**PHP** is a scripting language designed to fill the gap between SSI (Server Side Includes) and Perl, intended for the Web environment. Its principal application is the implementation of Web pages having dynamic content. PHP has gained quite a following in recent times, and it is one of the frontrunners in the Open Source software movement. Its popularity derives from its C-like syntax, and its simplicity. The newest version of PHP is 5.5 and it is heavily recommended to always use the newest version for better security, performance and of course features.

If you've been to a website that prompts you to login, you've probably encountered a server-side scripting language. Due to its market saturation, this means you've probably come across PHP. PHP was designed by Rasmus Lerdorf to display his resume online and to collect data from his visitors.

Basically, PHP allows a static webpage to become dynamic. "PHP" is an acronym that stands for "**P**HP: **H**ypertext **P**reprocessor". The word "Preprocessor" means that PHP makes changes before the HTML page is created. This enables developers to create powerful applications that can publish a blog, remotely control hardware, or run a powerful website such as Wikipedia or Wikibooks. Of course, to accomplish something such as this, you need a database application such as MySQL.

Before you embark on the wonderful journey of Server Side Processing, it is recommended that you have a basic understanding of the HyperText Markup Language (HTML). But PHP can also be used to build GUI-driven applications for example by using PHP-GTK.

PHP is used extensively on the Web as a general purpose programming language. PHP interacts with HTML forms, to provide back-end processing of forms, and PHP can be embedded directly into an HTML file.  
  
PHP includes all of the same functionality you are used to in a programming language, including variables, arrays, strings, functions and the ability to interact with databases. PHP is easy to learn if you have experience in at least one other programming language, such as C-Language or Perl.  
  
Many of the programs running on the Web today are built using PHP. All of the major Content Management Systems, such as Joomla, Wordpress and Drupal, as well as the most popular open source forum systems were written in PHP. Once you learn PHP, you can work with these, and many other commercial and open source systems, and will have the ability to modify and extend these systems to suit your own purposes. You will also be able to write your own Web based programs.  
  
This course assumes that you have the prerequisites, specifically, a good, working knowledge of HTML and experience with at least one other high level programming language.

PHP code is contained in a beginning and an ending PHP tag:

Any code placed inside the block is parsed as PHP code. You can embed this into a regular HTML page as follows:



Notice that each line of PHP code ends with a semi-colon. Your page must be named using the .php file extension in order for the server to recognize the embedded PHP code. So you could name this file myfirst.php.

**Note**: **A Note on Word Processors**

Word processors such as StarOffice Writer, Microsoft Word and Abiword are not optimal for editing PHP files. If you wish to use one for this test script, you must ensure that you save the file as *plain text* or PHP will not be able to read and execute the script.

**Note**: **A Note on Windows Notepad**

If you are writing your PHP scripts using Windows Notepad, you will need to ensure that your files are saved with the *.php* extension. (Notepad adds a *.txt* extension to files automatically unless you take one of the following steps to prevent it.) When you save the file and are prompted to provide a name for the file, place the filename in quotes (i.e. "*hello.php*"). Alternatively, you can click on the 'Text Documents' drop-down menu in the 'Save' dialog box and change the setting to "All Files". You can then enter your filename without quotes.

**Comments**

Comments are pieces of code that the PHP parser skips. When the parser spots a comment, it simply keeps going until the end of the comment without doing anything. PHP offers both one line and multi-line comments.

**One-Line Comments**

One-line comments are comments that start where ever you start them and end at the end of the line. With PHP, you can use both // and # for your one-line comments (# is not commonly used). Those are used mainly to tell the reader what you're doing the next few lines. For example:

//Print the variable $message echo $message;

It's important to understand that a one-line comment doesn't have to 'black out' the whole line, it starts where ever you start it. So it can also be used to tell the reader what a certain variable does:

$message = ""; //This sets the variable $message to an empty string

The *$message = "";* is executed, but the rest of the line is not.

**One-line Comment Issues**

* One-line comments end by either:

1. a newline (an actual newline, not the \n newline mark ) OR:
2. a closing PHP tag of the ?> variety

* If a one-line comment is closed by a closing PHP tag, it will not be commented. The following will thus print out "2":

// echo "1"; ?> echo "2";

**Multi-Line Comments**

This kind of comment can go over as many lines as you'd like, and can be used to state what a function or a class does, or just to contain comments too big for one line. To mark the beginning of a multiline comment, use /\* and to end it, use \*/ . For example:

/\* This is a multiline comment And it will close

When I tell it to. \*/

While declaring variables in PHP, the following points should be noted:

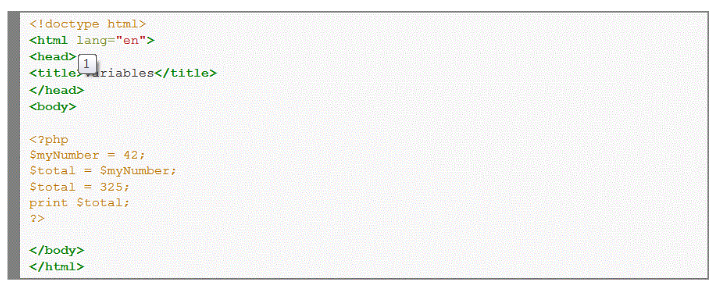
• The name of a variable should start with the $ sign

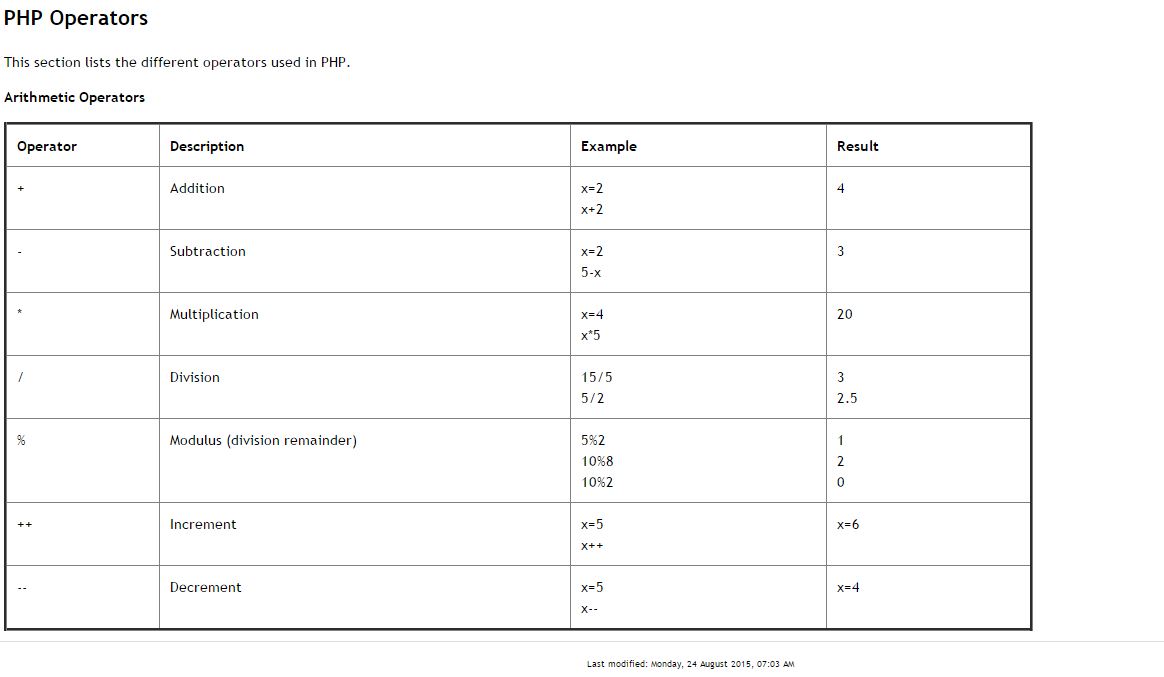
• The name of a variable should start with a letter or the underscore character

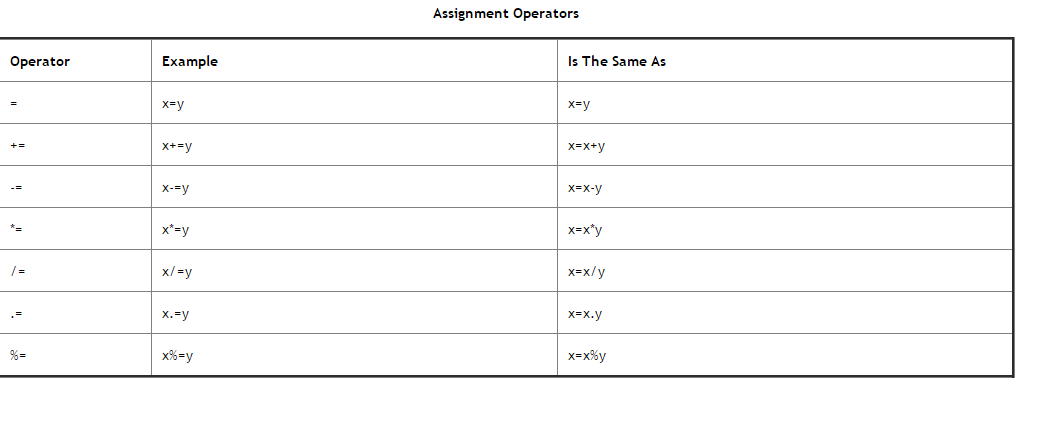
• The name of a variable cannot start with a number

• The name of a variable can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )

• Variable names in PHP are case-sensitive. Hence, $myVar and $MYVAR are two different variables in PHP.







