PROGRAMMING IN A SCRIPTING LANGUAGE

We will be using PHP which is a server-side script language. In actual fact it is one of the world's most popular programming languages for use on web servers. PHP officially stands for "PHP:Hypertext Processor", but it is also known by its original name "Personal Home Page". You can find information about PHP, its newest version and older versions, code definition etc. at the official web site http://www.php.net

Rasmus Lerdorf created PHP in 1994. It is estimated that by late 1996 PHP was in use on at least 15,000 web sites around the world. By mid-1997 this number had grown to over 50,000. Today PHP is in use on over 100 million sites around the world.

What can PHP do?

You can use PHP to read and write to files on the server. For example a hit counter for a web page. PHP can connect to a database and execute SQL queries as well as create online databases, guest books, chat rooms, file uploading, shopping carts, etc. Perhaps the strongest and most significant feature in PHP is its support for a wide range of databases. Writing a database-enabled web page is incredibly simple. In addition to MySQL, the following databases are currently supported:
Adabas D, InterBase, PostgreSQL, dBase, FrontBase, Solid, Empress, mSQL, Sybase, FilePro (read-only), Direct MS-SQL, Velocis, IBM DB2, Unix dbm, Informix, ODBC, Ingres, Oracle (OCI7 and OCI8),

Why PHP?

PHP is often recommended as the first server-side language to learn as it is:

- intuitive and fairly easy to read
- highly configurable and flexible
- a great companion language for MySQL (a free, fully-featured database software)
- portable – PHP can be run on Windows, Mac OS and Linux
- it's free

HTML and PHP

All www pages are written in HTML (or XHTML). Although some files may have different file extensions (such as .php, .cfm, .aspx) their output is still HTML. HTML is sent to the web browser which interprets it and displays it on your monitor.

PHP is a server-side scripting language, which means the web-server does all the work, not the browser. In other words, the PHP interpreter is at the server end and not at the client PC end where the browser lives.

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server.

Using PHP

You can add PHP to your web pages in a number of ways:

```
1. <?php
      echo "Hello World";
    ?>

2. <?
      echo 'Hello World';
    ?>

3. <?= "Hello World";
    ?>

4. <script language="php">
      echo 'Hello World';
    </script>
```

We will be using 1 and 3 from above.

Example1:
This is an example of a php file

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>Example 1</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>

<body>
  <p>Today’s message is
  <?php
      echo “Hello and Welcome to PHP”;
  ?>
  </p>
</body>
</html>
```

Most of this is plain HTML (XHTML). However, the line between <?php and ?> is written in php code.
PHP files have to be saved with a .php extension. This way the server knows to treat the file as a PHP file.

If you view the source code you will only see HTML code - you won’t see the PHP code. This is because the file is processed on the server and the server sends to the client (browser) the result, which is HTML code. For the above file, if you selected View -> Source from the menu bar, the following code would be displayed:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>Example 1</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>

<body>
  <p>Today's message is
    Hello and Welcome to PHP
  </p>
</body>
</html>
```

The server did all the work, processed all the code it had to, and returned only the HTML to the screen. The tags that do the work are called the **"delimiters"** (ie., <?php ……..?>).

<?php shows the start of the PHP code, and ?> tells you that's the end of the PHP code. The **delimiters** can be placed as many times as you like in the document.

Note:
- **After each line of code you need the semicolon. These separate the set of instructions from one another.**
- **PHP code is case sensitive.**

Inserting PHP Comments

A program interpreter will ignore a comment. Comments are there to be read by you or other programmers who may need to work on your program. Comments explain the code so that updating at a later time becomes an easier task.

There are two types of comments:

i) Single line comments

Use "//", for a single line comment. (The same convention used in Java and C++). For example:

```
echo 'This is very good';      // writes out my opinion
```

or use "#" for a single line comment:

```
echo "This is very good";      # writes out my opinion
```

ii) Multiple line comments

Use "/*" to denote the beginning of a comment and "*/" to denote end of a comment. For example:

```
/*  the following script was created by Mary Smith
    on the 25th June 2007  */
```

Exercise 1

1. You will create your first Web page as follows and save it as **ex1-1.php**. This script will write a line to the Web page.
   a) Use Dreamweaver to create this file.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>A JavaScript Example</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>

<body>
  <?php
      echo 'Welcome to PHP 1';  // you can use single quotes
      echo "Welcome to PHP 2";  // you can use double quotes
      echo 123456789;         // numbers don't need quotes
  ?>
</body>
</html>
```

   b) Upload to your /home/loginName/www/exercises directory on the Linux server
   c) View the file in the browser – enter the following URL:
   http://infoweb.hornsby.tafensw.edu.au/~loginName

2. Modify ex1-1.php by changing the code as follows and save it as **ex1-2.php.**

```
<?php
    echo 'Welcome to PHP 1<br />';  // <br /> will create a new line
    echo "Welcome to PHP 2<br />";  // <br /> will create a new line
    echo 123456789;         // numbers don't need quotes
?>
```

Displaying Literal Characters

To display double quotes or single quotes on the screen, precede this character by **"\\"** character, which will literally display the character that follows it. For example:

