaced font. The browser's address bar shows the file path "C:\Program Files\Internet Explorer\iexplore.exe". The Windows taskbar is visible at the bottom of the screen.

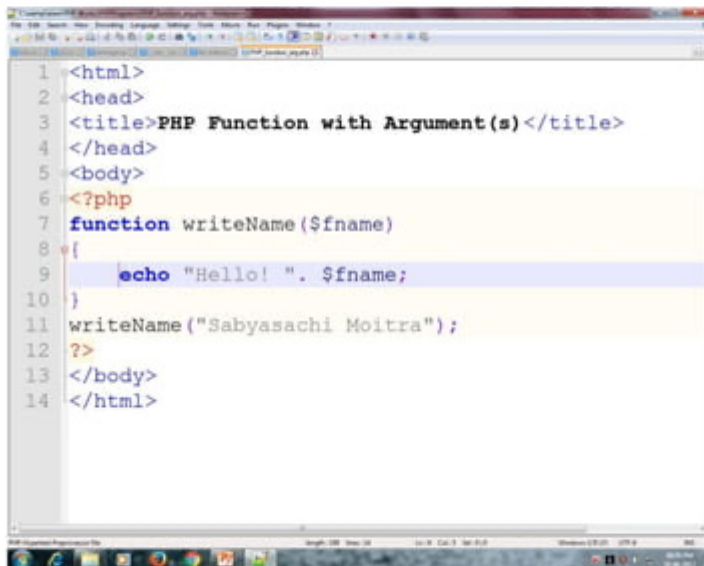
```
1 <html>
2 <head>
3 <title>PHP foreach Loop</title>
4 </head>
5 <body>
6 <?php
7 $x=array("one","two","three");
8 foreach ($x as $value)
9 {
10     echo $value . "<br />";
11 }
12 ?>
13 </body>
14 </html>
```

# PHP Function

A screenshot of a web browser window displaying the output of a PHP script. The browser's address bar shows a local file path. The page content is the result of the PHP code: "Kai Jim Refsnes" followed by "My name is " and a blank space. The browser's status bar at the bottom indicates the page is from "localhost" and was last modified on 12/12/2012.

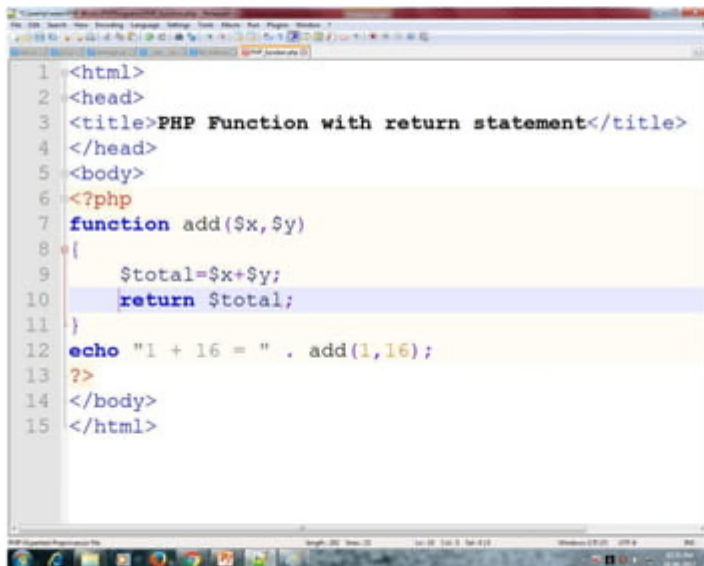
```
1 <html>
2 <head>
3   <title>PHP Function</title>
4 </head>
5 <body>
6 <?php
7   function writeName()
8   {
9     echo "Kai Jim Refsnes";
10  }
11  echo "My name is ";
12  writeName();
13 ?>
14 </body>
15 </html>
```

# PHP Function with Argument(s)



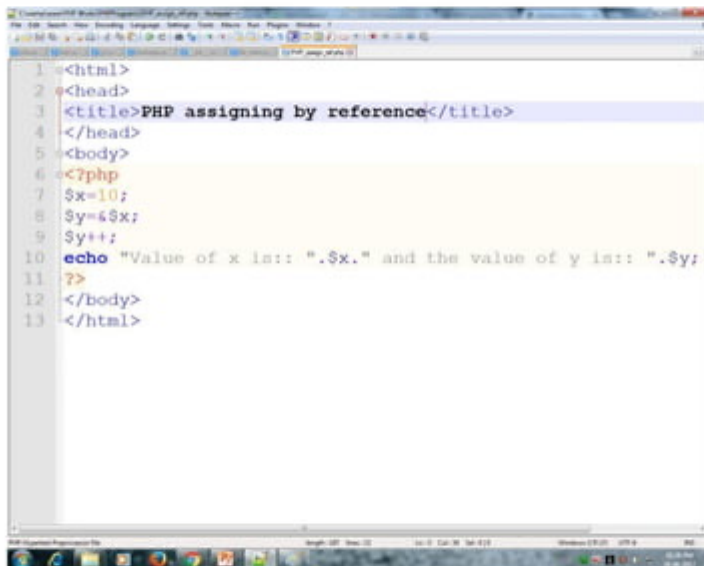
```
1 <html>
2 <head>
3 <title>PHP Function with Argument(s)</title>
4 </head>
5 <body>
6 <?php
7 function writeName($fname)
8 {
9     echo "Hello! ". $fname;
10 }
11 writeName("Sabyasachi Moitra");
12 ?>
13 </body>
14 </html>
```

