PHP Tutorials: Programming with PHP and MySQL

Paul Gibbs
PHP Tutorials – Programming with MySQL and PHP

A set of tutorials derived from a series of lectures

Paul Gibbs – Ilfracombe, North Devon, England
Table of Contents

Preface

Basic Programming Terms

Book Version

PHP Tutorial 1 – Introduction to PHP
  1.1 The PHP language
  1.2 What you need
  1.3 The processing of PHP
  1.4 A simple PHP script
  1.5 Error reporting
    TASK 1 - A PHP page
    TASK 2 - echo and print
  1.5 PHP documentation
    TASK 3 - PHP syntax

PHP Tutorial 2 – Variables in PHP
  TASK 1 - Some examples of variables
  TASK 2 - Pre-defined variables
  TASK 3 - Strings and string manipulation
  TASK 4 - Some string functions
  TASK 5 - Numeric date types
  TASK 6 - Constants
  TASK 7 - Single and double quoted strings
  TASK 8 - PHP and HTML
  TASK 9 - Formatting the outputs and data types

PHP Tutorial 3 - Forms and PHP
  Part 1 - HTTP is Stateless
  Part 2 - HTML Forms
    TASK 1 - Create an HTML Form
    TASK 2 - Handling HTML form data
    2.1 Other FORM input types
    TASK 3 - Handling Magic Quotes
    TASK 4 - Conditional processing
    TASK 5 - Adding Conditionals to our script
    TASK 6 - Validating Form Data
TASK 7 - The switch statement

PHP Tutorial 4 – Arrays and Loops

Part 1 - Creating Arrays
1.1 Indexed arrays
1.2 Associative arrays
1.3 Assigning a range of values
1.4 Getting the Size of an Array

Part 2 - Accessing arrays
2.1 Accessing individual elements of an array
2.2 Traversing through an array

Part 3 - Sorting Arrays

Part 4 - Testing arrays
   TASK 1 - Testing an associative array
   TASK 2 - Testing the foreach loop
   TASK 3 - HTML and an associative array
   TASK 4 - Using arrays

Part 5 - The for and while Loops
   TASK 5 - Using loops and if statements

Part 6 - Multi-dimensional Arrays

PHP Tutorial 5 – Basic PHP Structures

Part 1 - Using include Files
   TASK 1 - Including Multiple Files

PHP Tutorial 6 - Functions

Part 1 - User defined functions
Part 2 - Functions that take arguments
Part 3 - Setting default argument values in a function

Part 5 - Variable Scope
   5.1 The global statement

Part 6 - Date and Time Functions
   6.1 date()
   6.2 getdate()
TASK 3 - Selecting dates

TASK 4 - Handling dates between mySQL and PHP

PHP Tutorial 7 - Posting Forms
Part 1 - Handling HTML Forms
  TASK 1 - Posting to itself
Part 2 - Recap on handling PHP and HTML on one page
  TASK 2 - Retaining data in forms
  TASK 2 - Questions

PHP Tutorial 8 - An Email Form
Part 1 - Email Handling
Part 2 - The mail() function
  TASK 1 - A Test Email Script
  TASK 2 - A basic PHP email form
Part 3 - Send email further development

PHP Tutorial 9 – JavaScript Submit
Part 1 – Submit a form with JavaScript
Part 2 – Submit a form with JavaScript input validation

PHP Tutorial 10 – SQL and MySQL
Part 1 - MySQL database and phpMyAdmin
Part 2 - Creating a database in MySQL
Part 3 - Database user permissions
  3.1 Creating a database user
    3.1.1 Basic method
    3.1.2 A more concise way of creating a user
Part 4 - Creating tables in MySQL
  4.1 MySQL Documentation
Part 5 - SELECT SQL statements
Part 6 - Conditional SQL statements
Part 7 - INSERT SQL statements
Part 8 - UPDATE SQL statements
Part 9 - DELETE SQL statements
Part 10 - Join tables
  10.1 The Inner Join
  10.2 The Outer Join
10.3 Some further points about joins

Part 11 - GROUP BY and HAVING
Part 12 - The BETWEEN operator
Part 13 - The IN operator
Part 14 - Sub queries
Part 15 - Views
Part 16 - Transactions

PHP Tutorial 11 – The Staff Table

Part 1 - The Staff table
  1.1 SQL Exercises based on the staff table
  1.2 SQL Exercises based on the Person, Student Grade and Course tables