```
echo 'Please answer with a \'Yes\' or \'No\'.  Thank you.';
```

will display: Please answer with a 'Yes' or 'No'. Thank you.

Exercise 1 cont.

3. Create code and utilize the previous example. Save it as **ex1-3.php**

<u>Embedding HTML Tags in PHP</u>

PHP and other server sided scripts have the ability to create an HTML document ‘on the fly’. This means that the content of the HTML document is only decided when the page is downloaded from the server. The HTML code is created on the server and sent to the client (browser). JavaScript, on the other hand, is embedded in the HTML document and is interpreted by the browser at the client end.

You can embed any of the HTML tags within the PHP code. For example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>Example 2</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>

<body>
  <?php
      echo ‘<span style=”color:#FFFF00”>Hello world</span>’;
  ?>
</body>
</html>
```

<u>Exercise 2</u>

1. Design a web page, which displays “Hello from PHP” in the font verdana and the colour blue. Save the file as **ex2-1.php**

However, it is much more efficient and faster for the document if the PHP is separated from the HTML tags. Using the same example as above the code would look as follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>Example 2</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>

<body>
  <span style=”color:#FFFF00”>
    <?php
       echo ‘Hello world’;
    ?>
  </span>
</body>
</html>
```

<u>Exercise 2 cont.</u>

2. Use **ex2-1.php** and modify the code so HTML and PHP are separated. Save the file as **ex2-2.php**

Variables in PHP

A variable is used to store a piece of data. All variables start with the dollar sign. A variable can be any letter or number combination (including the character '_') up to 32 characters in length. It may begin with either a letter or an underscore ('_'), but not a number. Names cannot have a space.

For example:

```
$numvar = 3;
```

The word numvar begins with a dollar sign. This tells PHP it's a variable, and to assign it a value of 3.

```
$myvar = 'Welcome to PHP';
```

The word myvar begins with a dollar sign. This tells PHP it's a variable, and to assign it the value "Welcome to PHP".

Variable Types

PHP supports the following variable types:

Boolean	holds true or false
integer	holds numbers like -1, 0, 5, etc
float	holds decimal numbers like 4.12345
string	holds text
array	holds array of data
object	holds programming objects
null	holds a value of null

The type of variable is usually decided at run time by PHP depending on the context in which that variable is used. If you need to force a variable to be converted to a certain type you can use the settype() function on it.

A variable does not need to be declared before it can be used. PHP automatically creates the correct type of variable based on the type of data that is put in it.
For example: `$i = 'hello'` results in a string variable,
whereas `$i = 1` results in an integer variable.
Note `$i = 1.234` would result in a float variable.

Exercise 3

1. Create this code and save it as **ex3-1.php**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title>Exercise 1-1</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>

<body>
  <?php
      $myvar = 'Hello world';
      echo $myvar;
  ?>
</body>
</html>
```

2. Modify ex3-1.php so that the code looks as follows and save it as **ex3-2.php**

```
$myName='Mary Smith';
echo 'My name is '.$myName;  /* the '.' symbol adds text together, we will
cover this later */
```

3. Modify ex3-2.php so that the code looks as follows and save it as **ex3-3.php**

    ```
    $myName=’Mary Smith’;
    echo “My name is $myName”;
    ```

    Note – try `echo ‘My name is $myName’;`
    What happens when single quotes are used?

4. Create a new file with the following code and save it as **ex3-4.php**

```
<?php
  // Create the variables
  $author = 'Kevin Yank';
  $book = 'Getting Started with PHP';

  // Print the values
  ?>
  <p>
    <?= 'I\’ve got the book '.$book.' written by '.$author; ?>
    <br />
    <?= “I’ve got the book $book written by $author"; ?>
    <br />
    <?= "The book $book was written by $author"; ?>
  </p>
```

5. Design a Web page using similar code as above. Use variables to store your name, a variable to store the course you are doing and display the information using font comic sans ms, 50% bigger than the default and green. Save as **ex3-5.php**

By now you should have worked out the system of naming files in this subject, so please continue to save with this name methodology.

Arrays

An array is a multy-storied variable. Arrays are used when you wish to group many quantities under the one variable name, for instance a group of names. Each data is placed in distinct slots and these slots are numbered so that each piece of data has a unique address. In so doing, you automatically have indexed data (data with ***slot*** numbers), that is data with keys. The first key is 0. In PHP we call the slot number or index number the **key.**

For example, if an array called *$userNames[ ]* was to contain five names, you could visualize it as follows:

$userNames[ ]

index no. or slot no. or **key** ➔	0	1	2	3	4
data ➔	Mary	John	Sam	Jane	Tom

To create an array variable called **$userNames** and store values in each slot or level, you use the following syntax:

```
$userNames[0] = “Mary”;
$userNames[1] = “John”;
$userNames[2] = “Sam”;
```

and so on…..

