

Introduction to PHP

9

By the end of this section of the practical, the students should be able to:

- ✚ Install PHP and get it works with Apache Web server
- ✚ Assign variables within PHP scripts and display the output in the browser
- ✚ Work with PHP's built-in variables, data types and operators
- ✚ Construct PHP programs that process form data
- ✚ Use string function and regular expression for server-side validation.
- ✚ Tackle PHP's error messages and warnings

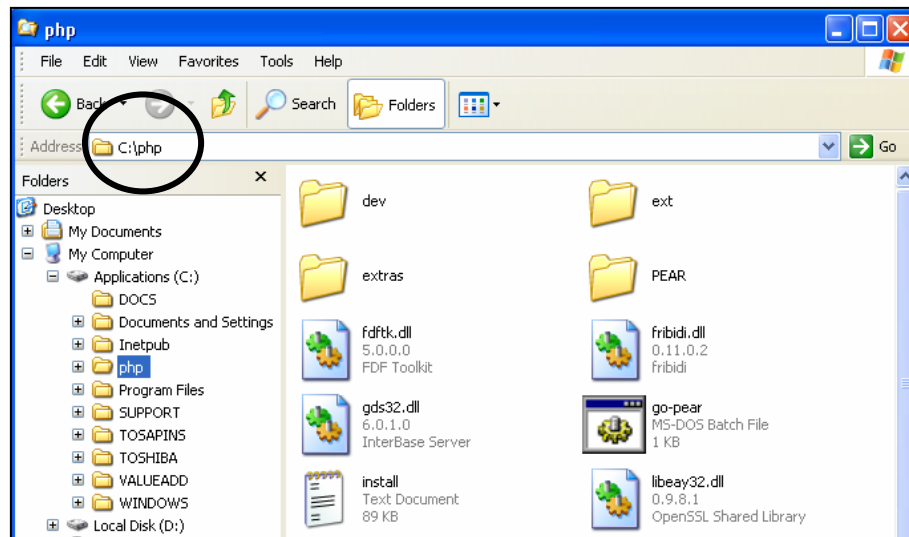
1 Introduction

PHP, an open-source product is among the most stable and efficient tools available for creating dynamic, interactive web pages. PHP is available for multiple operating systems and Web servers. When a browser requests an HTML document that includes PHP script, the Web server that provides the document calls the PHP processor. The server determines that a document includes PHP script by the filename extension. For example, if it is *.php, *.php3 or *.phtml, it has embedded the PHP. This chapter provides the students an overview of absolute basics for PHP server-side scripting. Within these practical exercises, the student will find out how to create dynamic Web page using PHP scripting, how to display PHP output and working with PHP's built in variables, data types and operators. Finally, this chapter also include example on how to match and manipulate strings within a target by using regular expressions.

1.1 Hands-on Exercises

1.1.1 Installing PHP on Windows

1. Download the '**php-5.1.4-Win32**' zip package at <http://www.php.net/downloads.php>. Save the installation file into your computer.
2. Rename the zip file as '**php**' and then extract the content to **c:\php**. The following figure shows directory of your php.



3. Now we are ready to install PHP for Windows. Make sure you stop Apache before you start the PHP installation process.
4. Use a text editor such as Notepad or Editplus to open '**php.ini-dist**'. You will perform following changes to the configuration file and save it as '**php.ini**'.
 - o To run scripts written for versions of PHP earlier than 4.1, search for 'register_globals' in '**php.ini**' and change it to the following:
`register_globals = on`

- To specify the directory in which the loadable extensions (modules) reside, set the following :

```
extension_dir = "c:\php"
```

5. Now you will configure Apache to work with PHP. To do this, navigate to the '*C:\Program Files\Apache Group\Apache2\conf*' and open the '**httpd.conf**' file in a text editor. Alternatively you can choose **Start > Apache HTTP Server > Configure Apache Server > Edit the Apache httpd.conf** to access the Apache's configuration file.