.(dot) is concatenation in PHP!(ex. $text1.$text2)

### Shortcut operators

This snippet demonstrates self-referential shortcut operators. The first such operator is the ++ operator, which increments $x (using the postfix form) by 1 giving it the value 2. After incrementing $x, $y is defined and assigned the value 5. The second shortcut operator is \*=, which takes $y and assigns it the value $y\*$x, or 10. After initializing $z to 180, the subsequent line performs two shortcut operations. Going by order of operations (see manual page below), $y is decremented (using the prefix form) and divided into $z. $z is assigned to the resulting value, 20.



Using PHP Includes (Video)

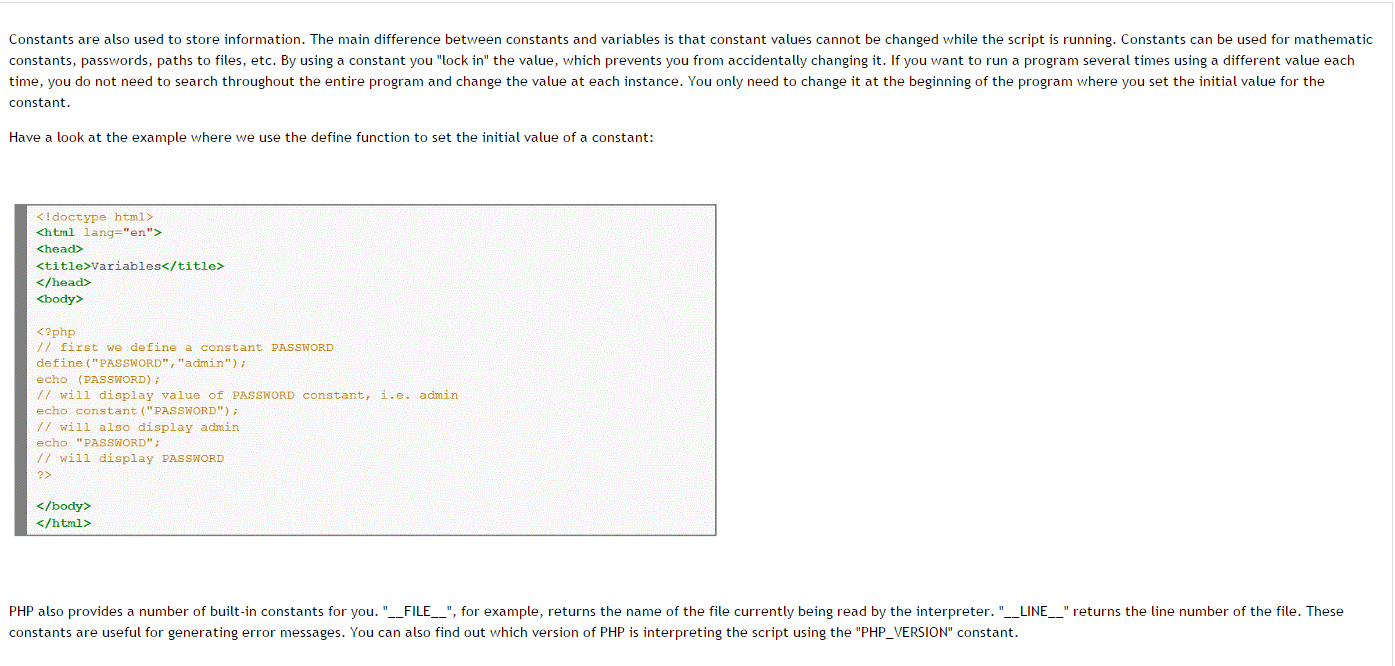
<?php

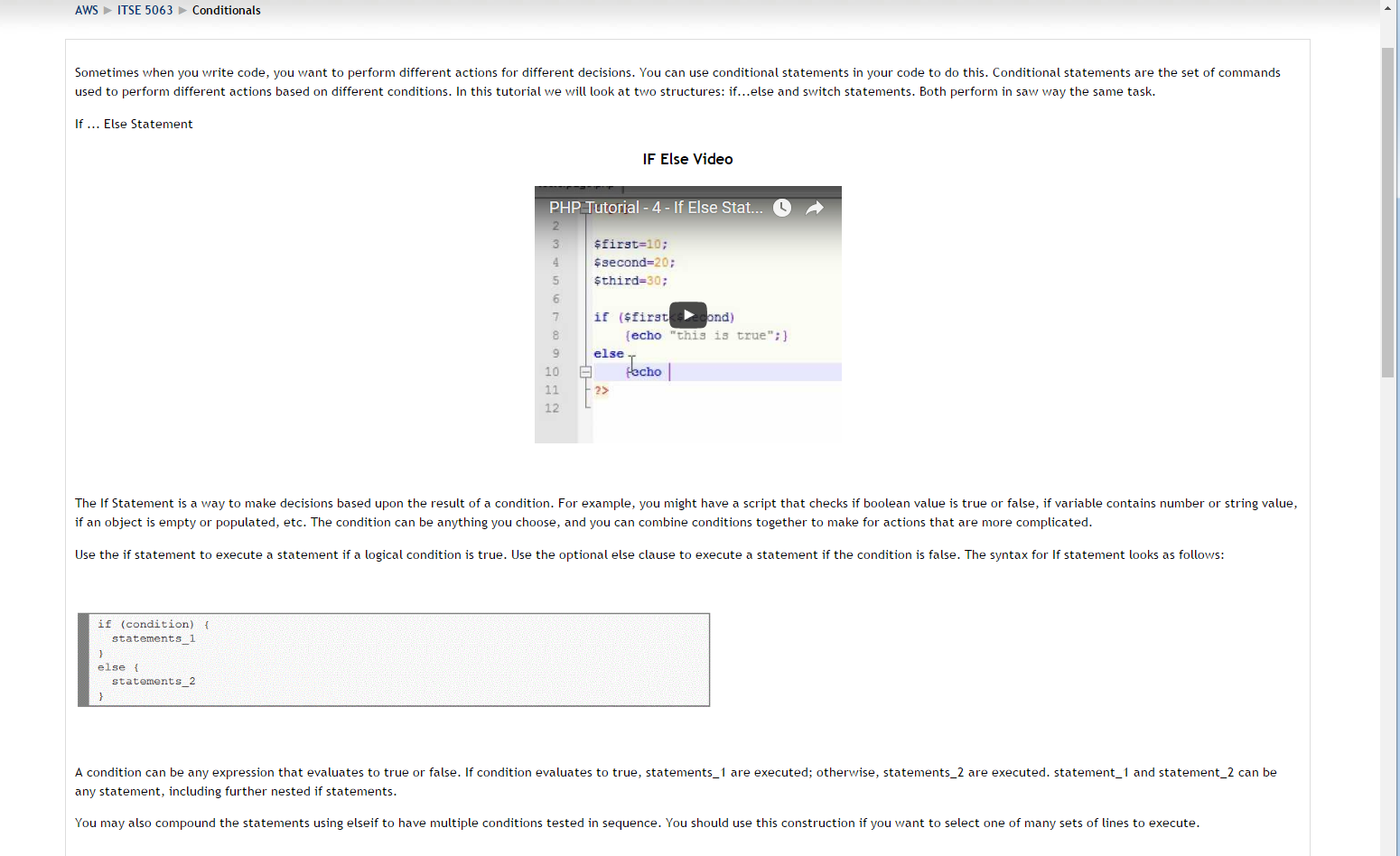
Include ‘filename.php’

