PHP Tutorials:

Programming with PHP and MySQL

A set of tutorials derived from a series of lectures

Paul Gibbs – Ilfracombe, North Devon, England

SAMPLE

PHP Tutorials: Programming with PHP and MySQL

written by Paul Gibbs

Copyright

Copyright © 2014 Paul Gibbs

All rights reserved. This book or any portion thereof may not be reproduced or used in any manner whatsoever without the express written permission of the publisher except for the use of brief quotations in a book review.

ISBN Numbers:
978-0-9928697-3-1 (This book)

978-0-9928697-0-0 (eBook – PDF)
978-0-9928697-1-7 (eBook - ePub)
978-0-9928697-2-4 (eBook – Kindle)

SAMPLE

Websites and Source Code

http://www.paulvgibbs.com
http://www.withinweb.com

Download all of the source code from:
http://www.paulvgibbs.com
or
http://www.withinweb.com/ebooks/

Email: paul@paulvgibbs.com

Trademarks

MySQL is a trademark of Oracle Corporation and/or its affiliates. Macintosh and Mac OS X are registered trademarks of Apple Inc. Microsoft and Windows are registered trademarks of Microsoft Corp. The WordPress Foundation owns and oversees the trademarks for the WordPress and WordCamp names and logos. Other product names used in this book may be trademarks of their own respective companies. This book is not affiliated with or endorsed by any of the products mentioned.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an 'as is' basis. The author and publisher shall have no liability or responsibility to any person or entity regarding any loss or damage incurred, or alleged to have incurred, directly or indirectly, by the information contained in this book. You hereby agree to be bound by this disclaimer.

SAMPLE

Preface

Written from a series of college lectures on PHP and MySQL, this book is a practical look at programming. It starts with an introduction to PHP and then goes on to MySQL and how to use SQL with the PHP language.

It provides an introduction to web programming, how to display data from a database and update data to the database. It explains issues that you encounter in real world situations and provides the basic code from which you can then use to further develop.

It would be helpful if you have some HTML knowledge, but the examples should provide you with what you need to know. It would also be helpful to have had some experience in programming using one of the more popular languages such as Visual Basic, although this is not necessary.

Many of the tutorials consists of a series of examples and tasks which illustrate each point and concentrate on simplified code so that you can see how they are used.

* Introduction – Basic PHP concepts.
* Variables - Variables, programming techniques and so on.
* Forms and PHP - Posting data between forms.
* Arrays – loops and array structures.
* Basic PHP structures - include files and how to use them.
* Functions – writing and using your own functions.
* Posting Forms – how forms work.
* Email Forms – an email form.
* JavaScript – posting forms with JavaScript.
* SQL and MySQL - Querying database tables using SQL.
* An Example table – some example data.
* Database Access – how to connect to a database.
* Using PHP and MySQL – an example form.
* Further PHP and MySQL - More PHP and MySQL.
* OpenCart – an example of an Open Source project.
* Error handling and debugging - Simple methods to find errors.
* Cookies and sessions - When and where to use them.
* Modifying Records - editing and updating databases with PHP and SQL.
* Classes - An introduction to object orientated programming.
* File Handling - Reading and writing to text files.
* Regular Expressions and Validation - Some validation methods.
* PHP security - Some methods to overcome this issue.

SAMPLE

* JQuery - An introduction to jQuery.
* htaccess and php.ini - how to use them.
* WordPress - an introduction to creating a WordPress Plugin.

Using the code examples

All the code examples with answers to the exercises may be downloaded to your computer the web site:

http://www.paulvgibbs.com

or directly from this link:

http://www.withinweb.com/phptutorials/phptutorials.zip

How to copy and paste text from Kindle books

You may want to copy and paste programming code examples from Kindle onto your computer so that you don't have to re-type them.

To do this in Kindle, highlight the text you want to copy and click on the "Highlight" button. Now go on the web to your Kindle account (https://kindle.amazon.com) and click on "Your Highlights" link. You'll see all the text you've highlighted in your Kindle books. From there, you can copy the text and paste it into another program.

SAMPLE

Basic Programming Terms

Variables - Areas of memory set aside by the programmer to store data and assigned a name by the programmer. These variables can then be used for such things as temporary storage during calculations or performing tests for decisions.