- i. Search for the section of the file that has a series of commented out '`LoadModule`' statements. Statements prefixed by the hash "`#`" sign are regarded as having been commented out. Add the following line after all the '`LoadModule`' statements:

```
LoadModule php4_module "c:/php/php5apache2.dll"
```

- ii. The following lines are added to specify how Apache should handle the various file extensions. In this case, we specify that Apache treats any file containing *.php, *.php3 and *.phtml as a PHP-interpreted file. Search for '`AddType`' in the file, and add the following line after the last '`AddType`' statement:

```
AddType application/x-httpd-php .php
```

```
AddType application/x-httpd-php .php3
```


```
AddType application/x-httpd-php .phtml
```

6. Copy '**php4ts.dll**' from **c:\php** folder into **c:\windows\system32**.
7. Save the '**httpd.conf**' file. Restart the Apache service again.

1.1.2 Testing the Installation of PHP


1. Create a sub folder in the 'C:\Program Files\Apache Group\Apache2\htdocs'. This folder will be your **work folder** to store all the files you create during these practical exercises.
2. Open a text editor and enter the following code:

```
<?php  
phpinfo();  
?>
```
3. Save the php file as '**test.php**' in your work folder (C:\Program Files\Apache Group\Apache2\htdocs\yourfolder).
4. Open browser with URL (*http://localhost:8080/your_folder_name/test.php*) to test your PHP installation.
5. The following page shows the PHP configuration with Apache is successful.

PHP Version 4.3.10

System	Windows NT ANUSURIYA 5.1 build 2600
Build Date	Dec 14 2004 17:46:48
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
PHP API	20020918
PHP Extension	20020429
Zend Extension	20021010
Debug Build	no
Thread Safety	enabled
Registered PHP Streams	php, http, ftp, compress.zlib

This program makes use of the Zend Scripting Language Engine:
Zend Engine v1.3.0, Copyright (c) 1998-2004 Zend Technologies