PHP Tutorial 12 – Database Access

Part 1 - SQL selecting and updating – standard methods
  1.1 A note about mysqli functions
  TASK 1 - Create table data
  TASK 2 - Reading database using mysqli_fetch_assoc
  TASK 3 - Modifying database using mysqli_query
Part 2 – SQL selecting and updating – PDO objects
  TASK 4 - Simple SQL Fetch and PDO Objects
  TASK 5 - SQL modify
  TASK 6 - SQL modify with try / catch and PDO objects
  TASK 7 - Other example of SQL Fetch with try / catch
  TASK 8 - SQL fetch and PDO objects reading into a $table array
  TASK 9 - SQL fetch using named values

PHP Tutorial 13 – Using PHP and MySQL

Part 1 - The MySQL Database and Table
  TASK 1 - Creating the table
Part 2 - Building the connection script and form
  TASK 2 - Creating the database connection script
  TASK 3 - Creating the Feedback Page
  TASK 4 - Improving the feedback form
  TASK 5 - Further exercises

PHP Tutorial 14 – Further PHP and MySQL

Part 1 - The connection string
TASK 1 - Connection string include file
Part 2 - Retrieving data from a database table
    TASK 2 - Viewing the feedback data
Part 3 - Improving security in the code
    TASK 3 - Using mysqli_real_escape_string();
    TASK 4 - Adding in the function
    3.1 Complete script for feedback.php
Part 4 - Counting records and mysqli_num_rows

PHP Tutorial 15 – OpenCart Installation

PHP Tutorial 16 - Error Handling and Debugging
    Part 1 - Types of Errors
    Part 2 - Error handling in PHP for security
    Part 3 - Types of Errors
    Part 4 - Choosing what errors to report in PHP
        Part 4.1 Error Reporting Levels
            TASK 1 - Controlling error reporting in a web application
    Part 5 - Debugging PHP
        5.1 Use the print() or echo() and exit() functions
        5.2 Use the print_r() function
        5.3 Reducing the code down to its minimum
        5.4 Reduce duplicate code
    Part 6 - try / catch PHP 5
        TASK 2 - Exercise in try / catch errors
    Part 7 - Debugging SQL
    Part 8 - Other methods of debugging

PHP Tutorial 17 – Cookies and Sessions
    Part 1 HTTP is stateless
    Part 2 - Sessions and Cookies
    Part 3 - Cookies
        3.1 Showing Cookies in Internet Explorer
        3.2 Setting Cookies
        3.3 Accessing Cookies
        3.4 Cookie Parameters
        3.5 Delete a Cookie
Part 4 - Sessions
  TASK 1 - A session test program
  TASK 2 - A session array script
Part 5 – Using sessions to create a login admin system
  TASK 3 - Create a login form
  TASK 4 - Create an authentication include file
  TASK 5 - Create dummy php pages
  TASK 6 - Create a menu page
  TASK 7 - Testing the login system

PHP Tutorial 18 – Modifying Records
Part 1 - Passing values between pages
  TASK 1 - Adding in edit and delete links
  TASK 2 - Creating the edit_feedback.php page
  TASK 3 - Creating the delete_feedback.php page

PHP Tutorial 19 - Classes
  Part 1 Basic principles
  TASK 1 - Object Orientated Terminology
  TASK 2 - Inheritance
  TASK 3 - Constructors and Destructors

PHP Tutorial 20 - File Handling
  Part 1 - File permissions on the server
    1.1 Short explanation of file permissions
    1.3 So what does 775 and 664 mean?
  Part 2 - Working with text files
    TASK 1 - Opening a file
    TASK 2 - Writing to the file
    TASK 3 - Reading from the file
  Part 3 - File and folder manipulation
  Part 4 – Example file operations
  Part 5 - Uploading files to a web server

PHP Tutorial 21 – Regular Expressions and Data Validation
  Part 1 - Introduction
    1.1 Meta characters
    1.2 Quantifiers
1.3 Character Classes

Part 2 - Some match examples

2.1 Example - simple search
2.2 Example - search with following characters
2.3 Example - validate phone function
2.4 Example - validate input function

Part 3 - Validation input for PHP 5.2.0 onwards

PHP Tutorial 22 - PHP Security

Part 1 - Web Server Configuration parameters

1.1 safe_mode
1.2 expose_php
1.3 register_globals
1.4 display_errors
1.5 log_errors