# PHP Function with *return* Statement



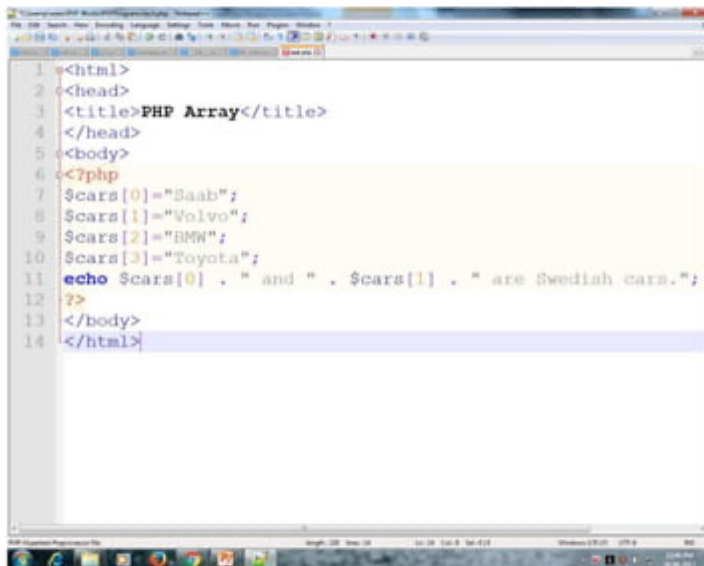
```
1 <html>
2 <head>
3 <title>PHP Function with return statement</title>
4 </head>
5 <body>
6 <?php
7 function add($x,$y)
8 {
9     $total=$x+$y;
10    return $total;
11 }
12 echo "1 + 16 = " . add(1,16);
13 ?>
14 </body>
15 </html>
```

# PHP Assigning by Reference

A screenshot of a web browser window showing a PHP script. The browser's address bar shows a local file path. The page title is "PHP assigning by reference". The script contains HTML tags for the head and body, and PHP code that initializes a variable \$x to 10, multiplies it by 4 to get \$y, increments \$y, and then echoes the values of \$x and \$y.

```
1 <html>
2 <head>
3 <title>PHP assigning by reference</title>
4 </head>
5 <body>
6 <?php
7 $x=10;
8 $y=4$x;
9 $y++;
10 echo "Value of x is:: ".$x." and the value of y is:: ".$y;
11 ?>
12 </body>
13 </html>
```

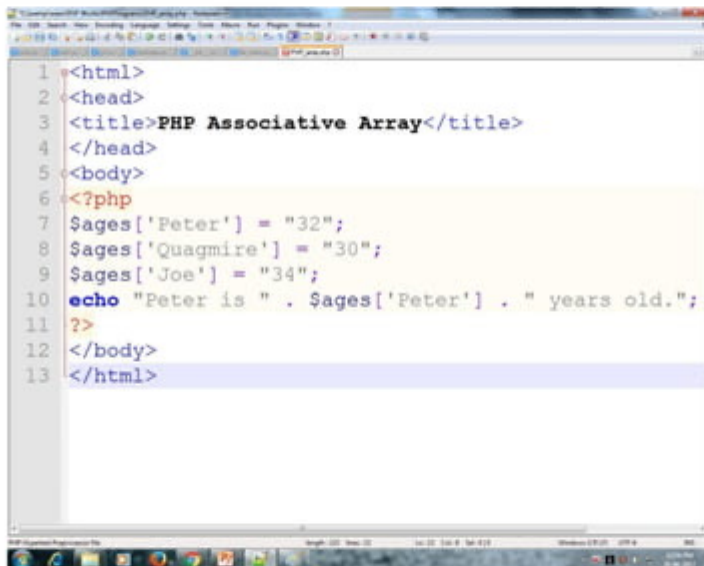
# PHP Array



The screenshot shows a web browser window with a single tab. The address bar shows a local file path. The page content is the output of a PHP script. The script defines an array named \$cars with four elements: "Saab", "Volvo", "BMW", and "Toyota". It then uses the echo statement to print the first two elements, \$cars[0] and \$cars[1], separated by a period and the word "and". The output displayed in the browser is "Saab. and Volvo. are Swedish cars."

```
1 <html>
2 <head>
3 <title>PHP Array</title>
4 </head>
5 <body>
6 <?php
7 $cars[0]="Saab";
8 $cars[1]="Volvo";
9 $cars[2]="BMW";
10 $cars[3]="Toyota";
11 echo $cars[0] . " and " . $cars[1] . " are Swedish cars.";
12 ?>
13 </body>
14 </html>
```

# PHP Associative Array



The screenshot shows a web browser window with a single page of HTML and PHP code. The code defines an associative array named \$Sages with three elements: 'Peter' with value '32', 'Quagmire' with value '30', and 'Joe' with value '34'. It then uses the echo statement to output the age of Peter. The browser's address bar shows a local file path. The Windows taskbar at the bottom indicates the system time is 10:00 AM on 10/10/2012.

```
1 <html>
2 <head>
3 <title>PHP Associative Array</title>
4 </head>
5 <body>
6 <?php
7 $Sages['Peter'] = "32";
8 $Sages['Quagmire'] = "30";
9 $Sages['Joe'] = "34";
10 echo "Peter is " . $Sages['Peter'] . " years old.";
11 ?>
12 </body>
13 </html>
```

# POST vs GET

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



# \$\_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>
```

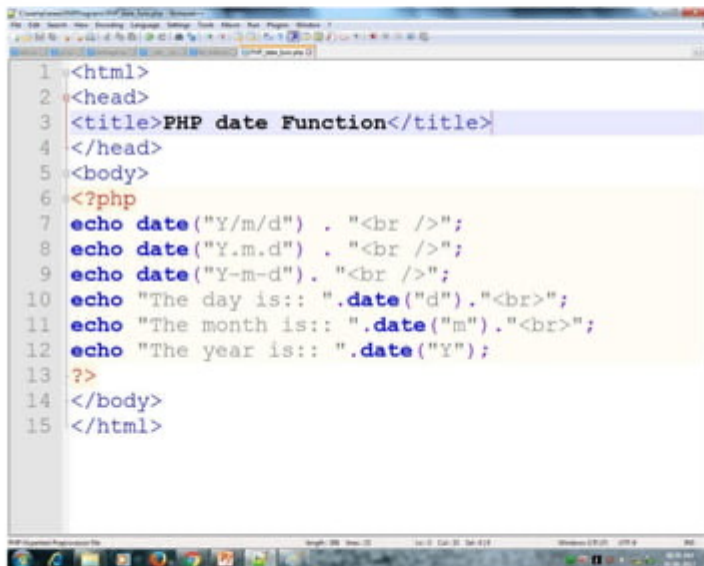
# \$\_REQUEST Variable

---

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

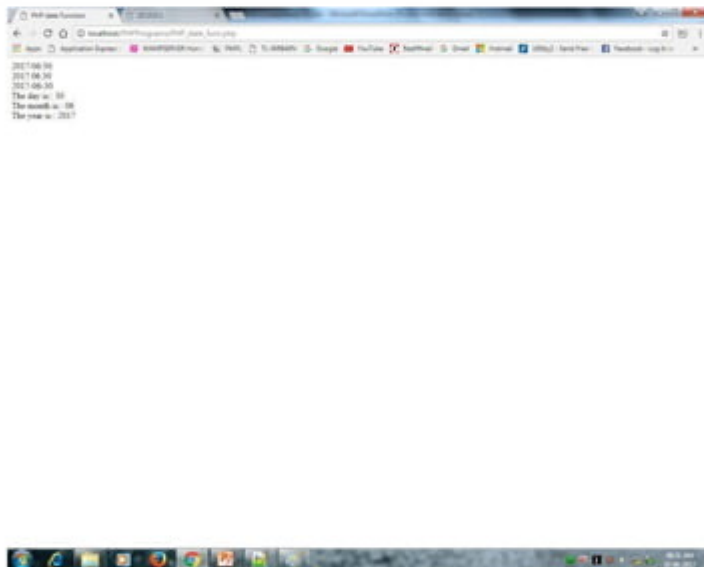
```
<html>
<head>
<title>PHP $_REQUEST variable</title>
</head>
<body bgcolor="lime">
<form action="PHP_request.php" method="get">
Name: <input type="text" name="name"><br>
<input type="submit" name="submit"
value="Submit">
</form>
Welcome <?php
if(isset($_REQUEST['submit'])) {echo
$_REQUEST["name"];}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