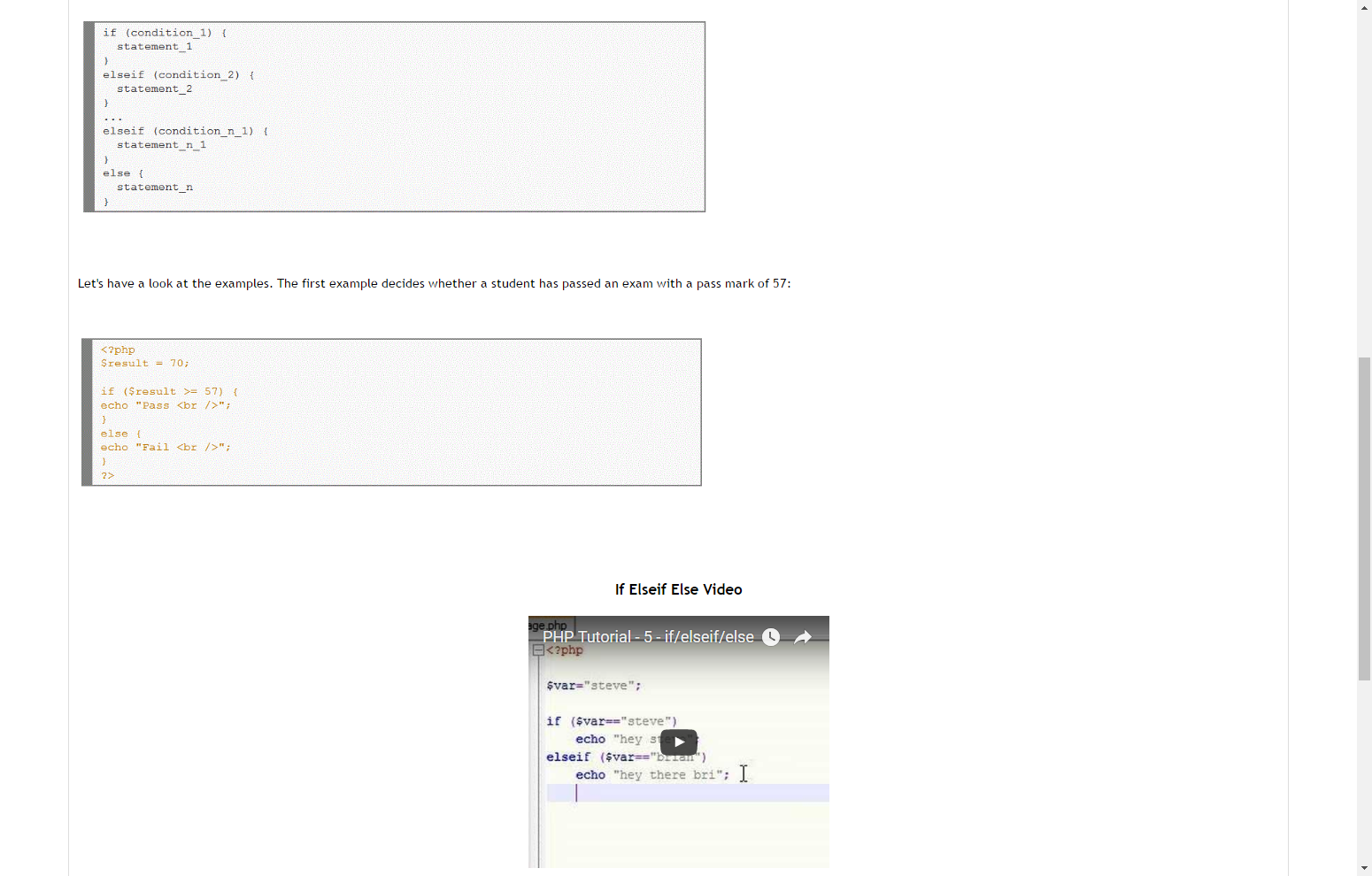
?>

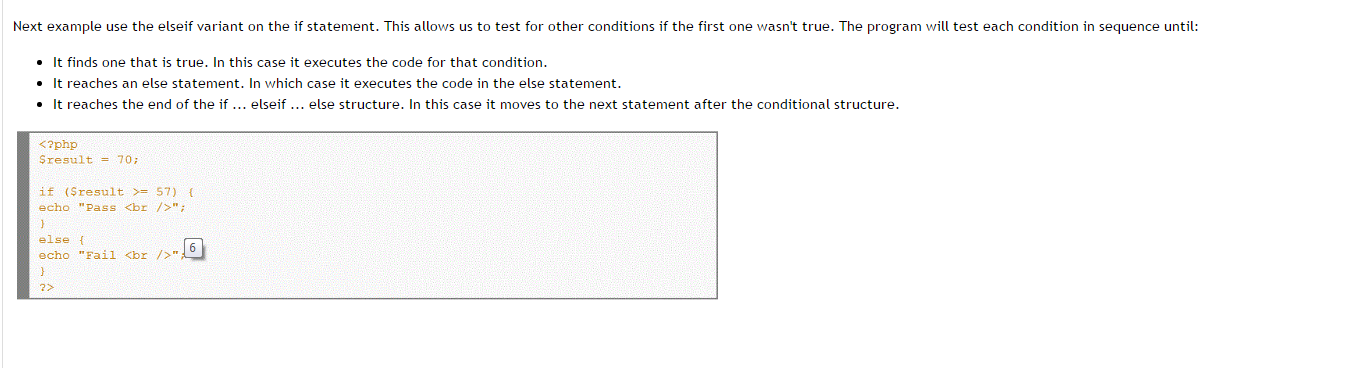
COMPOSER free software can be used to build sites!

Insert this anywhere in an HTML file and the server will execute the content of the filename.php when it is processing this tag.









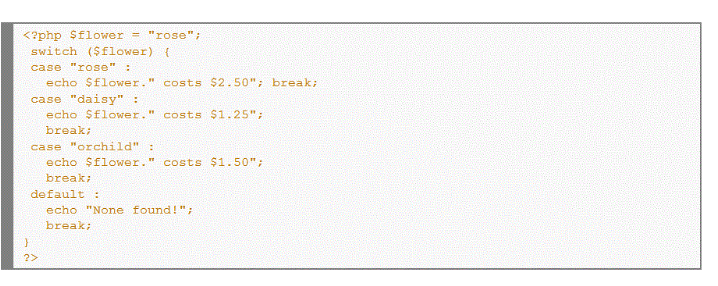
Switch Statement  
  
Switch statements work the same as if statements. However the difference is that they can check for multiple values. Of course you do the same with multiple if..else statements, but this is not always the best approach.

A switch statement allows a program to evaluate an expression and attempt to match the expression's value to a case label. If a match is found, the program executes the associated statement. The syntax for the switch statement as follows:

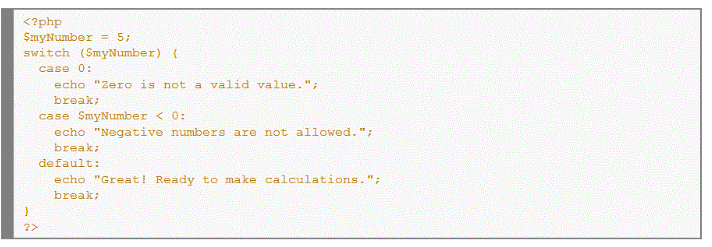


The program first looks for a case clause with a label matching the value of expression and then transfers control to that clause, executing the associated statements. If no matching label is found, the program looks for the optional default clause, and if found, transfers control to that clause, executing the associated statements. If no default clause is found, the program continues execution at the statement following the end of switch. Use break to prevent the code from running into the next case automatically.

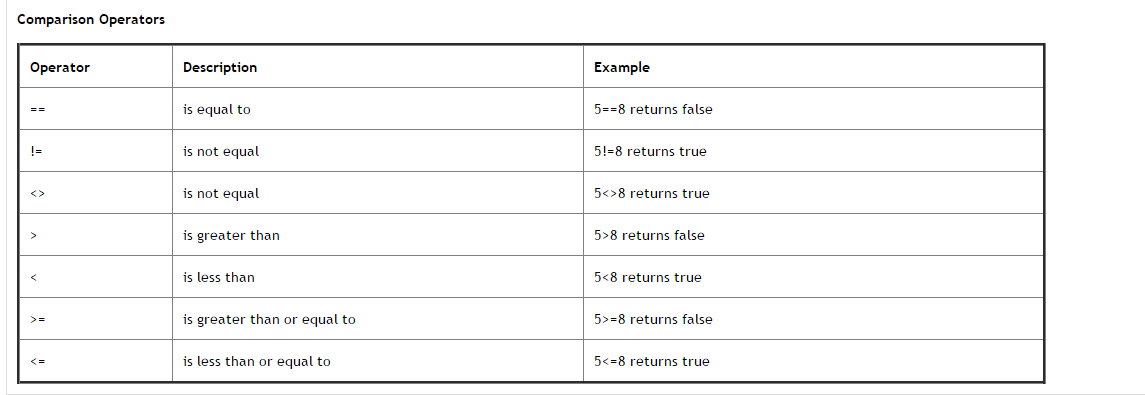
Let's consider an example:

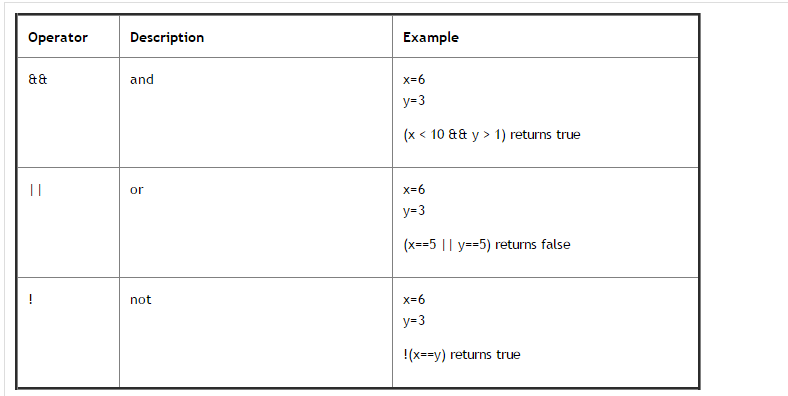


However, in addition to simple equality you can also test the expression for other conditions, such as greater than and less than relationships. The expression you are testing against must be repeated in the case statement. Have a look at the example:



If an expression successfully evaluates to the values specified in more than one case statement, only the first one encountered will be executed. Once a match is made, PHP stops looking for more matches.