Data types - Defines the type of data that a variable can hold. This might be a string of characters such as for the name of 'Paul', or may be a numeric value such as someone's age, e.g. 56, or decimal value for the price of a product, e.g. 23.12

Arrays - These are variables, but instead of holding one value, can hold multiple values with each value identified by a reference name. So, you may have an array of towns such as Trowbridge, Chippenham, Lackham referenced by 0, 1 and 2.

Decision-making with if statements - Decision-making or conditional statements allow you to control the flow of a program. We can test a value after doing a calculation, or when someone enters a value into a form. The result of the test will be either true or false and we can then make the script do something depending on the result.

Programming loops - these are for repeating code a set number of times or until a condition is met. They are often used when we have rows of data in a table, and we loop around the rows performing some action until there are no more rows left.

POST and GET - This is how you send data to another web page where you can then process the data. POST and GET work with forms where the user enters information.

Cookies - A cookie is small text file that sits on the user's computer. It is often used to store data so that the user has a better browsing experience should they return to the web site. For example, a shopping cart system may store the items that the user has selected. When they return to the store, the contents of the cart can be displayed and the user can continue shopping.

Sessions - Another way for the programmer to retain information; however, session data is deleted and lost when the user closes the browser window.

OOP or Object Orientated Programming - A class in OOP programming terms is a definition which represents something using variables (called properties in OOP) and functions (called methods in OOP). Objects are then built from the class definition and it is possible for many objects to be based on the same class.

SAMPLE

Book Version

The current version is May 2014:

* A new section on WordPress Plugin and an Appendix has been added to cover phpMyAdmin.

January 2014 update:

* A new section on PHP security now replaces the original section on SQL Injection.
* The section on file handling has been improved.
* General reformatting changes.

November 2013 update:

* Reformatting of code examples to improve layout
* Change to some code examples to cover mysqli rather than mysql functions
* Minor text changes to text.

SAMPLE

Table of Contents

SAMPLE

SAMPLE

SAMPLE

SAMPLE

SAMPLE

SAMPLE

SAMPLE

SAMPLE

SAMPLE

1 - Introduction to PHP

What's in this chapter:

* Requirements for PHP.
* Simple PHP script and a PHP page.
* Language syntax.

PHP (*Hypertext Preprocessor*) is an open source scripting language that is especially suited for web development.

PHP code is embedded within HTML pages and enclosed within the special start and end tags
<?php and **?>**

The code is executed on the web server and NOT on your local client computer; the server generates the resulting HTML and sends it on to the client where you see it in your web browser.

PHP is quite easy for newcomers to learn but has many advanced features particularly when working with databases. Many of the programming modules that you require are installed as standard, and with a wide range of inbuilt functions and its ability to integrate with HTML it has made it a preferred language for web applications.

SAMPLE

What you need

If you want to work with PHP at home, you can either use a remote web server, in which case you will need to purchase web space and a domain name from a hosting company, or you can create a system on your local computer.

The easiest solution for most people is to purchase web space from a hosting company so that you don't have to worry about installing or updating PHP and other programs on your local computer. Hosting companies are very competitive in pricing, they provide large disc space for your files and provide all the administration tools to create and work with MySQL databases.

We can break down the required elements that you need as follows:

A web server

The web server is the computer system that sends web pages to your browser. Apache is often used as the web server but Microsoft servers are also capable of handling **PHP**.

PHP and MySQL are usually associated with LAMP (**L**inux, **A**pache, **M**ySQL, **P**HP). However, many **PHP** developers use Windows when developing the PHP application which is called WAMP (**W**indows, **A**pache, **M**ySQL, **P**HP).

There are several PHP / MySQL / Apache packaged systems that you can download and install on Windows computers. One of these is EasyPHP http://www.easyphp.org/ which can be used for development.

PHP

If you are using a hosting company then you don't have to do anything other than check that you are working with the latest version. Some hosts enable you to change the **PHP** version number for defined folders which can be useful if you have a script which uses out dated functions.

MySQL

Ensure that you have a way to administer you MySQL databases, this is usually done with a web based application called **phpMyAdmin**.

Web browsers

It is useful to have a selection of web browsers on your computer as then you can test your displays. The browsers you choose depends on personal choose although some have better web development features than other.