# PHP *date()* Function



```
1 <html>
2 <head>
3 <title>PHP date Function</title>
4 </head>
5 <body>
6 <?php
7 echo date("Y/m/d") . "<br />";
8 echo date("Y.m.d") . "<br />";
9 echo date("Y-m-d") . "<br />";
10 echo "The day is:: ".date("d")."<br>";
11 echo "The month is:: ".date("m")."<br>";
12 echo "The year is:: ".date("Y");
13 ?>
14 </body>
15 </html>
```

# Output



# 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.

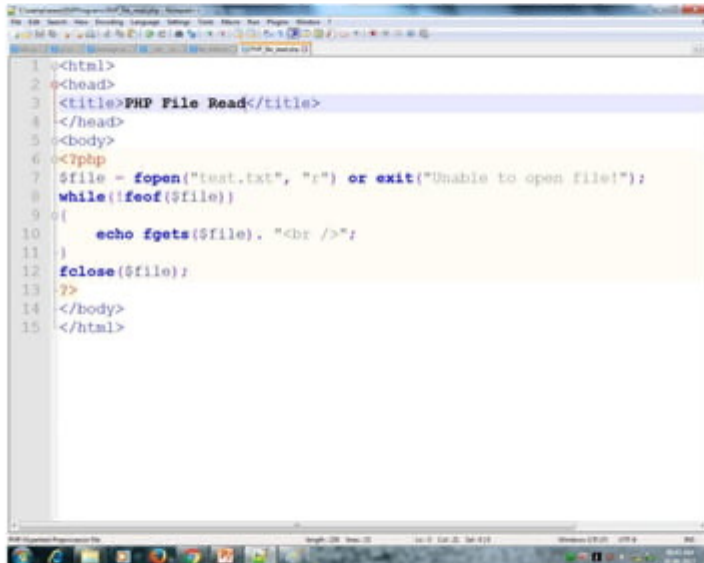
**M:** A short textual representation of a month, three letters.

**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.
<pre>&lt;?php echo "Different Date Formats:&lt;br&gt;"; include("PHP_date_func.php "); ?&gt;</pre>	<pre>&lt;?php echo "Different Date Formats:&lt;br&gt;"; require("PHP_date_func.php "); ?&gt;</pre>

# PHP File Read

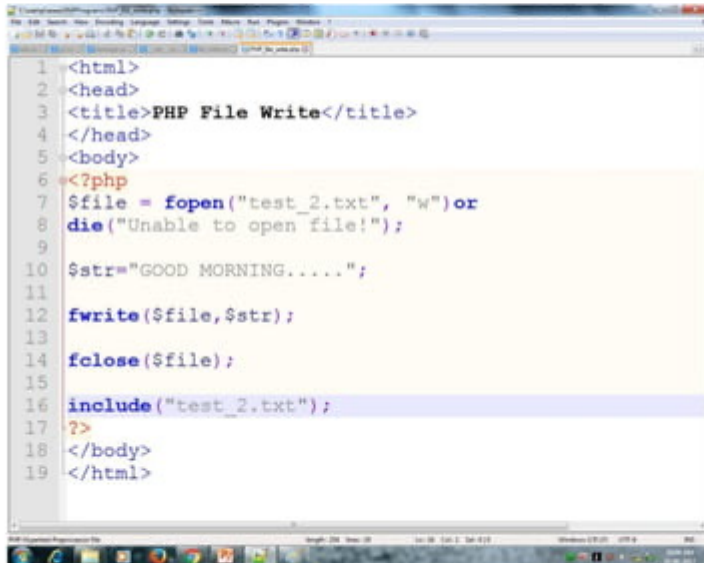
A screenshot of a Notepad++ text editor window. The title bar reads 'Untitled - Notepad++'. The menu bar includes File, Edit, Search, New, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and Help. The toolbar contains icons for file operations and editing. The text area shows PHP code with line numbers 1 through 15 on the left margin. The code is as follows:

```
1 <html>
2 <head>
3   <title>PHP File Read</title>
4 </head>
5 <body>
6 <?php
7   $file = fopen("test.txt", "r") or exit("Unable to open file!");
8   while(!feof($file))
9   {
10      echo fgets($file). "<br />";
11   }
12   fclose($file);
13 >?>
14 </body>
15 </html>
```

The status bar at the bottom shows 'Full Screen: Disabled', 'Length: 209 (lines: 15)', 'Ln: 11, Col: 25, Sel: 410', 'Windows: 1/1/2011 11:59:59', and '88'.