Looping is used in programming to perform some operation over and over again. You will remember these constructs from your basic programming class, so I won't go into how they work in detail. The object here is to learn the syntax of how to implement these constructs in PHP. So here they are...

The While Loop

The While statement executes a block of code if and as long as a specified condition evaluates to true. If the condition becomes false, the statements within the loop stop executing and control passes to the statement following the loop. The While loop syntax is as follows:

The block of code associated with the While statement is always enclosed within the { opening and } closing brace symbols to tell PHP clearly which lines of code should be looped through.

While loops are most often used to increment a list where there is no known limit to the number of iterations of the loop.

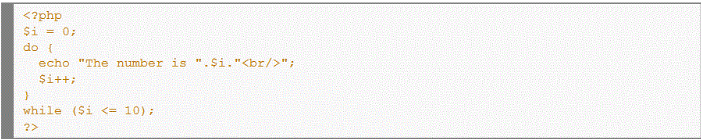
Let's have a look at the examples. The first example defines a loop that starts with i=0. The loop will continue to run as long as the variable i is less than, or equal to 10. i will increase by 1 each time the loop runs:



The Do...While Loop

The Do...While statements are similar to While statements, except that the condition is tested at the end of each iteration, rather than at the beginning. This means that the Do...While loop is guaranteed to run at least once.

The example below will increment the value of i at least once, and it will continue incrementing the variable i as long as it has a value of less than or equal to 10:



The For Loop

The For statement loop is used when you know how many times you want to execute a statement or a list of statements. For this reason, the For loop is known as a definite loop. The syntax of For loops is a bit more complex, though for loops are often more convenient than While loops. The For loop syntax is as follows:

for (initialization; condition; increment)   
{  
*code to be executed;*  
}

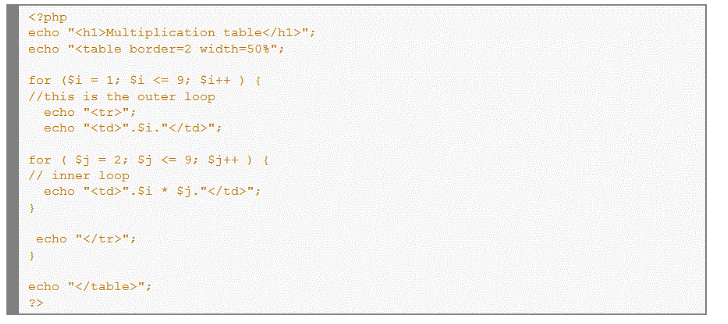
The For statement takes three expressions inside its parentheses, separated by semi-colons. When the For loop executes, the following occurs:

* The initializing expression is executed. This expression usually initializes one or more loop counters, but the syntax allows an expression of any degree of complexity.
* The condition expression is evaluated. If the value of condition is true, the loop statements execute. If the value of condition is false, the For loop terminates.
* The update expression increment executes.
* The statements execute, and control returns to step 2.

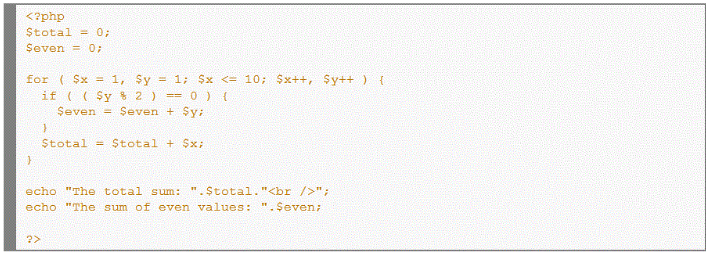
Have a look at the very simple example that prints out numbers from 0 to 10:



Next example generates a multiplication table 2 through 9. Outer loop is responsible for generating a list of dividends, and inner loop will be responsible for generating lists of dividers for each individual number:



And finally, a more complex for block that shows that we can put more complex statements inside the condition:



The Foreach Loop

The Foreach loop is a variation of the For loop and allows you to iterate over elements in an array. Since we haven't discussed arrays yet, please note the general syntax, and then return to this section to study the example in more detail later...

There are two different versions of the Foreach loop.

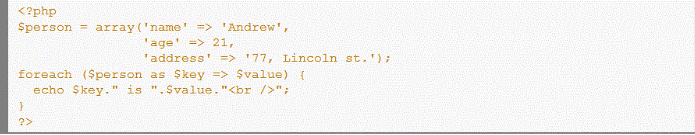
foreach (array as value)  
{  
*code to be executed;*  
}  
  
foreach (array as key => value)  
{  
*code to be executed;*  
}

The example below demonstrates the Foreach loop that will print the values of the given array:

[image here]

PHP executes the body of the loop once for each element of $email in turn, with $value set to the current element. Elements are processed by their internal order. Looping continues until the Foreach loop reaches the last element or upper bound of the given array.

An alternative form of Foreach loop gives you access to the current key:



In this case, the key for each element is placed in $key and the corresponding value is placed in $value.

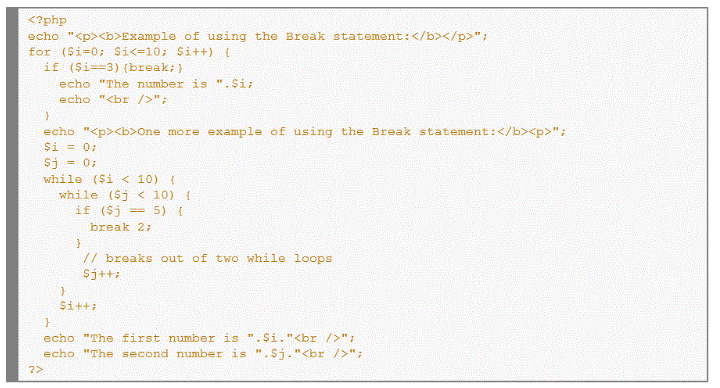
The Foreach construct does not operate on the array itself, but rather on a copy of it. During each loop, the value of the variable $value can be manipulated but the original value of the array remains the same.

Break and Continue Statements

Sometimes you may want to let the loops start without any condition, and allow the statements inside the brackets to decide when to exit the loop. There are two special statements that can be used inside loops: Break and Continue.

The Break statement terminates the current While or For loop and continues executing the code that follows after the loop (if any). Optionally, you can put a number after the Break keyword indicating how many levels of loop structures to break out of. In this way, a statement buried deep in nested loops can break out of the outermost loop.

Examples below show how to use the Break statement:



A function is a block of code which can be called from any point in a script after it has been declared. It is basically a compartmentalized PHP script designed to accomplish a single task. Furthermore, code contained within functions is ignored until the function is called from another part in the script. Functions are useful because they contribute to rapid, reliable, error-reducing coding, and increase legibility by tiding up complicated code sequences.

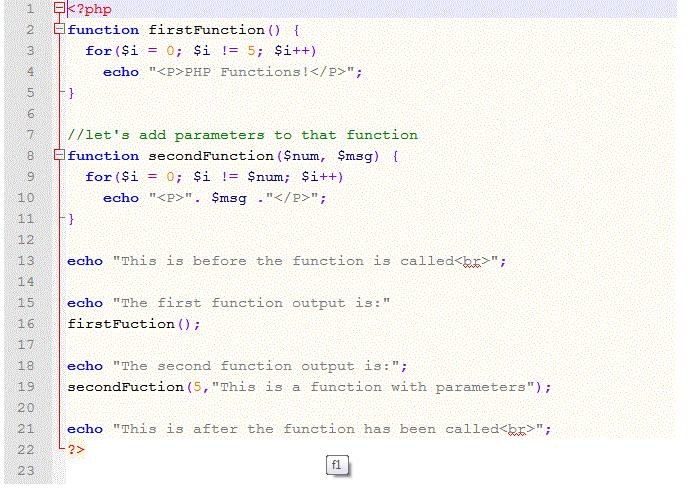
It is good programming practice to use functions to modularize your code and to better provide reuse. To declare a function, you simply type:

|  |
| --- |
| <?php  function function\_name(param\_1, ... , param\_n)  {  statement\_1;  statement\_2;  ... statement\_m;   return return\_value;  }  ?> |

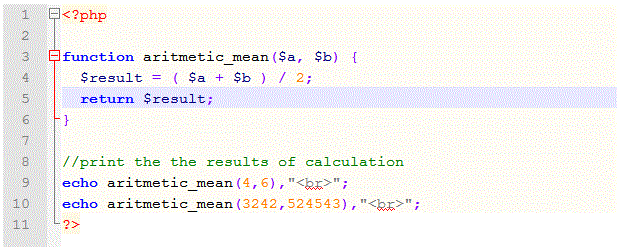
We can optionally pass parameters to the functions to be known as local variable, and we can also return a result with the "return value" statement. This produces the ending of the function returning a value.

Creating a simple function

Let's create two functions that will print the string "PHP Functions" five times, but the first one will not contain parameters, and the second one will. A function parameter is nothing more than a piece of data that the function requires to execute. In above example also included code to call the function. For the time being, ignore the "for" statement. We will talk about using these looping constructs in a later chapter.