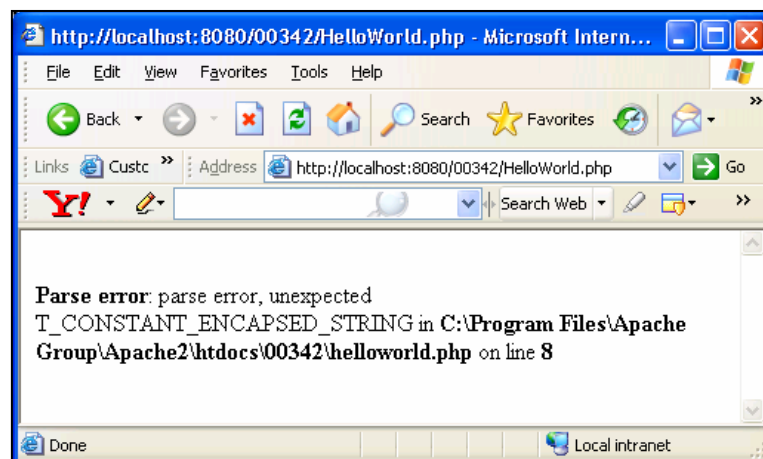
Powered by


1.1.3 Embedding PHP in HTML

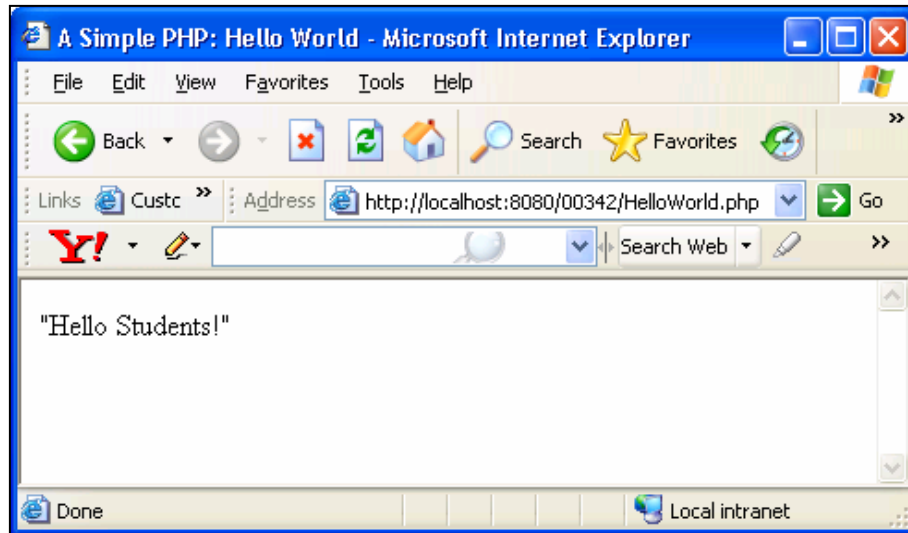
1. In this step you will write a simple PHP code that will displays a simple text output. Open your text editor and enter the following code. In this exercise, PHP code embedded within HTML tags, yet the file is called a PHP file and has a ***.php** file extension. Save your document as **'helloworld.php'** in your work folder.

```
1 <html>
2 <head>
3     <title>A Simple PHP: Hello World</title>
4 </head>
5 <body>
6 <?
7     # Using comments in code..
8     echho "\"Hello Students!\""
9 ?>
10 </body>
11 </html>
12
```

2. Start your browser and type *http://localhost:8080/your_folder_name/helloworld.php* as the URL. The page will display parsing error message.
3. For fatal errors like misspelled keywords, function names, or symbols, PHP will terminate the program and output an error message returned as an HTML document. In this exercise, we purposely misspelled 'echo'. Note that the name of the file and the line number are displayed in the figure below.



4. Now, change the 'echho' to 'echo' to correct the error in the PHP script. Save the PHP script and open it again in the browser. The page appears as shown in next figure.



5. PHP code is identified to the Apache server with the special HTML `<?php ?>` tag combination. These tags are needed to tell the web server to process PHP code in the file.
6. In **Line 7**, to insert the comment in a single line of PHP code, you preface the comment with either symbol `#`. Apart from this symbol, you can use the two forward slashes `//` or `/*.....*/` tags to enclose the multiple lines of code.
7. If you want to send PHP reserved characters such as double quotations to the web browser within the `echo` command, you must use the backslash character as shown in **Line 8**.

8. PHP display the text in your browser using the `echo` function. Each code line in PHP must end with a semicolon `;`.



The common two ways to create output in the browser: `echo` and `print`.

- *The `echo` function can be called with or without parenthesis around its parameters. If parentheses are included, only a single string parameter is acceptable; otherwise any number of parameters can be included. For example,*
`echo "Roses
" , "Carnations
" ;`
- *The `print` function is like `echo`, except it can never be called with more than one parameter, and it returns a value. The parentheses are optional. The returned value indicates whether the operation was completed (1 if it is succeeded and 0 if it is failed)*

1.1.4 Including Files

1. We probably want our pages that more than a single page has a common look and feel. PHP can make this possible with provide a method called **included files**.

2. Open a new document in your text editor and type the following code:

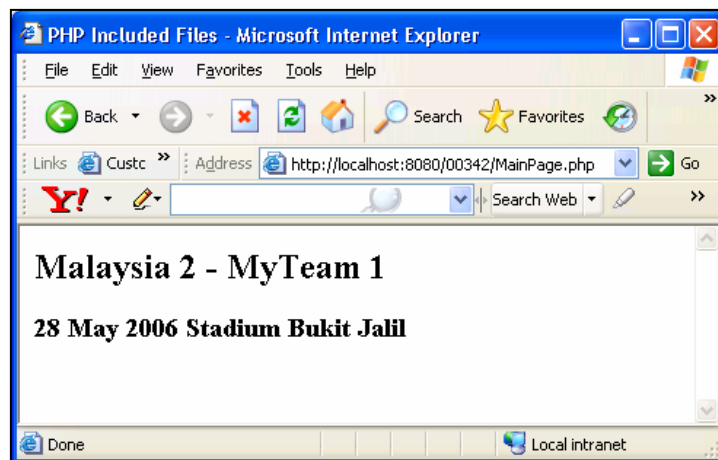
```
<html>
<head>
<title>PHP Included Files</title>
</head>
<body>
<?
    /* To concatenate more than one variable together, use the
    dot (.) operator */
    echo "<h2>Malaysia 2 - " . " MyTeam 1 <br>";
    echo "<h3>28 May 2006 Stadium Bukit Jalil";
?>
```