# PHP File Write

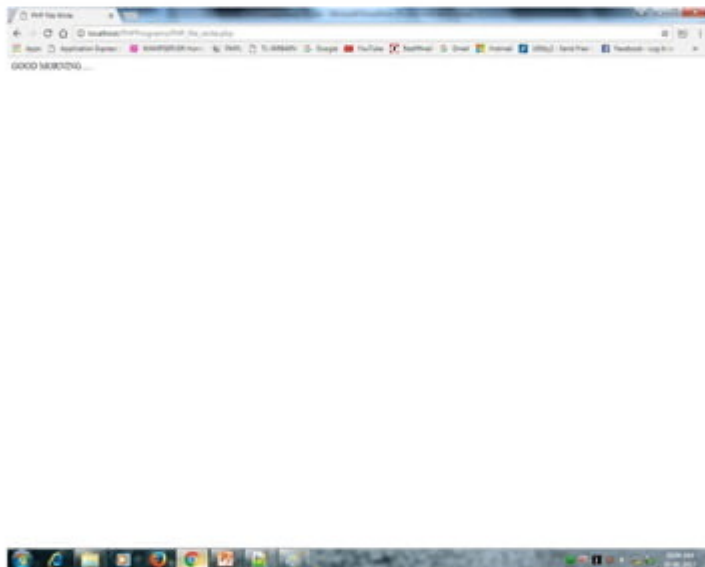
A screenshot of a Notepad++ text editor window. The title bar reads "Untitled - Notepad++". The menu bar includes File, Edit, Search, Run, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Windows, and Help. The toolbar contains icons for file operations and editing. The code is written in a syntax-highlighted format: HTML tags are in blue, PHP opening and closing tags are in orange, and PHP code is in black. The code is as follows:

```
1 <html>
2 <head>
3 <title>PHP File Write</title>
4 </head>
5 <body>
6 <?php
7 $file = fopen("test_2.txt", "w") or
8 die("Unable to open file!");
9
10 $str="GOOD MORNING....";
11
12 fwrite($file,$str);
13
14 fclose($file);
15
16 include("test_2.txt");
17 ?>
18 </body>
19 </html>
```

The status bar at the bottom shows "Notepad++ v5.9.6 (64-bit)", "UTF-8", "100%", and "Ln 10, Col 1, Sel 0/0". The Windows taskbar is visible at the very bottom.

# Output

---



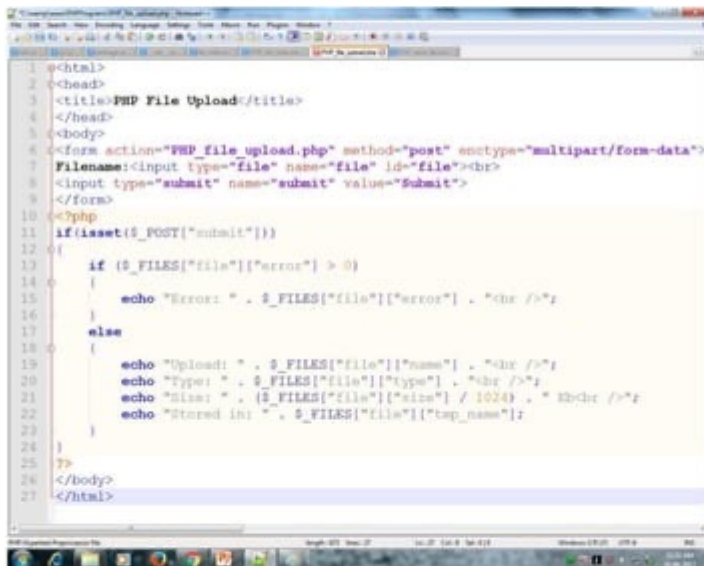
# PHP File Opening Modes

Mode	Description
<b>r</b>	Read only. Starts at the beginning of the file. Error if the file doesn't exist.
<b>r+</b>	Read/Write. Starts at the beginning of the file.
<b>w</b>	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist.
<b>w+</b>	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
<b>a</b>	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist.
<b>a+</b>	Read/Append. Preserves file content by writing to the end of the file.
<b>x</b>	Write only. Creates a new file. Returns FALSE and an error if file already exists.
<b>x+</b>	Read/Write. Creates a new file. Returns FALSE and an error if file already exists.

# PHP File Upload

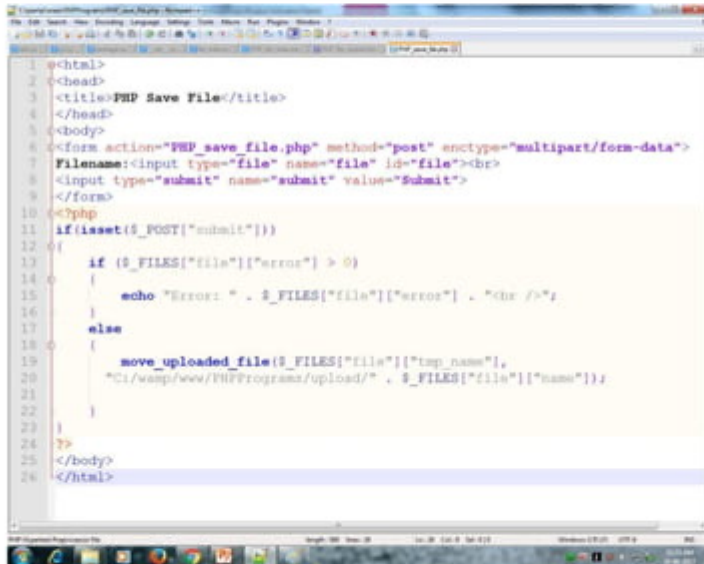


```
1 <html>
2 <head>
3 <title>PHP File Upload</title>
4 </head>
5 <body>
6 <form action="PHP_file_upload.php" method="post" enctype="multipart/form-data">
7   Filename:<input type="file" name="file" id="file"><br>
8   <input type="submit" name="submit" value="Submit">
9 </form>
10 <?php
11 if(isset($_POST["submit"]))
12 {
13     if ($_FILES["file"]["error"] > 0)
14     {
15         echo "Error: " . $_FILES["file"]["error"] . "<br />";
16     }
17     else
18     {
19         echo "Upload: " . $_FILES["file"]["name"] . "<br />";
20         echo "Type: " . $_FILES["file"]["type"] . "<br />";
21         echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
22         echo "Stored in: " . $_FILES["file"]["tmp_name"];
23     }
24 }
25 ?>
26 </body>
27 </html>
```



	Description
<b>enctype="multipart/form-data"</b>	<ul style="list-style-type: none"> <li>The <i>enctype</i> attribute specifies how the form-data should be encoded when submitting it to the server.</li> <li><i>multipart/form-data</i> value specifies that no characters are encoded.</li> </ul>
<b>\$_FILES</b>	An associate double dimension array that keeps all the information related to the uploaded file.
<b>\$_FILES['file']['error']</b>	The error code associated with the uploaded file.
<b>\$_FILES['file']['name']</b>	The actual name of the uploaded file.
<b>\$_FILES['file']['type']</b>	The MIME type of the uploaded file.
<b>\$_FILES['file']['size']</b>	The size in bytes of the uploaded file.
<b>\$_FILES['file']['tmp_name']</b>	The uploaded file in the temporary directory on the web server.