SAMPLE

A Text Editor

You obviously need an editor to write your **PHP** scripts. As **PHP** and HTML are just text files, you can use any text editor such as NotePad or NotePad++ However, there are a number of commercial products such as Dreamweaver. Kompozer is a free HTML editor http://www.kompozer.net/ which also has the ability to create **PHP** documents.

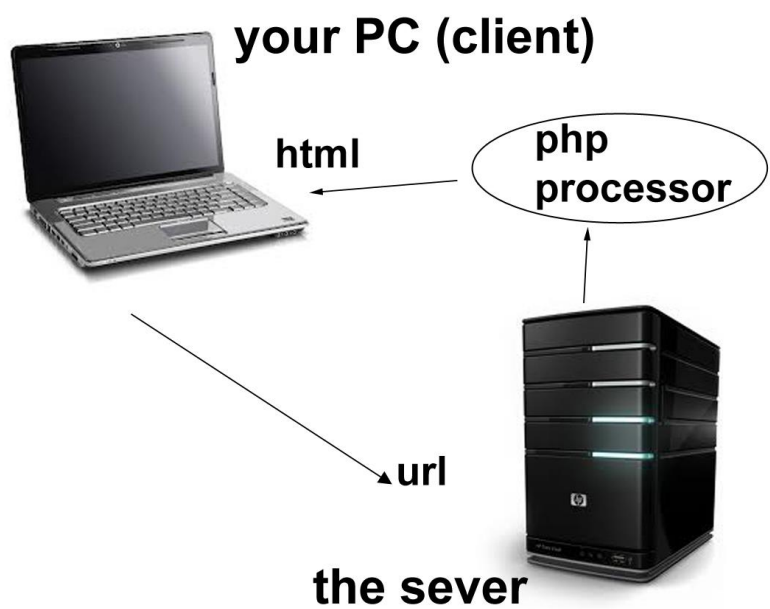
FTP Program

An FTP program (such as FileZilla) is needed if you are using a remote Server although some editors such as Dreamweaver have FTP built in.

The processing of PHP

When you use a browser to display a web page on your computer, the remote web server reads the **PHP** and processes it according to the code. The **PHP** processor then sends the generated HTML to your web browser.

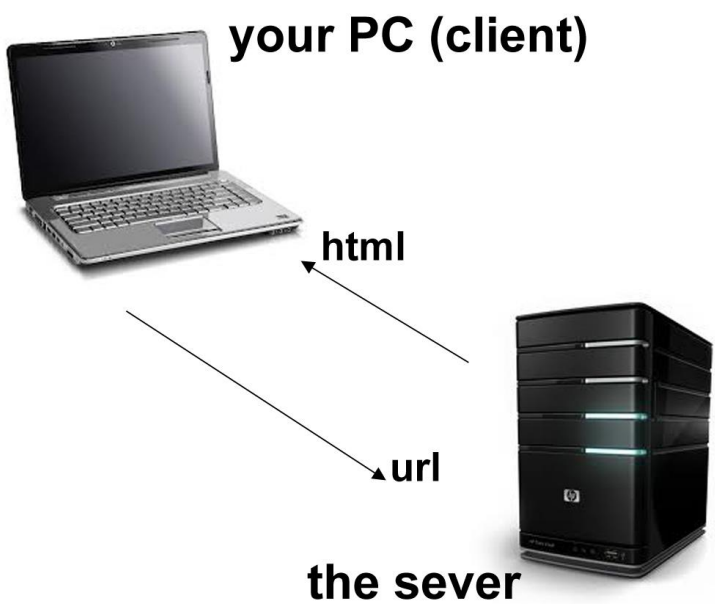
So **PHP** creates an HTML page on the fly based on the coding that you have created in the **PHP** page.



The Client is your local computer, while the Server is the remote computer which will be a web server.

The figure above demonstrates how the process works between a **Client**, the **PHP** processing module and the **Server** to send HTML back to the browser. All server-side technologies (**PHP**, for example) use some sort of processing module on the server to process the data that gets sent back to the client.

If you have a site which consists only of HTML web pages, the server merely sends the HTML data to the web browser as shown in the figure below.



This is why HTML pages can be viewed in your web browser from your own computer since they do not need to be "served," but dynamically generated pages need to be accessed through a server which handles the processing.

To the end user, there will not be any obvious difference between a page delivered through **PHP** and one delivered as just HTML.