Part 2 - PHP Code scripting security

2.1 XSS
2.2 SQL Injection
2.3 Ways to Counter SQL Injection
2.4 Code Injection
2.5 Email Injection
2.6 Filters

PHP Tutorial 23 – Introduction to jQuery

Part 1 The jQuery framework

1.1 Uses of jQuery

Part 2 jQuery on your web page

2.1 Download the latest jQuery file
2.2 Use a hosted js file such as from Google libraries

Part 3 Using jQuery

Part 4 Selecting Page Elements

Part 5 Event Handling

Part 6 A simple calculator

Part 7 DOM Manipulation

Part 8 Other examples

8.1 UI Accordion interface
8.3 Menu Interface
8.3 Date Picker
8.4 Image Galleries, Sliders and Carousels
8.5 Twitter Bootstrap
8.6 AJAX and jQuery

PHP Tutorial 24 - htaccess and php.ini files

Part 1 Introduction - .htaccess

Part 2 Things that can be done with an .htaccess file

Part 3 To use an .htaccess file

Part 4 Examples of .htaccess files
   4.1 301 is permanent redirect
   4.2 302 is temporary redirect
   4.3 Defining a custom 404 error page
   4.4 Stop users displaying the directory listing of a web site
   4.5 Stop users displaying the details of an .htaccess file

Part 5 To protect your admin area you can create an .htaccess / .htpasswd file.
   5.2 Create the .htpasswd username / password file

Part 6 The .php.ini file
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 to use it.

* Introduction - Variables, programming techniques and so on
* Forms and PHP - Posting data between forms
* Basic PHP structures - include files and how to use them
* SQL and MySQL - Querying database tables using SQL
* Using PHP and MySQL - Connecting to MySQL with PHP
* Further PHP and MySQL - More PHP and MySQL
* 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
* SQL Injection - Some methods to overcome this issue
* jQuery - An introduction to jQuery

Using the code examples

All the code examples with answers to the exercises may be downloaded to your computer from this link [http://www.withinweb.com/phptutorials/phptutorials.zip](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.
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.

Book Version

The current version is January 2014:

* 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.
PHP Tutorial 1 – Introduction to PHP

1.1 The PHP language

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.

PHP versions are regularly released. At present, the latest version is 5.5.

1.2 What you need

* A web server such as Apache.
* PHP
* A MySQL database and administration interface.
* Web Browsers – Fire Fox, Internet Explorer etc.
* A Text Editor
* An FTP program (such as FileZilla) if using a remote Server although some editors such as Dreamweaver have FTP built in.

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.

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/](http://www.easyphp.org/) which can be used for development.

A suitable Text editor might be NotePad or Dreamweaver (commercial).

Kompozer is a free HTML editor [http://www.kompozer.net/](http://www.kompozer.net/) which also has the ability to create PHP documents.

1.3 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.

With a purely HTML-generated site, 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 creating of dynamic web pages which can display different data on a web page depending on the programming of the PHP script.

1.4 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

```html
<!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.

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
<?php
?
```

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

1.5 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.

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

4. Inside the opening and closing tags enter the following:

```
echo("PHP programming");
```

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

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()`:

```php
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:

```php
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
```

There are two solutions to this problem.

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

```php
echo "Paul\'s house.";
```

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

```php
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.

   ```php
   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 **Tuesday 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.

### 1.5 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.


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


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

```php
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.

### TASK 3 - PHP syntax

* **Variables**

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

* **Variables and functions are case sensitive**
PHP in built functions, user defined functions, and user defined 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:

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

To comment a single line use # or //</p>

```php
# This is a comment
```

The second uses two slashes.

```php
//This is also a comment
```
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. 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

**Part 1 - Variable Types**

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

* The first character after the dollar sign must be a letter or an underscore (it cannot be a number).
* The variable’s name can contain a combination of strings, numbers and the underscore, e.g. $my_address1.
* Variable names in PHP are case sensitive so $address and $Address are two different variables.

Variables can be assigned values using the equals sign (=), also called the **assignment operator**.

Variables have no intrinsic type other than the type of their current value.

There are a total of 8 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** – database references for connections

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.

Some examples of variables and their use in PHP would be:
Note that strings and dates are enclosed with quote marks while numbers do not have quote marks.

Some examples of variable names:

- intAge
- strName

Camel notation would be firstName as a variable name.