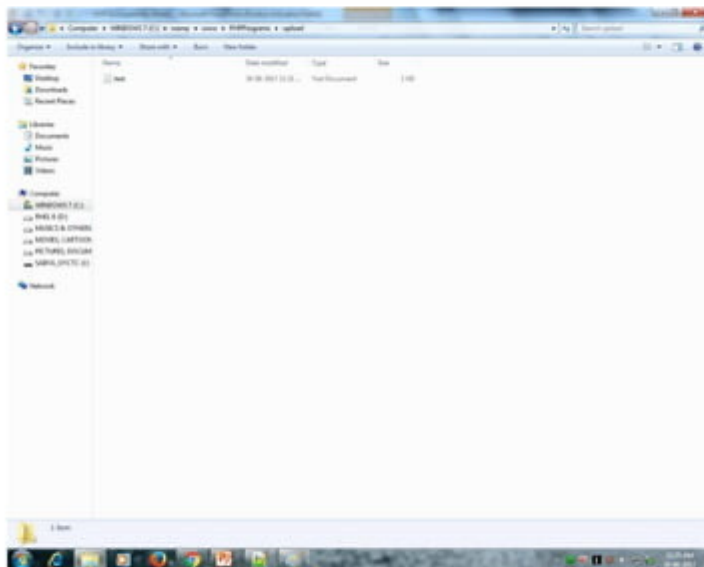
# PHP Save File



```
1 <html>
2 <head>
3 <title>PHP Save File</title>
4 </head>
5 <body>
6 <form action="PHP_save_file.php" method="post" enctype="multipart/form-data">
7   Filename:<input type="file" name="file" id="file"><br>
8   <input type="submit" name="submit" value="Submit">
9 </form>
10 <?php
11 if(isset($_POST["submit"]))
12 {
13     if ($_FILES["file"]["error"] > 0)
14     {
15         echo "Error: " . $_FILES["file"]["error"] . "<br />";
16     }
17     else
18     {
19         move_uploaded_file($_FILES["file"]["tmp_name"],
20             "C:/wamp/www/PHPPrograms/upload/" . $_FILES["file"]["name"]);
21     }
22 }
23 ?>
24 </body>
25 </html>
```

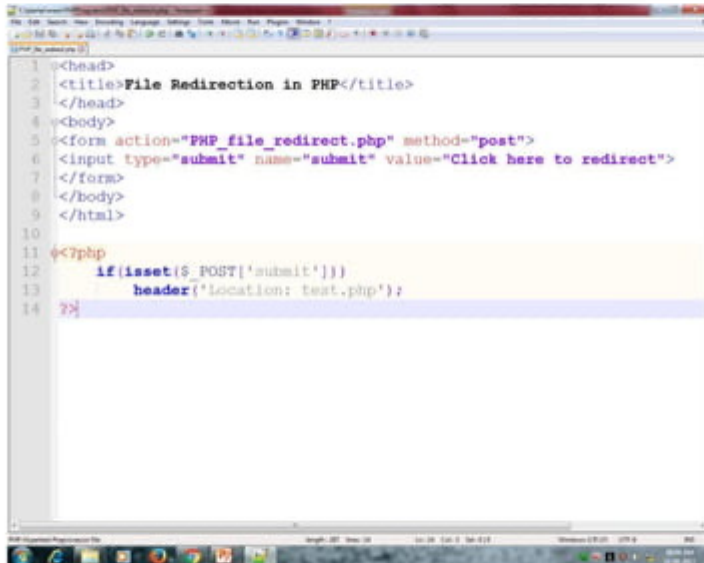


# Output



	Description
<b>move_uploaded_file(file,newloc)</b>	Moves an uploaded file to a new location.

# File Redirection in PHP

A screenshot of a web browser window displaying a PHP script for file redirection. The browser's address bar shows a local file path. The page content is a simple HTML form with a single submit button labeled "Click here to redirect". The PHP code is visible in the background, showing a conditional statement that checks if the submit button was clicked and then issues a header to redirect to "test.php".

```
1 <head>
2 <title>File Redirection in PHP</title>
3 </head>
4 <body>
5 <form action="PHP_file_redirect.php" method="post">
6 <input type="submit" name="submit" value="Click here to redirect">
7 </form>
8 </body>
9 </html>
10
11 <?php
12     if(isset($_POST['submit']))
13         header('location: test.php');
14 >>
```

# 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	<pre>&lt;?php setcookie("user", "Alex Porter"); ?&gt;</pre>
Retrieve a Cookie	<pre>&lt;?php // Print a cookie echo \$_COOKIE["user"];  // A way to view all cookies print_r(\$_COOKIE); ?&gt;</pre>
Delete a Cookie	<pre>&lt;?php // set the expiration date to after one hour setcookie("user", "", time()+3600); ?&gt;</pre>

# 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 & 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
<pre>&lt;?php unset(\$_SESSION['views']); ?&gt;</pre>	The <i>unset()</i> function is used to free the specified session variable.
<pre>&lt;?php session_destroy(); ?&gt;</pre>	The <i>session_destroy()</i> function is used to completely destroy the session. <i>session_destroy()</i> 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
<b>filter_var()</b>	Filters a single variable with a specified filter.
<b>filter_var_array()</b>	Filter several variables with the same or different filters.
<b>filter_input()</b>	Get one input variable and filters it.
<b>filter_input_array()</b>	Get several input variables and filter them with the same or different filters.