3. Save the file as **header.inc** in your work folder and then close the document.

4. Open a new document in your text editor and enter the following code:

```
<? include("header.inc"); ?>
<p> This is the Main Page </p>
</body>
</html>
```

5. Save the file as '**Mainpage.php**' in your work folder. The *header.inc* that we created in the Step 2 in this exercise is a simple HTML code that includes PHP output statements. The PHP code includes headers signified by <h2> and <h3> tag combination and called in the '**Mainpage.php**' and included in the PHP file by the Apache web server before being served to the browser.
6. Start your browser and type *http://localhost:8080/your_folder_name/Mainpage.php* as the URL. The main page appears as shown below :



1.1.5 Accessing Form Variables

1. Depending on your PHP version and setup, you can access the form data via variable in three ways. Let's say you are going to access the contents of the text field 'Name', you may use one of the following way:

Form	Description
<code>\$_POST['Name']</code> or <code>\$_GET['Name']</code>	Supported by version 4.1 and later
<code>\$HTTP_POST_VARS['Name']</code> or <code>\$HTTP_POST_VARS['Name']</code>	Supported by version 4.0 and later
<code>\$Name</code>	To enable this kind of access, you need to turn on the 'register_globals' in your PHP configuration file. In PHP 4.2 and later this parameter is set to Off by default for security reasons.

- In this exercise, you will use the implicit array for form values, `$_POST['...']` for referencing form variables.
- The following is a basic HTML form that uses the `post` method to redirect the form to another PHP page that displays the entered information in table format. Open a new document in your text editor and enter the following code:

```
<html>
<head>
<title>Sample form to take user input</title>
</head>
<body>
<h3>Please fill in all fields and click Submit Form. </h3>
<!-- Post form data to Display.php -->
<form method = "post" action = "Display.php">
<hr />
<!-- create four text boxes for user input -->
<table>
  <tr> <td>First Name</td>
  <td><input type="text" name="fname" /></td>
</tr>
  <tr> <td>Last Name</td>
  <td> <input type = "text" name = "lname" /></td>
</tr>
  <tr><td>Address</td>
```

```

        <td><textarea name = "address" rows=5
        cols=25></textarea></td>
    </tr>
    <tr><td>City</td>
    <td><input type = "text" name = "city" /></td>
    </tr>
    <tr> <td>Phone Number</td>
    <td><input type = "text" name = "phone" /></td>
    </tr>
    <tr>
    <br />
    <!-- create a submit button -->
    <td><input type="submit" value= "Submit Form"/>      </td>
    <td><input type="reset" value="Clear Form"/> </td>
    </tr>
</table>
</form>
</body>
</html>