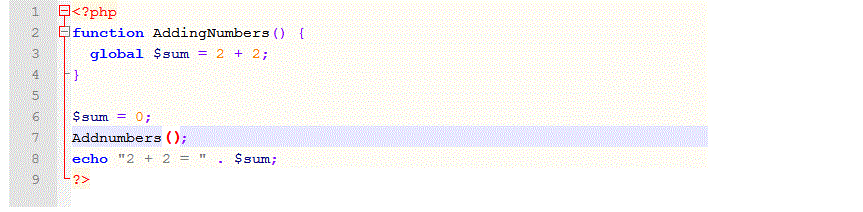
Next example creates a function that will calculate arithmetic mean and return a result with the "return value" statement:



Variable Scope and Lifetime

It's important to note that if you define a variable within a function, that variable is only available within that function; it cannot be referenced in another function or in the main body of your program code. This is known as a variable's scope. The scope of a variable defined within a function is local to that function.

If a function needs to use a variable that is defined in the main body of the program, it must reference it using the "global" keyword, like this:



While the scope of a variable defined in a function is local to that function, a variable defined in the main body of code has a global scope. The "global" keyword tells PHP to look for a variable that has been defined outside the function.

A variable defined within a PHP program script exists only while that script is running. When the script ends, the variable ceases to exist. And, a variable defined within a function exists only while that function is being processed; when the function ends, the variable ceases to exist.

Arrays can be used in many ways to store and organize data quickly and efficiently. It is one of the more useful data types available to any programming language.

Arrays can most easily be described as an ordered list of elements. You can access the individual elements by referring to their index position within the array. The position is either specified numerically or by name. An array with a numeric index is commonly called an indexed array while one that has named positions is called an associative array. In PHP, all arrays are associative, but you can still use a numeric index to access them.

An Example of an indexed Array:

|  |
| --- |
| <?php  $seven = 7;  $arrayname = array( "this is an element", 5, $seven );   echo $arrayname[0]; //prints: this is an element  echo $arrayname[1]; //prints: 5  echo $arrayname[2]; //prints: 7 ?> |

As you can see, elements in an array can be any type of data (string, integer, double) and can also be other variables. An individual array element can be of any type, including another array. If you want to find out if a variable contains an array you can use the is\_array() function. Notice that Indexed arrays start at position zero, not at position one, so your first array element has an index of 0, and the highest position in the array is one less than the number of elements in the array.

Associative Arrays

Associative arrays are arrays that use named keys that you assign to them. Have a look at the following example:

|  |
| --- |
| <?php  $first\_array = array("key1" => "the first element", "key2" => "the second element");   $second\_array = array(  "key3" => "this is the first element of the second array",  "key4" => "this is the second element of the second array",  );  echo $first\_array[key1]; //prints "the first element."  echo $second\_array[key3]; //prints "the first element of the second array"  echo $first\_array[key2]; //prints "the second element"  echo $second\_array[key4]; //prints "this is the second element of the second array"  ?> |

Right, now you know how to define an associative array, but you probably don't see yet how useful are they. Well think of this, say you have a flower-shop. You have 3 different flowers, and each flower has a different price. Let's make this example in php. Notice that you can add parenthesis to your echo statements when they are longer for readability:

<?php   
//We initialize the array using the array() function.   
//Note that for readability one can spread the argument over several lines.  
$flower\_shop = array (   
"rose" => "5.00",   
"daisy" => "4.00",   
"orchid" => "2.00"  
);   
  
echo ("rose costs $flower\_shop[rose], daisy costs $flower\_shop[daisy], and orchid costs $flower\_shop[orchid].");   
?>

Because the indices in this associative array are not numbers, we cannot use a simple counter in a for loop to work with the array. We can use the foreach loop. In the following example we use the foreach loop to iterate through our flowers\_shop array, and read them into a table. Note carefully the syntax.

<?php  
//We initialize the array using the array() function.   
//Note that for readability one can spread the argument over several lines.  
$flower\_shop = array (   
"rose" => "5.00",   
"daisy" => "4.00",   
"orchid" => "2.00",   
);   
//let's print out the headers to our table  
echo "<table border='1' cellpadding='5'>";  
echo("<tr><th>Flower</th><th>Price</th></tr>");  
//Now we start the foreach loop using the variable $flower to hold our key   
//and $price to hold our cost.  
foreach($flower\_shop as $Flower=>$Price)  
{  
echo ("<tr><td>$Flower </td><td>$Price</td></tr> ");

//print the values into a table cell for each iteration  
}  
//finally close the table  
echo ("</table>");  
?>

It is often necessary to arrange the elements in an array in numerical order from highest to lowest values (descending order) or vice versa (ascending order). If the array contains string values, alphabetical order may be needed. Sorting a one-dimensional arrays is quite easy.

Sorting Numerically Indexed Arrays

At first we will consider an array which contains string values. The code below sorts array elements in ascending alphabetical order:

|  |
| --- |
| <?php  $flowers = array("rose", "daisy" , "orchid", "tulip", "camomile"); sort($flowers);  for ($i=0; $i <= 4; $i++) //prints the array elements echo $flowers[$i]."<br \>";  ?> |

Now array elements will go in the alphabetical order. Output will be the following:

camomile  
daisy  
orchid  
rose  
tulip

We can sort values by numerical order too. If we have an array containing the prices of flowers, we can sort it into ascending numeric order. Have a look at the example:

|  |
| --- |
| <?php  $prices = array(1.25, 0.75 , 1.15, 1.00, 0.50); sort($prices);  for ($i=0; $i <= 4; $i++) //prints the array elements echo $prices[$i]."<br \>";  ?> |

The prices will now be in the following order:

0.50   
0.75  
1.00  
1.15  
1.25

Note that the sort function is case sensitive, i.e. all capital letters come before all lowercase letters. So "A" is less than "Z", but "Z" is less than "a".

Sorting Associative Arrays

If we are using an associative array we cannot sort an array by using the sort() function. If we apply the sort() function on an associative array, it is sorted by the numeric value of the index. To sort an associative array we need to use the asort() function to keep keys and values together as they are sorted.

The following code creates an associative array containing the three flowers and their associated prices, and then sorts the array into ascending price order:

|  |
| --- |
| <?php  $shop = array ( "rose" => "5.00",  "daisy" => "4.00",  "orchid" => "2.00",  );   asort($shop);  foreach($shop as $key => $value) echo $key." costs ".$value." dollars<br />"; //print the array elements  ?> |

The above example will output:

orchid costs 2.00 dollars  
daisy costs 4.00 dollars  
rose costs 5.00 dollars

The asort() function orders the array according to the value of each element. In the array, the values are the prices and the keys are the names of the flowers.

If instead of sorting by price we want to sort by flower name, we use ksort() function to sort an associative array according to the key.

The following code will result in the keys of the array being ordered alphabetically:

|  |
| --- |
| <?php  $shop = array ( "rose" => "5.00",  "daisy" => "4.00",  "orchid" => "2.00",  );  ksort($shop);  foreach($shop as $key => $value) echo $key." costs ".$value." dollars<br />"; //print the array elements  ?> |

Output will be the following:

daisy costs 4.00 dollars  
orchid costs 2.00 dollars  
rose costs 5.00 dollars

Sorting Arrays in the Reverse Order

We discussed sort(), asort(), and ksort() functions. All these functions sort array in the ascending order. Each of them has corresponding function that sorts an array in the descending order. These reverse functions are called rsort(), arsort() and krsort() respectively.

Reverse sorting functions are used the same way as usual sorting functions. Rsort() function sorts one-dimensional numerically indexed array by the values in reverse order. Arsort() function sorts one-dimensional associative array by the values in reverse order. Krsort() function sorts one-dimensional associative array by the keys in reverse order.

One of the best features of PHP is possibility to respond to user queries or data submitted from HTML forms. You can process information gathered by an HTML form and use PHP code to make decisions based off this information to create dynamic web pages.

The HTML Form

Before you can process the information, you need to create an HTML form that will send information to your PHP script. There are two methods for sending data: POST and GET. These two types of sending information are defined in your HTML form element's method attribute. Also, you must specify the location of the PHP file that will process the information.

Below is an HTML form that will send the data using the POST method. Information sent from a form with the POST method is invisible to others and has no limits on the amount of information to send. Copy and paste this code and save it as *form.html*.

<html>   
<head>   
<title>Process the HTML form data with the POST method</title>   
</head>   
<body>   
<form name="myform" action="process.php" method="POST">  
<input type="hidden" name="check\_submit" value="1" />  
Name: <input type="text" name="Name" /><br />  
Password: <input type="password" name="Password" maxlength="10" /><br />  
Select something from the list: <select name="Seasons">  
<option value="Spring" selected="selected">Spring</option>  
<option value="Summer">Summer</option>  
<option value="Autumn">Autumn</option>  
<option value="Winter">Winter</option>  
</select><br /><br />  
Choose one:   
<input type="radio" name="Country" value="USA" /> USA  
<input type="radio" name="Country" value="Canada" /> Canada  
<input type="radio" name="Country" value="Other" /> Other  
<br />  
Choose the colors:   
<input type="checkbox" name="Colors[]" value="green" checked="checked" /> Green  
<input type="checkbox" name="Colors[]" value="yellow" /> Yellow  
<input type="checkbox" name="Colors[]" value="red" /> Red  
<input type="checkbox" name="Colors[]" value="gray" /> Gray  
<br /><br />  
Comments:<br />   
<textarea name="Comments" rows="10" cols="60">Enter your comments here</textarea><br />  
<input type="submit" />  
</form>  
</body>  
</head>  
</html>

The example HTML page above includes different form elements: input fields, select list, text area, radio buttons, checkboxes and a submit button. When a user fills in this form and clicks on the submit button, the form data is sent to the *process.php* file.