You can also use the following syntax:

```
$userNames = array(“Mary”, “John”, “Sam”, “Jane”, “Tom”);
```

You can store number values in an array also. For example:

```
$age = array(‘20’, ‘25’, ‘30’, ‘40’, ‘50’);
```

You can output the stored values from an array as follows:

```
echo “The first name is $userNames[0]<br />”;    //outputs Mary
echo “The last name is $userNames[4]<br />”;     //outputs Tom
```

Exercise 3 cont.

6. Create a web page that will store the 7 days of a week in a variable called $day. This web page is to display the 7 days in a column.

7. Use file ex3-6.php and modify output so that the 5 week days are displayed. Save file as **ex3-7.php**

By default arrays start with index ‘0’. If you wanted to start with index ‘1’, you would use PHP => operator as follows:

```
$userNames = array(1 => “Mary”, “John”, “Sam”, “Jane”, “Tom”);
```

Now to output the first and last stored values from this array, code would be as follows:

```
echo “The first name is $userNames[1]<br />”;    //outputs Mary
echo “The last name is $userNames[5]<br />”;     //outputs Tom
```

Exercise 3 cont.

8. Create a web page that will store your favorite 4 actors in an array called $actors. The array is to start with key no. ‘1’. Display your favorite 4 actors in a column.

Associative Array or Key-Value Pair

Instead of using numbers for keys (or index) you may wish to use text that has a meaning. For example:

```
$profit[‘Jan’] = 580;
$profit[‘Feb’] = 790;
$profit[‘Mar’] = 820;
Or shorter code would be
$profit = array(‘Jan’ => 580, ‘Feb’ => 790, ‘Mar’ => 820);
```

Then

```
echo “The profit for January was $profit[‘Jan’]”;
//this will display  - The profit for January was 580
```

Exercise 3 cont.

9. Create a web page that will store three names and their ages in an associative array. It will then display the three ages in a list (using <ol> and <li> tags).

Operators

You can apply mathematical operators to numbers. For example:

```
$total = 14 + 1;    // $total will store 15
```

You can also add variables together. For example

```
$num1 = 5;
$num2 = 7.6;
$total = $num1 + $num2;    //$total will store 12.6
```

OPERATORS

Mathematical

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus

Relational

>	Greater than
>=	Greater than or equal to
==	Equal (used to check if 2 variables are equal)
<	Less than
<=	Less than or equal to
!=	Not equal to
<>	Not equal to

Note: One "=" means assignment.

Logical

and	And	$a and $b	True if $a and $b are true (same as "&&")
or	Or	$a or $b	True if $a or $b are true (same as "||")
!	Not	!$a	True if $a is false
<>	Not	$a<>$b	True if $a is not equal to $b
&&	And	$a && $b	True if $a and $b are true (same as "and")
||	Or	$a || $b	True if $a or $b are true (same as "or")
xor	Exclusive or	$a xor $b	True if $a or $b are true but not both.

Increments & decrements

$x++	Return $x then increment $x by 1
++$x	Increment $x by 1 then return $x
$x--	Return $x then decrement $x by 1
--$x	Decrement $x by 1 then return $x

Note

A very common procedure used in programming (that does not have mathematical sense) is:

example

```
$cost = 50
$cost = $cost + 1
```

this statement says: new $cost = current $cost + 1 so $cost=51 now

there is a short cut for this expression

```
$cost += 1
```

So an expression like `$age = $age +10` could be written as `$age +=10`

Exercise 4

1. Create code and use the above example. Display results so you can see that `$cost = $cost + 1` is the same as `$cost += 1`

2. Create a web page containing the PHP code to store two numeric variables, add the two variables together and display the result.

3. Create a web page containing the PHP code to store two numeric variables, multiply the two variables together and display the result.

String Operator

A string is a series of characters composed of text or numbers or symbols or a combination of all. You use quotes around a string. For example:

```
$userName = "Mary Smith";
$address = '50 Happy Street';
```

You can add or concatenate strings together using the ‘.’ dot symbol. For example:

```
$firstName = “Tom”;
$surname = “Cruise”;
$userName = $firstName . $surname;    // $userName will store Tom Cruise
```

You can **concatenate different objects** together.

```
echo ‘The user name is: ‘ . $userName . ‘<br />’;
```

Here you are concatenating a literal string – “The username is:”
a variable – $userName
and a tag – <br />

<u>**Exercise 4 cont.**</u>

4. Use ex3-4.php and modify the code so that:

```
$firstName = ‘Kevin’;
$lastName = ‘Yank’;
$author = ………    //concatenate the previous 2 variables to get one full name
```

   Then display to screen the author’s full name and title of book.

5. Use file ex4-2.php and modify so that the output looks like (for example): 4 + 5 = 9. Save the file as **ex4-5.php**

6. Use file ex4-3.php and modify so that the output looks like (for example): 4 * 6 = 24. Save the file as **ex4-6.php**