PHP allows the creation of dynamic web pages which can display different data on a web page depending on the programming of the **PHP** script. It is often used in conjunction with a database, enabling programs to be written for e-commerce shopping systems, content management systems and discussion forums.

A simple PHP script

We start with a simple web page which should be a standard XHTML document but can actually be just a blank page without any HTML code at all. The following is an example of an HTML page which you should be familiar with if you have done some web design. A **PHP** page is identified to the server by having a file extension of **.php**

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
</head>

<body>
</body>
</html>
```

PHP is an HTML embedded scripting language. This means you can combine **PHP** and HTML code within the same script and why it is helpful to have some knowledge of HTML when you do **PHP** programming.

To place **PHP** code within a document, you surround the code within **PHP** tags as shown below. These need to be placed inside the **<body> … </body>** tags of the HTML page.

```
<?php

?>
```

Anything placed within these tags will be treated by the web server as **PHP** code.

Error reporting

When an error occurs in a **PHP** script it will display that error to the browser. Most web servers will be set up in this way. However, you may find that your server is not displaying errors especially if you are running **PHP** on a Microsoft IIS server. To overcome this, you can use the following code at the top of each **PHP** script and just after the <?php tag.

```
ini_set('display_errors','On');
error_reporting( E_ALL | E_STRICT | E_DEPRECATED );  // Show all errors.
```

SAMPLE

TASK 1 - A PHP page

You may want to start your programming by creating a folder called **php_tutorials** and then another folder called **tutorials01**. As you progress through the tutorials you can then place your files into these folders.

(1) Create a new document in your text editor. If you are using Dreamweaver create a web page which will look similar to the following:

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1" />
<title>Untitled Document</title>
</head>

<body>
</body>
</html>
```

(2) Add the opening and closing **PHP** tags just before the closing body tag.

(3) Save the file as **example1.php**

01_introduction/example1.php

(4) Inside the opening and closing tags enter the following:

echo("PHP programming");

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1" />
<title>A PHP Web Page</title>
</head>

<body>

<?php
      echo("PHP programming");
?>
</body>
</html>
```

(5) Save and upload your file to a **PHP** compatible web server.

(6) Open file in a web browser. To display the page in the browser you will need the Url of the web page on your web server, so this will be something like http://www.yourserver.com/tutorials/file.php

SAMPLE

Note that the page must be saved with a .php extension to tell the server what type of processing it has to do.

This first script simply displays some text on the page using the **PHP** echo function. This page does little but it does test that your web server and ftp are working.

Note the following:

*** PHP** is case sensitive so you must enter echo as all lower case.
* The text to be displayed requires quote marks at the start and quote marks at the end.
* The line of code must be ended by a semicolon (;)

TASK 2 - echo and print

To display information on your web browser we use **echo()** and **print()** statements. There are no real differences between the echo and print statement so most programmers use **echo().**

(There is actually a special version of print which is **printf().** This is to allow formatting of data such as dates and times).

An example of **echo()**:

```
echo 'There was an error connecting to the database';

echo "Thank you for your submission to our forum";
```

You may use single or double quotation marks with either function and each line must end with a semicolon **(;)**.

One issue is displaying single and double quotation marks on the web page, so the following will generate an error:

```
echo "He asked, "What is your name?"";

Parse error: syntax error, unexpected T_STRING, expecting ',' or ';'
in /mnt/sw/22180/web/tutorial_docs/php/escape.php on line 11
```

```
echo 'Paul's house.';

Parse error: syntax error, unexpected T_STRING in
/mnt/sw/22180/web/tutorial_docs/php/escape.php on line 13
```

SAMPLE

There are two solutions to this problem.

First, use single quotation marks when printing a double quotation mark and vice versa.