# The *filter\_var()* Function

	Explanation
<pre>&lt;?php \$int = 123; if(!filter_var(\$int, FILTER_VALIDATE_INT)) { echo("Integer is not valid"); } else { echo("Integer is valid"); } ?&gt;</pre>	<ul style="list-style-type: none"><li>• The code uses the "FILTER_VALIDATE_INT" filter to filter the variable.</li><li>• Since the integer is valid, the output of the code above will be: "Integer is valid".</li><li>• If we try with a variable that is not an integer (like "123abc"), the output will be: "Integer is not valid".</li></ul>

# 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

	Explanation
<pre>&lt;?php \$var=300; \$int_options = array(     "options"=&gt;array     (         "min_range"=&gt;0,         "max_range"=&gt;256     ) ); if(!filter_var(\$var, FILTER_VALIDATE_INT, \$int_options)) {     echo("Integer is not valid"); } else {     echo("Integer is valid"); } ?&gt;</pre>	<ul style="list-style-type: none"><li>• The <i>filter_var()</i> function uses <i>min_range</i> &amp; <i>max_range</i> options.</li><li>• 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.</li><li>• 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".</li></ul>

# 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

	Explanation
<pre>&lt;?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"; } } ?&gt;</pre>	<p>The example has an input (email) sent to it using the "GET" method:</p> <ul style="list-style-type: none"><li>• Check if an "email" input variable of the "GET" type exist.</li><li>• If the input variable exists, check if it is a valid e-mail address.</li></ul>

# Sanitize URL by PHP

	Explanation
<pre>&lt;?php if(!filter_has_var(INPUT_POST , "url")) {     echo("Input type does not exist"); } else {     \$url = filter_input(INPUT_POST,     "url",     FILTER_SANITIZE_URL); } ?&gt;</pre>	<p>The example above has an input (url) sent to it using the "POST" method:</p> <ul style="list-style-type: none"><li>• Check if the "url" input variable of the "POST" type exists.</li><li>• If the input variable exists, sanitize (take away invalid characters) and store it in the \$url variable.</li><li>- <i>If the input variable is a string like this</i> <i>"http://www.W3ååSchøøools.com/"</i> <i>, the \$url variable after the sanitizing will look like this:</i> <i>http://www.W3Schools.com/</i></li></ul>

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

## PHP MySQL Connectivity (2)

Type	Code
MySQLi (POP)	<pre>&lt;?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"; ?&gt;</pre>

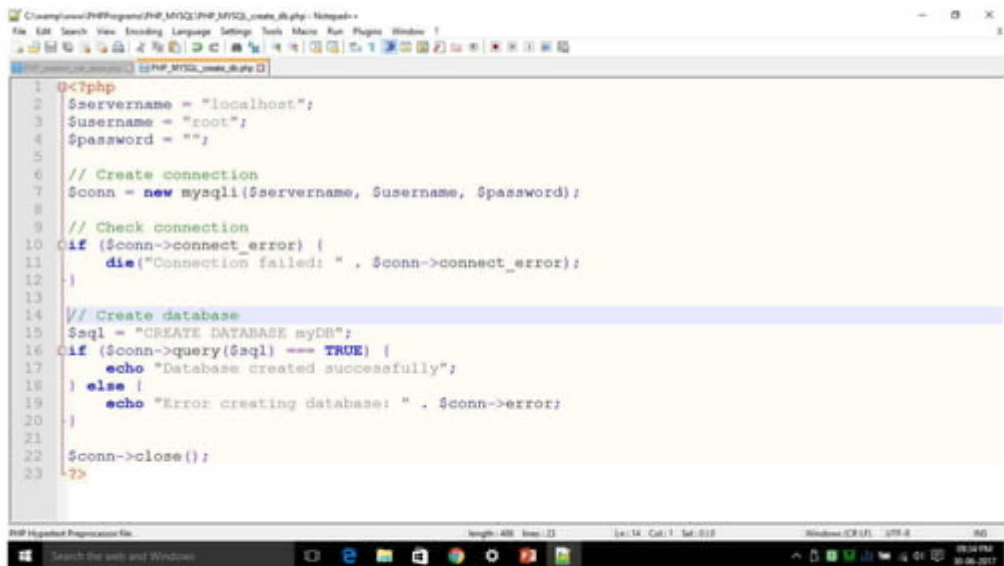
# PHP MySQL Connectivity (3)

Type	Code
PDO	<pre>&lt;?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- &gt;setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);     echo "Connected successfully"; } catch(PDOException \$e) {     echo "Connection failed: " . \$e- &gt;getMessage(); } ?&gt;</pre>

# 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

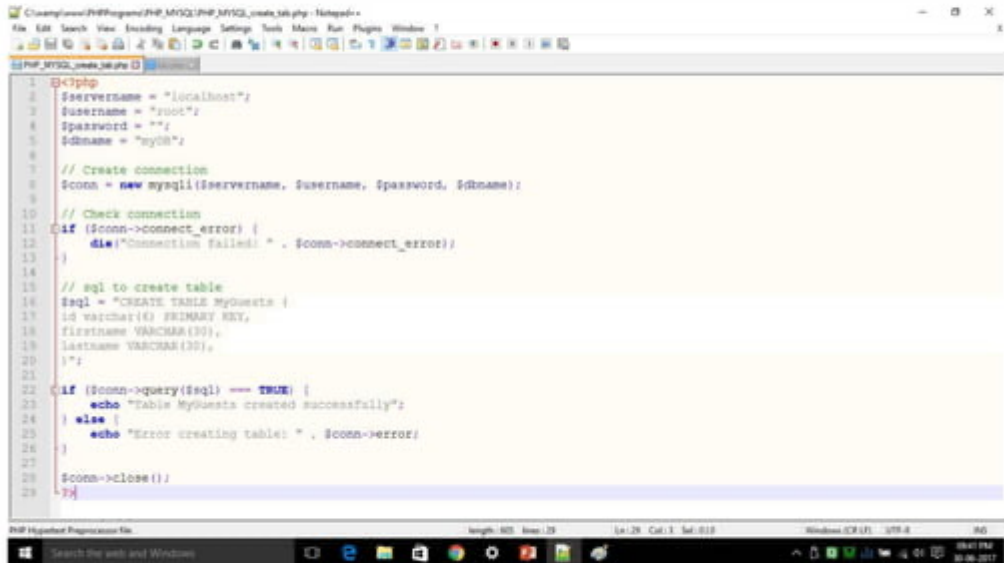


```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5
6 // Create connection
7 $conn = new mysqli($servername, $username, $password);
8
9 // Check connection
10 if ($conn->connect_error) {
11     die("Connection failed: " . $conn->connect_error);
12 }
13
14 // Create database
15 $sql = "CREATE DATABASE myDB";
16 if ($conn->query($sql) === TRUE) {
17     echo "Database created successfully";
18 } else {
19     echo "Error creating database: " . $conn->error;
20 }
21
22 $conn->close();
23 ?>
```

The screenshot shows a Notepad++ window titled "C:\xampp\htdocs\PHP\MySQL\PHP\_MYSQL\_create\_db.php - Notepad++". The code is a PHP script that connects to a MySQL database using the mysqli extension. It sets the server name to "localhost", the username to "root", and the password to an empty string. It then creates a new MySQLi connection object. A check is performed to see if the connection was successful; if not, it dies with an error message. Next, it executes a SQL query to create a database named "myDB". If the query is successful, it echoes "Database created successfully"; otherwise, it echoes an error message. Finally, it closes the connection.