Notice that we have added square brackets [] to the name of the checkbox element. The reason for the square brackets is that it informs PHP that the value may be an array of information. Users can select multiple values, and PHP will place them all into an array of the value of the name attribute.

For example, if you pick Green and Yellow and submit the form, then $\_POST['Colors'] is itself an array. $\_POST['Colors'][0] is Green and $\_POST['Colors'][1] is Yellow. That's because the name attribute of the checkbox element is Colors[]. If the name was just Colors, then $\_POST['Colors'] would be a string, holding only one of the selected values.

Processing the Form Data ( PHP Code )

Next, we are going to create our PHP file that will process the data. When you submit your HTML form, PHP automatically populates two super global arrays, $\_GET and $\_POST, with all the values sent as GET or POST data, respectively. Therefore, a form input called 'Name' that was sent via POST, would be stored as $\_POST['Name'].

Copy and paste this code and save it as *process.php* in the same directory as *form.html*.

<?php  
//Check whether the form has been submitted  
if (array\_key\_exists('check\_submit', $\_POST)) {  
//Converts the new line characters (\n) in the text area into HTML line breaks (the <br /> tag)  
$\_POST['Comments'] = nl2br($\_POST['Comments']);   
//Check whether a $\_GET['Languages'] is set  
if ( isset($\_POST['Colors']) ) {   
$\_POST['Colors'] = implode(', ', $\_POST['Colors']); //Converts an array into a single string  
}  
  
//Let's now print out the received values in the browser  
echo "Your name: {$\_POST['Name']}<br />";  
echo "Your password: {$\_POST['Password']}<br />";  
echo "Your favourite season: {$\_POST['Seasons']}<br /><br />";  
echo "Your comments:<br />{$\_POST['Comments']}<br /><br />";  
echo "You are from: {$\_POST['Country']}<br />";  
echo "Colors you chose: {$\_POST['Colors']}<br />";  
}  
else {  
echo "You can't see this page without submitting the form.";  
}  
?>

The first line checks whether the form has been submitted, and that the php script has not been called directly. Next, we convert the new line characters in the text area into HTML line breaks. Then we check whether a $\_POST['Colors'] is set, and if so we use the *implode()* function to convert $\_POST['Colors'] array into a single string. Finally, we print out all received values in the browser. When printing an array to the screen, left and right brackets are used around the array:

echo "Your name: {$\_POST['Name']}<br />";

GET and POST

When defining the method to send information to the PHP script, you either use GET or POST. Both send variables across to a script, but they do so in different ways.

The GET method sends its variables in the web browser's URL, which makes it easy to see and possibly change the information that was sent. So this method should not be used when sending passwords or other sensitive information. It also should not be used for any actions that cause a change in the server, such as placing an order or updating a database. However, because the variables are displayed in the URL, it is possible to bookmark the page.

The GET method has a limit on the amount of information than can be sent. As a result, if you send long variables using GET, you are likely to lose large amounts of them.

The POST method sends its variables behind the scenes and has no limits on the amount of information to be sent.

Extremely complex Web applications are created using this basic structure. HTML forms gather data from users, the data is processed by a back-end script, and a response of some type is sent back to the user. These back-end scripts also write data to databases, and can do much more complex activities, but the basic data gathering and processing is similar. Once you master gathering information from an HTML form, and writing a PHP script that can retrieve and work on that data, you are on your way to being able to create more complex programs.

An important part of any script is the ability to store information for later use. This data may be user data, or may be needed for the script itself. Relational databases are normally used for data storage, but there are many times that a programmer needs to write information into files on the server, instead of entering the data into a database system. Programmers can actually create very extensive "flat-file" databases using only text files to store information. The first CGI Shopping Cart system that I wrote in the Perl language, for example, used a flat-file database system that contained dozens of individual files that stored information for the script.   
  
At the end of this module, you should be able to:

* Open a file on the server
* Write information to a file
* Open and read information from a file

Opening a File:  
  
The fopen() function is used to open a file on the server. When opening a file, attributes are used to tell the server how you want to open and work with the file. You can open a file for reading only, for writing only, or for reading and writing. When writing to a file, you can also specify if you want to overwrite the data that exists in the file, or append data to the end of the file. You can also specify how data is written into the file, whether it is written as binary data or text.

|  |
| --- |
| $fp = fopen("./public\_html/~jcannon/php/myfile.txt", "w"); |

The code above opens the file "myfile.txt" using the writing option. The $fp variable is called the "file pointer". This variable is used later to reference the file that was opened.  
  
Here are some additional attributes that can be used instead of "w":

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | r | Read mode | | r+ | Read and Write | | w | Write only | | w+ | Read and write - delete existing file first | | a | Write - append data to the end of file | | a+ | Read and write, append the data | | b | Binary mode | |

To write to a file, use the fwrite() function. For example:

|  |
| --- |
| $fwrite($fp, $output); |

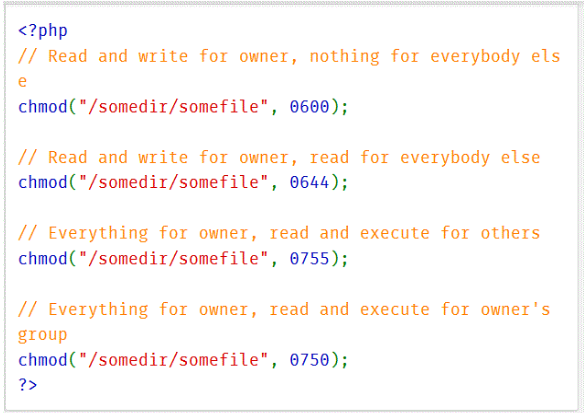
As you can see, you first open the file for writing, and then use the file pointer variable to write to the file.  
  
When you are finished with a file, the file needs to be closed. This is done using the fclose() function. 

|  |
| --- |
| fclose($fp); |

# Unix File Permissions:

On Unix servers files have a set of permissions that determine what users can do with the file. This is called the mode of the file, and in Unix we use the chmod() function to set and change the mode of a file.

The following diagram shows the PHP code for setting different file permissions on a Unix server.



There is a nice calculator at <http://www.onlineconversion.com/html_chmod_calculator.htm> that will help you convert to the correct chmod() values for your PHP scripts.

Important! The chmod command does not work on the Webtrain server. Set your file permissions using your FTP software when working on Webtrain.

Create a File  
$my\_file = 'file.txt';  
$handle = fopen($my\_file, 'w') or die('Cannot open file: '.$my\_file);  
  
Open a File  
$my\_file = 'file.txt';   
$handle = fopen($my\_file, 'w') or die('Cannot open file: '.$my\_file);   
//open file for writing ('w','r','a')...  
  
Read a File  
$my\_file = 'file.txt';   
$handle = fopen($my\_file, 'r');   
$data = fread($handle,filesize($my\_file));  
  
Write to a File  
$my\_file = 'file.txt';  
$handle = fopen($my\_file, 'w') or die('Cannot open file: '.$my\_file);   
$data = 'This is the data';   
fwrite($handle, $data);  
  
Append to a File  
$my\_file = 'file.txt';  
$handle = fopen($my\_file, 'a') or die('Cannot open  
file: '.$my\_file);   
$data = 'New data line 1';   
fwrite($handle, $data);   
$new\_data = "\n".'New data line 2';  
fwrite($handle, $new\_data);  
  
Close a File  
$my\_file = 'file.txt';  
$handle = fopen($my\_file, 'w') or die('Cannot open  
file: '.$my\_file);   
//write some data here   
fclose($handle);  
  
Delete a File  
$my\_file = 'file.txt';  
unlink($my\_file);

The following code reads from a file named "myfile.txt" using the fread() function. The myfile.txt file has tabbed delimited data in the file.

fread() reads the data from a file until a specified number of bytes has been read. The filesize() function is retrieving the number of bytes in the file, so the entire file is read in this case. The explode() function is used to find the tabs in the data file that separate the data items. This is stored in the $splitcontents array. Then we loop through the array, and print the data items to the screen.

<?php

$filename = "myfile.txt";

$fd = fopen($filename,"r");

$contents = fread($fd,filesize($filename));

fclose($fd);

$splitcontents = explode(" ", $contents);

$counter = 0;

foreach($splitcontents as $data){

echo $counter++;

echo ": " . $data;

}

?>

It is a very good idea to introduce some error checking into your file access routines. To do this, we can modify the code as follows:

<?php

$filename = "myfile.txt";

if (!$fd = fopen($filename,"r")) {

echo("Cannot open file" $filename);

exit;

}

if (!$contents = fread($fd,filesize($filename))) {

echo("Cannot read file" $filename);

exit;

}

fclose($fd);

$splitcontents = explode(" ", $contents);

$counter = 0;

foreach($splitcontents as $data){

echo $counter++;

echo ": " . $data;

}

?>

Exercise:

This is a good time to introduce the official PHP website. Go over to [PHP.net](http://www.php.net/) and use the search box at the top right to search for PHP functions. Search for the term, "filesystem", and choose, "function list", from the drop down menu. You will then see a list of all PHP functions related to working with a server's file system. Spend some time on the site.