```
echo “Paul’s house.”;
```

Or you can ***escape*** the character by preceding it with a backslash like this:

```
echo 'Paul\’s house.';
```

The following is a further example of the **echo** statement using the built in **date** function.

(1) Open the previous page **example1.php** in your text editor.

(2) Between the PHP tags add a simple message.

echo date("d m y H:i:s ");

(3) Open the file in your Web Browser to test the script is working properly.

If you do a search on the internet for “PHP date” you should get a detailed description of the way this function works and the different formatting that it can display.

Try modifying the format elements to display the date in different ways such as T**uesday 10th January 2013**

In **example1.php** you can add your new PHP commands into the same **<?php .. ?>** area as previously or you can create another **<?php … ?>** start and end tags within the same document.

SAMPLE

PHP documentation

The main source of documentation on the web is the PHP manual. It is important to understand how functions are written in PHP documentation because you will be continuously coming across it.

http://php.net/manual/en/index.php

If you do an internet search for 'PHP date' you should get to this page:

http://php.net/manual/en/function.date.php

It tells you the function name, which PHP versions it is available in, a description and the function in a descriptive notation:

string date (string $format [, int $timestamp = time()])

string means that it returns a string,

date is the function name

$format refers to a set of parameters which will be listed further down

Anything in [] brackets are optional parameters.

SAMPLE

TASK 3 - PHP syntax

*** Variables**

All variables should begin with a $ symbol. e.g. $myName

**** Variables are case sensitive***

PHP in built functions and user defined variables are case insensitive, although variables are case sensitive, so $myName is different to $myname.

**** PHP is white space insensitive***

PHP will ignore white space characters, tabs etc. in the same way as HTML pages do. This means you can space out your PHP code using tabs and spaces and on different lines to make it easier to read.

**** Statements and expressions are terminated by a semicolon character***

A statement is any expression followed by a semicolon (;)

**** Comments can be single or multiline comments***

The multiline comment is the same as in the C language and is:

```
/* this is
a comment
*/
```

To comment a single line use # or //

```
# This is a comment
```

The second uses two slashes.

```
//This is also a comment
```

SAMPLE

2 - Variables in PHP

What's in this chapter:

* Understanding variables and variable data types.
* **PHP** built in server variables.
* Functions for manipulating strings.
* Using **PHP** on a **HTML** page.
* Formatting data outputs.

Variables are used to store data during the processing of the web page. The data can then be displayed on the web page or calculations performed on it using mathematical calculations like multiplication and division. A variable is simply an area of memory to store information which has been assigned a particular identifier by the programmer.

Some examples of variables might be:

45 - a whole number representing someone's age
Peter - a string representing a person's name
2013-01-28 - a string representing a date
22.40 - a decimal number representing the cost of an item

SAMPLE

Variable Types

A variables name - also called its identifier – must start with a dollar sign ($), for example: $address.

* Variable names are case sensitive so $address and $Address are two different variables.

* The first character after the $ must be a letter or underscore, followed by any number of letters, numbers, or underscores.

Variables are assigned values using the equals sign (=) like the following:

$address = “London”;

There are a total of 8 data types:

* **Integers** – whole numbers
* **Doubles** – floating point numbers
* **Booleans** – true or false
* **Null** – a special type with just the value of NULL
* **Strings** – sequences of characters
* **Arrays** – indexed collections
* **Objects** – instances of classes
* **Resources** – references for database connections or files

In **PHP** you do not require to declare variables or give them data types as you may have to do with other programming languages like Visual Basic. **PHP** automatically converts the variable to the correct data type, depending on its value. So a **PHP** variable does not know in advance whether it will be used to store a number or a string. This is termed ‘loosely typed’.

There are advantages and disadvantages with a loosely typed language. Coding is simpler as you don’t have to worry about data types, but it can cause potential problems when you may use a variable expecting one data type and it actually holds a different data type.

Some examples of variables and their use in **PHP** would be:

```
$name = "Paul";
$age = 27;
$isStudent = true;
$cost = 1.99;
$currentDate = "2014-01-05";
```

SAMPLE

Note that strings and dates are enclosed with quote marks while numbers do not have quote marks.

There is no specification on the way to define a variable name, but quite often we may use something called the **Camel Notation**.

An example of this is:

```
$firstName = "Paul";
```

The **Camel Notation** is where the first word starts with a lower case, and the following words start with an upper case.

Another convention is where we prefix the name with the data type, so we might use the following:

```
intAge = "47";
strFirstName = "Paul";
```

This helps us to identify the purpose of a particular variable.

Whatever system you choose, make sure that you are consistent throughout your script.

SAMPLE

TASK 1 - Some examples of variables

To demonstrate the use of variables we will do a simple calculation.

(1) Create a new **PHP** document in your text editor. If you are using Dreamweaver, it will automatically create some HTML code for you.

(2) Save the file with the name **variables.php**

(3) Within the <body> tags of the html page, add your **PHP** tags.