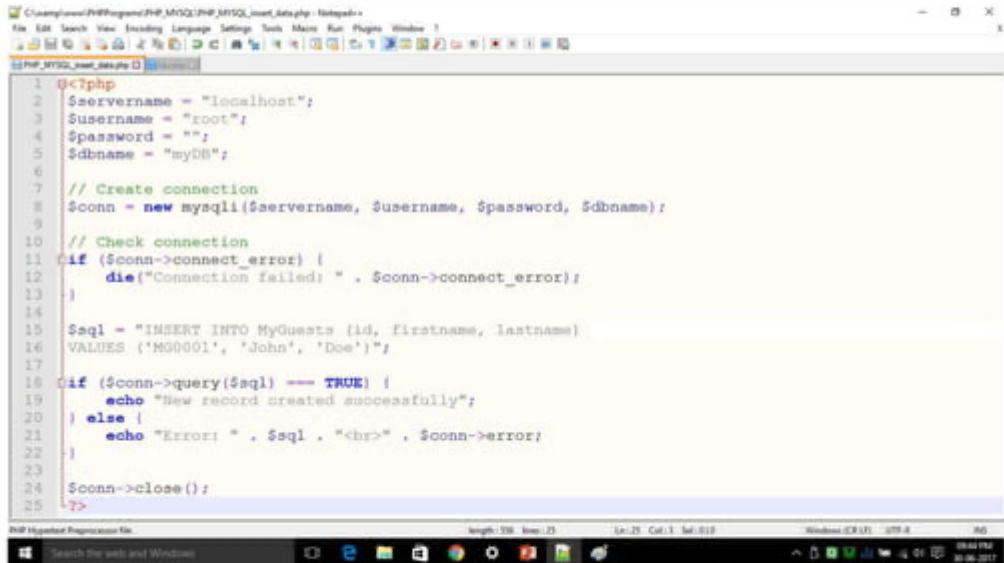
# PHP MySQL Create Table



```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "myDB";
6
7 // Create connection
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Check connection
11 if ($conn->connect_error) {
12     die("Connection failed: " . $conn->connect_error);
13 }
14
15 // sql to create table
16 $sql = "CREATE TABLE myGuests (
17     id varchar(4) PRIMARY KEY,
18     firstname VARCHAR(30),
19     lastname VARCHAR(30),
20     )";
21
22 if ($conn->query($sql) === TRUE) {
23     echo "Table myGuests created successfully";
24 } else {
25     echo "Error creating table: " . $conn->error;
26 }
27
28 $conn->close();
29 ?>
```

PHP Hypertext Preprocessor File Length: 602 Size: 129 La: 28 Col: 1 Sel: 019 Windows (C#) UTF-8 BOM

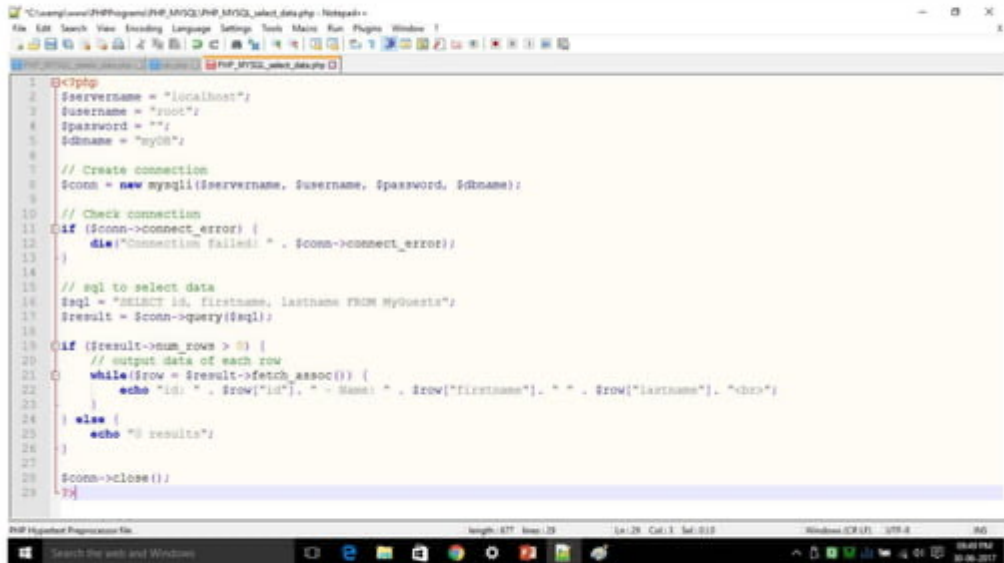
# PHP MySQL Insert Data



```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "myDB";
6
7 // Create connection
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Check connection
11 if ($conn->connect_error) {
12     die("Connection failed: " . $conn->connect_error);
13 }
14
15 $sql = "INSERT INTO MyGuests (id, firstname, lastname)
16 VALUES ('MG0001', 'John', 'Doe')";
17
18 if ($conn->query($sql) === TRUE) {
19     echo "New record created successfully";
20 } else {
21     echo "Error: " . $sql . "<br>" . $conn->error;
22 }
23
24 $conn->close();
25 >
```