**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
<?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";
$sage = 27;
$isStudent = true;
$cost = 1.99;
$currentDate = "2013-01-05";

$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
   <?php
   ?>
   ```

3. Create the following variables using $_SERVER as follows:

   ```php
   $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.

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

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

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

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

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

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

4. Enter the following code in the page.

   ```php
   echo "<p>You are running the file <strong>$file</strong>.</p>";
   echo "<p>You are viewing this page using <strong>$user</strong> <br/> <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:

```
$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.
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:

```php
//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.

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

**TASK 4 - Some string functions**

The Concatenation operator is (.) and has the effect of adding two strings:

```php
$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.

```php
$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:

```php
$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 [http://php.net/manual/en/ref.strings.php](http://php.net/manual/en/ref.strings.php) which lists all the functions available. Here are a few that you may use more often:
<table>
<thead>
<tr>
<th><strong>Trimming Strings functions</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>trim()</td>
<td>Removes whitespace at beginning and end of a string.</td>
</tr>
<tr>
<td>ltrim()</td>
<td>Removes whitespace at the beginning of a string.</td>
</tr>
<tr>
<td>rtrim()</td>
<td>Removes whitespace at the end of a string.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Presentation functions</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>htmlentities()</td>
<td>Escapes all HTML entities.</td>
</tr>
<tr>
<td>strtoupper()</td>
<td>Converts a string to uppercase</td>
</tr>
<tr>
<td>strtolower()</td>
<td>Converts a string to lowercase.</td>
</tr>
<tr>
<td>ucfirst()</td>
<td>Converts the first character of a string to uppercase.</td>
</tr>
<tr>
<td>ucwords()</td>
<td>Converts the first character of each word in a string to uppercase.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Converting Strings and Arrays functions</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>explode()</td>
<td>Splits a string into an array on a specified character or group of characters.</td>
</tr>
<tr>
<td>implode()</td>
<td>Converts an array into a string, placing a specified character or group of characters between each array element.</td>
</tr>
<tr>
<td>join()</td>
<td>Same as implode().</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Substrings functions</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>substr(str,pos)</td>
<td>Returns the substring from the character in position pos to the end of the string</td>
</tr>
<tr>
<td>substr(str,-len)</td>
<td>Returns the substring from len characters from the end of the string to the end of the string</td>
</tr>
<tr>
<td>substr(str,pos,len)</td>
<td>Returns a len length substring beginning with the character in position pos.</td>
</tr>
<tr>
<td>substr(str,pos,-len)</td>
<td>Returns a substring beginning with the character in position pos and chopping off the last len characters of the string.</td>
</tr>
<tr>
<td>strstr(haystack,needle, before_needle)</td>
<td>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).</td>
</tr>
<tr>
<td>stristr(haystack,needle, before_needle)</td>
<td>Same as strstr(), but case insensitive</td>
</tr>
<tr>
<td>strpos(haystack,needle)</td>
<td>Finds the position of the first 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).</td>
</tr>
<tr>
<td>strrpos(haystack,needle)</td>
<td>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).</td>
</tr>
<tr>
<td>str_replace()</td>
<td>Replaces all occurrences of one string with another string.</td>
</tr>
</tbody>
</table>
Comparing Strings functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>strcmp()</td>
<td>Compares two strings. Returns &lt; 0 if str1 is less than str2, &gt; 0 if str1 is greater than str2, and 0 if they are equal.</td>
</tr>
<tr>
<td>strcasecmp()</td>
<td>As above but case sensitive.</td>
</tr>
<tr>
<td>strlen()</td>
<td>Returns the length of a string.</td>
</tr>
</tbody>
</table>

Exercise 3.1

This exercise is to try out str_replace:

```php
$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 3.2

This exercise is to try out strpos:

If we have a string which is http://www.withinweb.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.

TASK 5 - Numeric date types

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

```php
$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.

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

```php
$n = 2.3456;
$n = round($n); // will return 2
```

Or:

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

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:

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

**Exercise 4.1**

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.

```php
define('NAME', 'value');
```
By convention, constants are named using all capital letters. Also constants do not use the initial dollar sign ($)

Constants are normally printed in the following way:

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

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 treated literally, whereas those 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:

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

Then:

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

will display:

```
$var is equal to Paul
```
While:

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

will display

```
\$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.

```php
' <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.

```php
"<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
<?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.
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)

Exercise 7.1

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 7.2**

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 7.3**

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.