```
<?php
?>
```

* Create a variable with your name and echo it to the web page.

* Create a variable called TAX and set it to 0.2 (which is 20%)
* Create a variable called productcost and set it to 34.20
* Calculate the TAX, display the TAX and display the new price of the item.

```
$name = "Paul";
echo("My name is $name<br/>");

$tax = 0.2;
$productcost = 34.20;
$taxcost = $productcost * $tax;
$cost = $taxcost + $productcost;

echo("TAX is $taxcost<br/>");
echo("Total cost is $cost<br/>");
```

(4) Upload the page to the web server, run it in the browser and check the results.

TASK 2 - Pre-defined variables

PHP includes a number of pre-defined variables which are used to display information about the system that you are using. We can use these to work with, or to display such things as your IP address, the name of the file that you are working on and so on.

These pre-defined variables are accessed using the **$_SERVER** array.

(1) Open the **variables.php** file in the editor.

(2) Within the <body> tags, add another set of PHP tags

```
<?php

?>
```

(3) Create the following variables using **$_SERVER** as follows:

```
$file = $_SERVER['PHP_SELF'];
$user = $_SERVER['HTTP_USER_AGENT'];
$address = $_SERVER['REMOTE_ADDR'];
```

This script will use three variables which come from the **$_SERVER** array.

SAMPLE

The name of the script being run is accessed using PHP_SELF:

```
$_SERVER['PHP_SELF'];
```

The Web Browser and other details of the user accessing the script are accessed using HTTP_USER_AGENT:

```
$_SERVER['HTTP_USER_AGENT'];
```

The IP address of the user accessing the script is accessed using REMOTE_ADDR:

```
$_SERVER['REMOTE_ADDR'];
```

4. Enter the following code in the page.

```
echo "<p>The current filename is: <strong>$file</strong>.</p>";

echo "<p>The user details are: <strong>$user</strong> <br/>with
address: <strong>$address</strong></p>";
```

5. Upload the file to the web server and open the file in the Web Browser to test the script is working properly.

TASK 3 - Strings and string manipulation

Strings hold text information as words and sentences and are enclosed in quote marks. Anything stored in quote marks becomes a string.

The follow are examples of strings:

```
'Paul'

"My name is Paul"

'July 5, 2012'
```

We assign a string variable to a variable name in the following way:

SAMPLE

```
$first_name = 'Paul';
$today = 'July 5, 2012';
```

In order to print out the value of a string, use either **echo()** or **print()**:

```
echo $first_name;
echo "Hello, $first_name";
```

Another example of a string is:

```
$age = "25";
```

As it is enclosed by quote marks it becomes a string.

To do mathematical operations on this value may give unexpected results as you should normally use:

```
$age = 25;
```

or convert the string to a numerical value:

```
$age = "25";
$intAge = (int)$age;
```

Whenever you do any numerical calculations, make sure you have not defined the numbers as strings.

(1).To illustrate the use of strings, using the same PHP file **variables.php** and within the **PHP** tags, create three variables by adding the code below inside the PHP tags.

SAMPLE

```
//Create the variables
$first_name = 'Paul';
$last_name = 'Gibbs';
$work = 'Super Software Company';
```

(2) Add the echo statement below, on one line and test the script in your Web browser.

```
echo "<p>$first_name $lastname works at $work</p>";
```

TASK 4 - Some string functions

One of the most common string functions is the Concatenation operator which is **(.)** and has the effect of adding two strings together:

```
$town = 'Trowbridge';
$county = 'Wiltshire';
$address = $town . ', ' . $county;
```

The **$address** variable now has the value ***Trowbridge, Wiltshire***

so that a comma and a space are added to the line.

Concatenation works with strings or numbers.