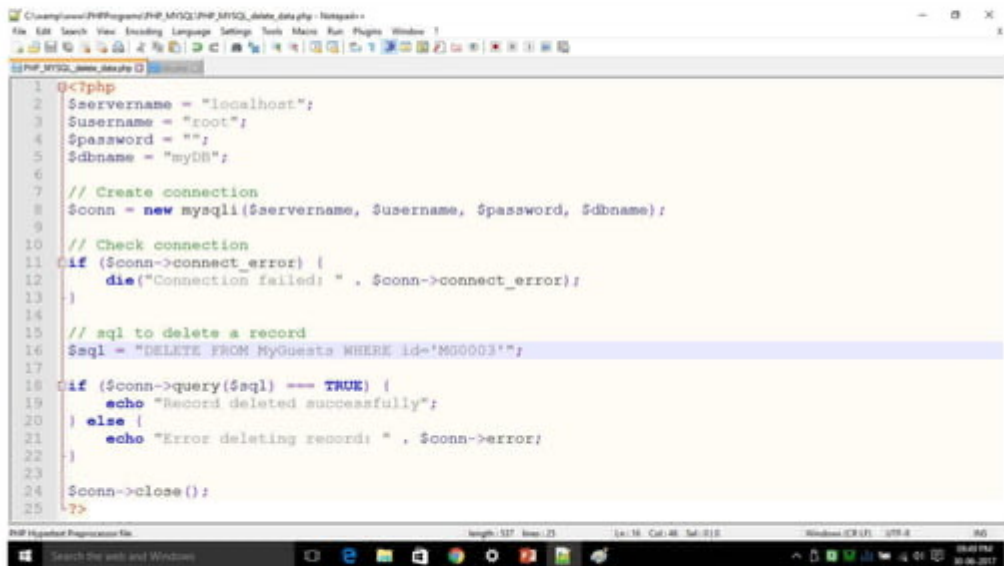
PHP Hypertext Preprocessor File length: 556 Size: 25 Ln: 25 Col: 1 Sel: 0.0 Windows (C#) 100% 865  
Search the web and Windows 15:44 PM 30-04-2017

# PHP MySQL Select Data



```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "myDB";
6
7 // Create connection
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Check connection
11 if ($conn->connect_error) {
12     die("Connection failed: " . $conn->connect_error);
13 }
14
15 // sql to select data
16 $sql = "SELECT id, firstname, lastname FROM MyGuests";
17 $result = $conn->query($sql);
18
19 if ($result->num_rows > 0) {
20     // output data of each row
21     while($row = $result->fetch_assoc()) {
22         echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "<br>";
23     }
24 } else {
25     echo "0 results";
26 }
27
28 $conn->close();
29 }
```

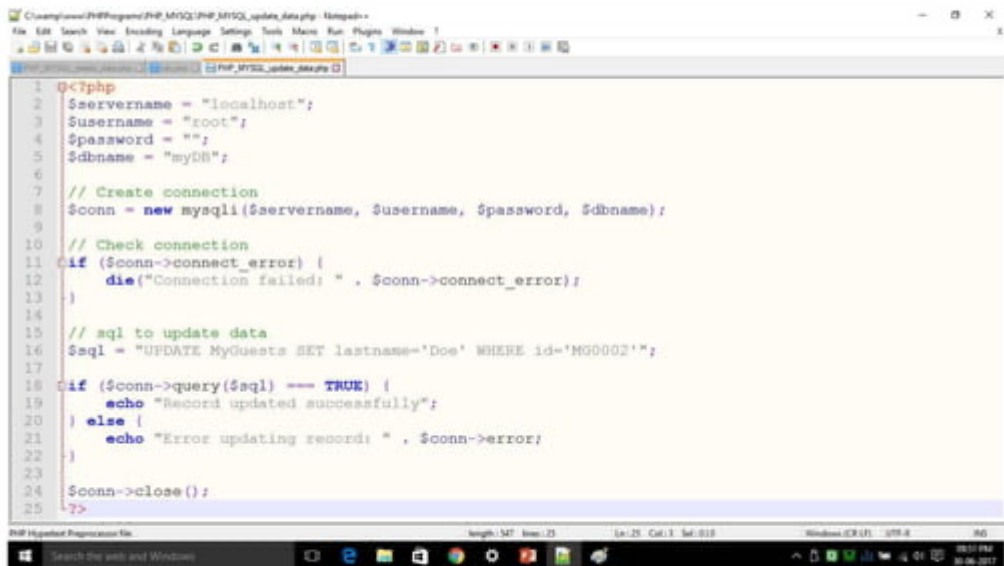
# PHP MySQL Delete Data



```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "myDB";
6
7 // Create connection
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Check connection
11 if ($conn->connect_error) {
12     die("Connection failed: " . $conn->connect_error);
13 }
14
15 // sql to delete a record
16 $sql = "DELETE FROM MyGuests WHERE id='MOOOO3'";
17
18 if ($conn->query($sql) === TRUE) {
19     echo "Record deleted successfully";
20 } else {
21     echo "Error deleting record: " . $conn->error;
22 }
23
24 $conn->close();
25 >
```

PHP Hypertext Preprocessor File length: 517 bytes Line: 25 Col: 46 Tab: 015 Windows (C#) 100% 865  
Search the web and Windows 15:45 PM 31-08-2017

# PHP MySQL Update Data



The screenshot shows a web browser window with the address bar displaying 'C:\xampp\htdocs\PHPProgram\PHP\_MYSQL\PHP\_MYSQL\_update\_data.php - Htmledit+'. The browser's toolbar includes standard navigation buttons and a search bar. The main content area displays the source code of a PHP script, which is a file named 'PHP\_MYSQL\_update\_data.php'. The script is a PHP file that connects to a MySQL database and updates a record. The code is as follows:

```
1 <?php
2 $servername = "localhost";
3 $username = "root";
4 $password = "";
5 $dbname = "myDB";
6
7 // Create connection
8 $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Check connection
11 if ($conn->connect_error) {
12     die("Connection failed: " . $conn->connect_error);
13 }
14
15 // sql to update data
16 $sql = "UPDATE MyGuests SET lastname='Doe' WHERE id='MG0002'";
17
18 if ($conn->query($sql) === TRUE) {
19     echo "Record updated successfully";
20 } else {
21     echo "Error updating record: " . $conn->error;
22 }
23
24 $conn->close();
25 ?>
```

The status bar at the bottom of the browser window shows 'PHP Hypertext Preprocessor File', 'length: 547', 'line: 25', 'Col: 1', 'Sel: 0:0', 'Windows (C# 1)', 'UTF-8', and 'MS'. The taskbar at the bottom of the screen shows the Windows Start button, a search bar, and several application icons including Edge, File Explorer, and the Settings app. The system clock in the bottom right corner shows '10:51 PM' and '31-Mar-2017'.

# 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, 4<sup>th</sup> Revised Edition, BPB Publications, 2010