As is necessary for any language, PHP has a complete set of directory support functions. PHP gives you a variety of functions to read and manipulate directories and directory entries. Like other file-related parts of PHP, the functions are similar to the C functions that accomplish the same tasks, with some simplifications. This tutorial describes how PHP handles directories. You will look at how to create, remove, and read them.

Reading the Contents of a Directory

Let's start with simple listing the contents of a directory. We need three functions to perform this task: opendir(), readdir() and closedir(). The opendir() function takes one parameter, which is the directory we want to read, and returns a directory handle to be used in subsequent readdir() and closedir() calls. opendir() returns False if the directory could not be opened.

The readdir() function takes one parameter, which is the handle that opendir() returned and each time we call readdir() it returns the filename of the next file in the directory. readdir() returns False if the end of the directory has been reached. Note that readdir() returns only the names of its items, rather than full paths.

The example below creates a select box that lists all the files in a directory. Copy and paste this code and save it as *index.php* in a directory you wish do display all files for. It automatically excludes itself from the list, and is easy to modify to make it ignore other files as well:

<?php  
// open the current directory  
$dhandle = opendir('.');  
// define an array to hold the files  
$files = array();  
  
if ($dhandle) {  
// loop through all of the files  
while (false !== ($fname = readdir($dhandle))) {  
// if the file is not this file, and does not start with a '.' or '..',  
// then store it for later display  
if (($fname != '.') && ($fname != '..') &&  
($fname != basename($\_SERVER['PHP\_SELF']))) {  
// store the filename  
$files[] = (is\_dir( "./$fname" )) ? "(Dir) {$fname}" : $fname;  
}  
}  
// close the directory  
closedir($dhandle);  
}  
  
echo "<select name=\"file\">\n";  
// Now loop through the files, echoing out a new select option for each one  
foreach( $files as $fname )  
{  
echo "<option>{$fname}</option>\n";  
}  
echo "</select>\n";  
?>

First, we open the directory for reading with the opendir() function and use a while statement to loop through each of its entries. We call readdir() as part of the while statement's test expression, assigning its result to the $fname variable. We are explicitly testing whether the return value is equal to and of the same type as False since otherwise, any directory entry whose name evaluates to False will stop the loop prematurely. Within the body of the while statement, we check if the file is not this file, and does not start with a **.** (current directory) or **..** (parent directory) and then store the file name into the $files array. We also do one more check. If a full file path leads to a directory then we add to the file name "(Dir)".

There is another way to iterate over all files in a directory. PHP 5 has a set of objects called iterators. Iterators help eliminate problems in your code. For instance, PHP 5 provides a DirectoryIterator:

<?php  
echo "<select name=\"file\">\n";  
foreach (new DirectoryIterator('.') as $file) {  
// if the file is not this file, and does not start with a '.' or '..',  
// then store it for later display  
if ( (!$file->isDot()) && ($file->getFilename() != basename($\_SERVER['PHP\_SELF'])) ) {  
echo "<option>";  
// if the element is a directory add to the file name "(Dir)"  
echo ($file->isDir()) ? "(Dir) ".$file->getFilename() : $file->getFilename();  
echo "</option>\n";  
}  
}  
echo "</select>\n";  
?>

This example produces the same results as the earlier code that uses the directory functions, but this code is shorter and safer because you cannot forget the !== = comparison.

Deleting the Directory and Its Contents

PHP has the rmdir[[egg]] function that takes a directory name as its only parameter and will remove the specified directory from the file system, if the process running your script has the right to do so. However, the rmdir() function works only on empty directories. The example below deletes empty directory named "temporary":

<?php  
rmdir( "temporary" );  
?>

If you want to delete a non-empty directory, you would use recursion. A recursive function is a function that calls itself. Sounds strange, but the technique is used quite a bit in Computer Science. In the following example we create a recursive function named deleteDir() that takes a directory name as a parameter and will go through each subdirectory, deleting files as it goes. When the directory is empty, we use rmdir() to remove it.

<?php  
function deleteDir($dir) {  
// open the directory  
$dhandle = opendir($dir);  
  
if ($dhandle) {  
// loop through it  
while (false !== ($fname = readdir($dhandle))) {  
// if the element is a directory, and   
// does not start with a '.' or '..'  
// we call deleteDir function recursively   
// passing this element as a parameter  
if (is\_dir( "{$dir}/{$fname}" )) {  
if (($fname != '.') && ($fname != '..')) {  
echo "<u>Deleting Files in the Directory</u>: {$dir}/{$fname} <br />";  
deleteDir("$dir/$fname");  
}  
// the element is a file, so we delete it  
} else {  
echo "Deleting File: {$dir}/{$fname} <br />";  
unlink("{$dir}/{$fname}");  
}  
}  
closedir($dhandle);  
}  
// now the directory is empty, so we can use  
// the rmdir() function to delete it  
echo "<u>Deleting Directory</u>: {$dir} <br />";  
rmdir($dir);  
}  
  
// call deleteDir function and pass to it   
// as a parameter a directory name  
deleteDir("temporary");  
?>

Another way to delete non-empty directory is to use RecursiveDirectoryIterator and RecursiveIteratorIterator. The RecursiveIteratorIterator must be told to provide children (files and subdirectories) before parents with its CHILD\_FIRST constant. Have a look at the code:

<?php  
function deleteDir($dir) {  
$iterator = new RecursiveDirectoryIterator($dir);  
foreach (new RecursiveIteratorIterator($iterator, RecursiveIteratorIterator::CHILD\_FIRST) as $file)   
{  
if ($file->isDir()) {  
rmdir($file->getPathname());  
} else {  
unlink($file->getPathname());  
}  
}  
rmdir($dir);  
}  
  
deleteDir("temporary");  
?>

Note: The CHILD\_FIRST constant was added with PHP 5.1

Creating New Directories

Creating new directories in PHP is accomplished using the mkdir() function, which takes two parameters. These parameters are, in order, a directory name you want to create and a permission mode for a new directory, which must be an octal number. The mode parameter is optional and has an effect only on Unix systems. Have a look at the example:

<?php  
// if /path/to/my exists, this should return true   
// if PHP has the right permissions  
mkdir("/path/to/directory", 0777);  
?>

By default, the mkdir() function only creates a directory if its parent exists. In PHP 5 the recursive parameter was added which should be true or false, depending on whether you want to create parent directories or not. In the following example, we pass true as a third parameter to mkdir(), so this makes the function act recursively to create any missing parent directories.

<?php  
// will create /path/to/directory and  
// also create /path and /path/to if needed and allowed  
mkdir("/path/to/directory", 0777, true);  
?>  
  
Please note that not all Web servers have been upgraded to PHP 5. Many are still running PHP 4.x, so the PHP 5 code displayed here may not work on all servers.

So far we've learned how to add PHP to HTML documents, to provide users with more interactive experiences. This is just the beginning of what can be done with PHP. To write more advanced, complex, multi-user programs, we need to store the data that the program needs in a database.   
  
Combining a programming language with a back-end database opens up all kinds of possibilities. Imagine some of the Web-based applications that you personally use. They all store information for you, so that you can return to the application later, and continue using the application by simply "logging into" the application. All of the complexity of storing your user name and password, and other more advance options, such as which dashboard of the application you have set to display first, is all data that the program has to store for every user. At some point, the amount of data being stored exceeds the limits of a simple flat-file database structure, and something much more robust and scalable is needed.   
  
The following sections introduce the MySQL database system, and the benefits of moving to Relational Database Management Systems...  
  
From this point on in the course you will not be able to run the example PHP scripts on the Webtrain server. There is no MySQL database set up on the server for these scripts. So, please study the material, but do not upload and run any of the scripts on the server.

PHP and MySQL:

Search youtube video on “introduction to MySql for website administrators”

Search youtube video on “introduction to relational databases”

Search youtube video on “database concepts & design”

The first step to integrating a PHP script with your MySQL database is to design and create the database itself, which is covered in the next class in this series (MySQL). We'll concentrate here on the PHP code that is needed to work with the database once it is created. The first section of code below shows how to connect to a previously created MySQL database that was created under the user name "pee\_wee". The database was created using the password, "let\_me\_in", and the host name can be "localhost" because we are running this PHP script on the same host that the database resides on. The code uses the mysql\_connect function to connect to the server, passes the authentication data to the function, and uses the die function for error handling:  
  
<?php  
$username = "pee\_wee";  
$password = "let\_me\_in";  
$hostname = "localhost";   
$dbh = mysql\_connect($hostname, $username, $password) or die("Unable to connect to MySQL");  
print "Connected to MySQL<br>";  
// you're going to do lots more here soon  
mysql\_close($dbh);  
?>  
  
Once we are connected to the server we can connect to a specific MySQL database and start working with the db. The next code snippet connects to the server and then selects a particular MySQL database named, "first\_test".   
  