```
$address = $town . ', ' . $county . 'BA14 OES';
```

(1) To test the concatenation of strings we can use the same PHP file variables.php and within the PHP tags add the following code:

```
$myname = $first_name . ' ' . $last_name;
echo "$myname works at $work.";
```

(2) Open and test in your Web browsers.

If you do a search in Google for **PHP string functions** it should return the page

SAMPLE

http://php.net/manual/en/ref.strings.php which lists all the functions available. Here are a few of the more common ones:

Trimming Strings functions	
trim()	Removes whitespace at beginning and end of a string.
ltrim()	Removes whitespace at the beginning of a string.
rtrim()	Removes whitespace at the end of a string.

Presentation functions	
htmlentities()	Escapes all HTML entities.
strtoupper()	Converts a string to uppercase
strtolower()	Converts a string to lowercase.
ucfirst()	Converts the first character of a string to uppercase.
ucwords()	Converts the first character of each word in a string to uppercase.

Converting Strings and Arrays functions	
explode()	Splits a string into an array on a specified character or group of characters.
implode()	Converts an array into a string, placing a specified character or group of characters between each array element.
join()	Same as implode().

Substrings functions	
substr(str,pos)	Returns the substring from the character in position pos to the end of the string
substr(str,-len)	Returns the substring from len characters from the end of the string to the end of the string
substr(str,pos,len)	Returns a len length substring beginning with the character in position pos.
substr(str,pos,-len)	Returns a substring beginning with the character in position pos and chopping off the last len characters of the string.
strstr(haystack,needle, before_needle)	If the third argument (before_needle) is false (default), then it returns the part of the haystack from the needle onwards. If the third argument (before_needle) is true, then it returns the part of the haystack before the needle. The needle can be a string or an integer (or a number that can be converted to an integer).
stristr(haystack,needle, before_needle)	Same as strstr(), but case insensitive
strpos(haystack,needle)	Finds the position of the first occurrence of a

SAMPLE

	specified needle in a haystack (string). The needle can be a string or an integer (or a number that can be converted to an integer).
strrpos(haystack,needle)	Finds the position of the last occurrence of a specified needle in a haystack (string). The needle can be a string or an integer (or a number that can be converted to an integer).
str_replace()	Replaces all occurrences of one string with another string.

Comparing Strings functions	
strcmp()	Compares two strings. Returns < 0 if str1 is less than str2, > 0 if str1 is greater than str2, and 0 if they are equal.
strcasecmp()	As above but case sensitive.
strlen()	Returns the length of a string.

Exercise 2.1

This exercise is to try out str_replace:

```
$mystring = "This script is written by name";
```

Look up **str_replace** on the web and use it to substitute the text of **name** in the string **$mystring** with your name, and display the result on the web page

Exercise 2.2

This exercise is to try out **strpos**:

If we have a string which is http://www.wwithinweb.com and we want to check that http has been entered in the string, we can use strpos to test for this.

Look up **strpos** on the web and use it to check for the occurrence of http in a string.

SAMPLE

TASK 5 - Numeric date types

PHP has both INTEGER and DOUBLE (floating-point decimal number) types, so valid numbers would be:

```
$an_integer = 27;
$a_double = 2.3456;
$e_notation = 2.3e2;
```

Numbers are not quoted in which case they would be strings, nor do they include commas to indicate thousands.

PHP has fewer data types than other programming languages so making it easier to work with.

The following is a list of maths operators which are used to perform calculations:

Simple mathematical operators

+ Addition, for example $result = $a + $b
- Subtraction, for example $result = $a - $b
* Multiplication, for example $result = $a * $b
/ Division, for example $result = $a / %b
% Modulus, for example $result = $a % $b
++ Increment, for example $result = ++$a
-- Decrement, for example $result = --$a

PHP has the standard arithmetic operators and many other functions to deal with numbers. Two that we will look at here are **round()** and **number_format().** The former rounds a decimal either to the nearest integer

```
$n = 2. 3456;
$n = round($n);   //will return 2
```

Or:

```
$n = 2.183645;
$n = round($n, 3);       //will return 2.183
```

SAMPLE

Round also has an optional parameter to define the type of rounding of

PHP_ROUND_HALF_UP
PHP_ROUND_HALF_DOWN
PHP_ROUND_HALF_EVEN
PHP_ROUND_HALF_ODD

The **number_format()** function turns a number into the more commonly written version, grouped into thousands using commas. For example:

```
$n = 493849;
$n = number_format ($n);      //will return 493,849
```

Exercise 2.3

As a simple exercise, we want to do a calculation on a salary to calculate tax and to display the result in our web browser

Assume the salary is £25,000 and the tax rate is 20%

First calculate the tax on the salary, and then calculate the salary less tax.

Display the two values to the web browser.

TASK 6 - Constants

Constants are values which cannot be changed during the execution of the script. Constants can be assigned any single value, a number or a string of characters.

To create a constant, you use the **define()** function.

```
define ('NAME', 'value');
```

By convention, constants are named using all capital letters. Also constants do not use the initial dollar sign (**$**) as with variables.

An example of displaying a constant is:

```
define ('USERNAME' , 'gibbpv');
echo "Hello, " . USERNAME;
```

SAMPLE

Constants are often used for configuration settings, so you may see it used with setting database username / passwords and similar functions.

TASK 7 - Single and double quoted strings

We can use single and double quote marks in **echo()** and **print()** statements, however, there is a difference in how they behave.

In **PHP**, values enclosed in single quote marks are **not** interpreted and are displayed 'as is'. However, values that are enclosed in double quote marks are interpreted.

If we want to display certain characters within double quote marks we have to **escape** those characters, that is place the \ character before it.

Escaped Characters Code

\"	Double quotation mark
\'	Single quotation mark
\\	Backslash
\n	Newline
\r	Carriage return
\t	Tab
\$	Dollar sign

As an example, assume you have:

```
$var = "Paul";
```

Then:

```
echo "\$var is equal to $var";
```

will display:

```
$var is equal to Paul
```

While:

```
echo '\$var is equal to $var';
```

will display

SAMPLE

```
\$var is equal to $var.
```

You can see in the above examples, that double quote marks will display the value of $var, while single quote marks will display $var exactly as it is.

TASK 8 - PHP and HTML

It is common in **PHP** to want to output HTML code and the easy way to do that is to use single quote marks when printing HTML with **PHP**.

```
echo '<table width="80%" border ="0" cellspacing ="2" cellpadding
="3" align= "center" >';
```

If you were to print out this HTML using double quotation marks you would have to escape all the double quotation marks in the string.

```
echo "<table width=\"80%\" border =\"0\" cellspacing =\"2\"
cellpadding =\"3\" align= \"center\">";
```

Another way of printing out HTML which is sometimes a preferred method instead of entering in echo statements is to do the following:

```
<?php
….. some code ……
?>
        <img src="delete.gif" width="10" height="10" />
<?php
….. some code ……
?>
```

In the above example we use <?php … ?> tags to go from PHP and HTML instead of writing the HTML in the **PHP** code.

SAMPLE

TASK 9 - Formatting the outputs and data types

As a recap, **PHP** does not require you to declare variables or give them data types as you may have to do with other programming languages like Visual Basic. **PHP** automatically converts the variable to the correct data type, depending on its value.

So $name = "fred"; works by just assigning the value to the variable name and it understand it as a string.

To forcibly convert a variable to a certain type, either cast the variable or use the **settype()** function on it.

The **printf()** function can be used to output to a particular format to make them look more presentable. So for example:

```
printf("%d", "17,999");
```

Will display **17.**

The general form of printf is:

```
printf( type specifier, value );
```

(Note that printf returns an integer value of the length of the string while sprintf returns a string).

The example above uses the **%d** format specifier. This formats the value as a signed decimal integer. The 'd' is known as a type specifier which says what type the output data should be. **printf()** supports a wide range as follows:

b Format the argument as a binary integer (e.g. 11000110)
c Format the argument as a character with the argument's ASCII value
d Format the argument as a signed decimal integer
e Format the argument in scientific notation (e.g. 9.344e+3)
f Format the argument as a floating-point number using the current locale settings (e.g. in France a comma is used for the decimal point)
F As above, but ignore the locale settings
o Format the argument as an octal integer
s Format the argument as a string
u Format the argument as an unsigned decimal integer
x Format the argument as a lowercase hexadecimal integer (e.g. 2faf47)
X Format the argument as an uppercase hexadecimal integer (e.g. 2FAF47)

SAMPLE

Exercise 2.4

If Australia has 6 states and 10 territories print this out using a **printf** statement :

Example `printf( "Australia comprises %d states and %d territories", 6, 10 );`

Exercise 2.5

Write a program to convert 20 degrees Fahrenheit to Centigrade and display the results as a floating point number:

Temperature in Centigrade = (5 / 9) * ($Tf - 32)

Exercise 2.6

Write a script that has three variables $a, $b and $c. Assign some numerical values to each of these three variables and then calculate the average. Display the result using an echo statement and display as a floating point number to 2 decimal places – you will need to use the number_format function.

SAMPLE