<?php  
$username = "pee\_wee";  
$password = "let\_me\_in";  
$hostname = "localhost";   
$dbh = mysql\_connect($hostname, $username, $password) or die("Unable to connect to MySQL");  
print "Connected to MySQL<br>";  
$selected = mysql\_select\_db("first\_test",$dbh) or die("Could not select first\_test");  
// you're going to do lots more here soon  
mysql\_close($dbh);  
?>  
  
The next snippet uses the mysql\_query built-in function to insert a value into a particular "table" in the database. The name of the table is, "people". You can see from the code that there are three items of data being written into the "people" table:  
  
<?php  
$username = "pee\_wee";  
$password = "let\_me\_in";  
$hostname = "localhost";   
$dbh = mysql\_connect($hostname, $username, $password) or die("Unable to connect to mysql");  
print "connected to mysql<br>";  
$selected = mysql\_select\_db("first\_test",$dbh) or die("Could not select first\_test");  
if (mysql\_query("insert into people values('5','Hazel','Burger')")) {  
print "successfully inserted record";  
}  
else {  
print "Failed to insert record";  
}  
mysql\_close($dbh);  
?>

In the next snippet we are accessing an array of data that is stored in the db in the "people" table. MySQL tables can hold all sorts of formats of data, from text items to whole arrays of data. First the code uses the "SELECT" SQL query to choose the names of the items it wants to retrieve, specifically, id, first\_name and surname. Then it has to separate the id, the first name and the last name from the array that was passed back:  
  
<?php  
$username = "pee\_wee";  
$password = "let\_me\_in";  
$hostname = "localhost";   
$dbh = mysql\_connect($hostname, $username, $password) or die("Unable to connect to MySQL");  
$selected = mysql\_select\_db("first\_test",$dbh) or die("Could not select first\_test");  
$result = mysql\_query("SELECT id, first\_name,surname FROM people");  
while ($row = mysql\_fetch\_array($result,MYSQL\_ASSOC)) {  
print "ID:".$row{'id'}." Name:".$row{'first\_name'}." ".$row{'surname'}."<br>";  
}  
mysql\_close($dbh);  
?>  
  
The next snippet shows how to update the "people" table, by using the SQL SET command. We are setting the last name to "Arendse" where the id is equal to 6. The else statement traps any errors that happen trying to perform the update:  
  
<?php  
$username = "pee\_wee";  
$password = "let\_me\_in";  
$hostname = "localhost";   
$dbh = mysql\_connect($hostname, $username, $password) or die("Unable to connect to MySQL");  
$selected = mysql\_select\_db("first\_test",$dbh) or die("Could not select first\_test");  
if (mysql\_query("UPDATE people SET surname='Arendse' WHERE id=6")) {  
print "Successfully updated data";  
}  
else {  
print "Failed to update data";  
}  
mysql\_close($dbh);  
?>  
  
The last snippet adds some code to ask the database to print the content of the entire database, using the "SHOW DATABASES" SQL statement.  
  
  
<?php  
$username = "pee\_wee";  
$password = "let\_me\_in";  
$hostname = "localhost";   
$dbh = mysql\_connect($hostname, $username, $password) or die("Unable to connect to MySQL");  
$selected = mysql\_select\_db("first\_test",$dbh) or die("Could not select first\_test");  
$result = mysql\_query("SHOW DATABASES");  
while ($row = mysql\_fetch\_array($result,MYSQL\_NUM)) {  
print "Database:".$row[0]."<br>";  
}  
mysql\_close($dbh);  
?>  
  
In order to write database applications in PHP, you will need to learn how to design and create your database, as well as become familiar with the various Structured Query Language (SQL) commands that are used to retrieve data from a database.

*/\**

*\*\* Connect to database:*

*\*/*

*// connect to the database*

$con = [mysql\_connect](http://www.php.net/mysql_connect)('localhost','testuser','testpassword')

or [die](http://www.php.net/die)('Could not connect to the server!');

*// select a database:*

[mysql\_select\_db](http://www.php.net/mysql_select_db)('testdb')

or [die](http://www.php.net/die)('Could not select a database.');

*/\**

*\*\* Fetch some rows from database:*

*\*/*

*// read username from URL*

$username = $\_GET['username'];

*// escape bad chars:*

$username = [mysql\_real\_escape\_string](http://www.php.net/mysql_real_escape_string)($username);

*// build query:*

$sql = "SELECT id, timestamp, text FROM logs WHERE username = '$username'";

*// execute query:*

$result = [mysql\_query](http://www.php.net/mysql_query)($sql)

or [die](http://www.php.net/die)('A error occured: ' . [mysql\_error](http://www.php.net/mysql_error)());

*// get result count:*

$count = [mysql\_num\_rows](http://www.php.net/mysql_num_rows)($result);

[print](http://www.php.net/print) "Showing $count rows:<hr/>";

*// fetch results:*

while ($row = [mysql\_fetch\_assoc](http://www.php.net/mysql_fetch_assoc)($result)) {

$row\_id = $row['id'];

$row\_text = $row['text'];

[print](http://www.php.net/print) "#$row\_id: $row\_text<br/>**\n**";

}

*/\**

*\*\* Do a insert query:*

*\*/*

*// create SQL query:*

$sql = "INSERT INTO logs (timestamp, text) VALUES (NOW(), 'some text here!')";

*// execute query:*

$result = [mysql\_query](http://www.php.net/mysql_query)($sql) or [die](http://www.php.net/die)('A error occured: ' . [mysql\_error](http://www.php.net/mysql_error)());

*// get the new ID of the last insert command*

$new\_id = [mysql\_insert\_id](http://www.php.net/mysql_insert_id)();

*/\**

*\*\* Do a update query:*

*\*/*

*// create SQL query:*

$sql = "UPDATE logs SET text='New text!' WHERE id='1'";

*// execute query:*

$result = [mysql\_query](http://www.php.net/mysql_query)($sql) or [die](http://www.php.net/die)('A error occured: ' . [mysql\_error](http://www.php.net/mysql_error)());

*/\**

*\*\* Do a delete query:*

*\*/*

*// create SQL query:*

$sql = "DELETE FROM logs WHERE id='1'";

*// execute query:*

$result = [mysql\_query](http://www.php.net/mysql_query)($sql) or [die](http://www.php.net/die)('A error occured: ' . [mysql\_error](http://www.php.net/mysql_error)());

Server side scripting languages, such as PHP, require a Web Server to operate. This section will help you install and configure a Web Server, along with the languages and tools you need to develop and run PHP/MYSQL scripts on your home computer.  
  
The XAMPP bundle, found at <https://www.apachefriends.org/index.html> was chosen because it includes distributions for both Mac and Windows machines. Windows users can alternately use the EasyPHP distribution, which is detailed in the Shop Script Free User Guide that is included with the shopping cart system download.  
  
XAMPP and EasyPHP both include the Apache Web Server, PHP, the PHPMyAdmin application and a MySQL database. Once installed, you will have your own Web Server on your computer, and will be able to run PHP/MySQL applications without needing a separate Web Hosting account.

Final Project

Write a program that creates a very simple text editing program in PHP. The script will display a form to the user, with the content of a file inside a textarea box. (You can set the specific name of the file to open in your script. The script does not have to allow the user to enter a file name).  
  
When the user makes any changes to the displayed file in the textarea box, and clicks on your form's submit button, your script saves the new version to the same file name.   
  
I have added some code for you view in the next section. See if you can write the script without copying my code, but feel free to use my code if needed.   
  
You will need access to your Webtrain Unix account in order to test your code before submitting this assignment. Name your file using the .php file extension, upload the file to your Webtrain account, and test the program in a Web browser.   
  
You will be graded on the presentation of your script, as well as the functionality, so please be sure the script is visually pleasing as well as functional.  
  
After you are sure the program functions correctly, upload a .zip, below, that contains the URL of your .php file on the Webtrain server and a copy of your PHP script.  
  
Remember! Every project you do can become part of your portfolio - so add HTML, CSS, graphics, etc. to make this project something you will be proud to add to your portfolio.

Final Project Tips and Hints.

// Script that displays a form and also gets the posted data from the form if   
// the user submitted the form. This method can be used in your script to   
// combine the task of displaying the contents of the existing file to the   
// user in a textarea window, as well as allowing them to change the file  
// and submit the new file content.  
  
<?  
//If we submitted the form  
if(isset($\_POST['submitMe']))  
{  
echo("Hello, " . $\_POST['name'] . ", we submitted your form!");  
}  
//If we haven't submitted the form  
else  
{  
?>  
<form action="<?=$\_SERVER['PHP\_SELF']?>" method="POST">  
<input type="text" name="name"><br>  
<input type="submit" value="submit" name="submitMe">  
</form>  
<?  
}  
?>  
  
// You may need to use the strip\_slashes() function   
// The result of the code below is that the quotes are stripped  
// off of the string $str

<?php  
$str = "Is your name O'reilly?";  
  
// Outputs: Is your name O'reilly?  
echo stripslashes($str);  
?>

Other functions you will need to use:  
  
fopen()  
fwrite()  
fclose()  
fread()