```

4. Save the file as '**UserForm.html**' in your work folder.
5. Now you will create a PHP file to retrieve the values inserted by user in the above form. Open a new document and type the following code:

```

<html>
<head> <title>Displaying the Information</title>
</head>
<body>
<?
// Access HTTP POST Data and assign them to local variables
$fname_n = $_POST['fname'];
$lname_n = $_POST['lname'];
$address_n = $_POST['address'];

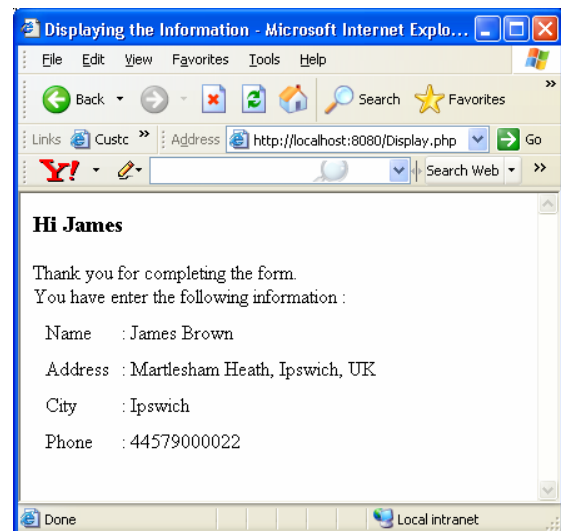
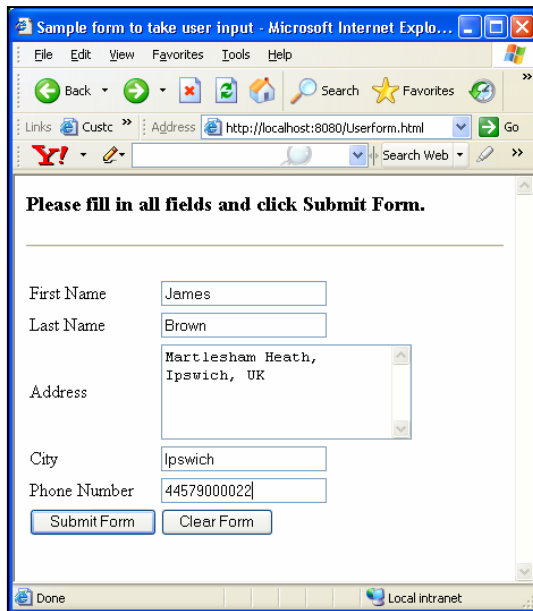
```

```

$city_n = $_POST['city'];
$phone_n = $_POST['phone'];
?>
<h3>Hi <?php echo $fname_n; ?></h3>
Thank you for completing the form.<br />
You have enter the following information :
<table border="0" cellpadding="0" cellspacing="10" >
    <tr>
        <td>Name</td>
        <td> : <? echo $fname_n, " " , $lname_n ?></td>
    </tr>
    <tr>
        <td>Address</td>
        <td> : <? echo $address_n ?></td>
    </tr>
    <tr>
        <td>City</td>
        <td> : <? echo $city_n ?></td>
    </tr>
    <tr>
        <td>Phone</td>
        <td> : <? echo $phone_n ?></td>
    </tr>
</table>
</body>
</html>

```

6. Save the file as **Display.php** in your work folder. In the browser, type *http://localhost:8080/your_folder_name/UserForm.html* as the URL. And then enter information on the Web and then click the **Submit** button. The entry form and the results page should resemble figures below.



*There are two methods to specify how the form's data is sent to the Web server which are **POST** and **GET**. The `method='post'` appends form data to the browser request, which contains the protocol **HTTP** and the requested resource's **URL**. The other possible value, `method='get'` appends the form data directly to the end of the **URL**.*

- Let's say the user wants to switch from using **HTTP post** to using **HTTP get**. Make relevant changes to your '**UserForm.html**' and '**Display.php**' documents. Test the pages in the browser. The pages should look like the above figures.

1.1.6 Using String Function and Regular Expression in Server-side Validation

- On the client side, validation is performed using scripting such as JavaScript. This can be very tricky, because some users might turn off JavaScript support in their browsers before they use your application. In this case, client side validation will not help you much if you try to verify data from a form because your JavaScript code will not be executed or interpreted by the browser.

2. To overcome such problem, in this exercise we will write a PHP script to perform server-side validation. The exercise will use a combination of string functions and regular expressions to validate common data submitted via a registration form. **‘Regular Expression’** is a function consisting of an expression used for complex string manipulation in PHP.
3. Open a new document in your text editor and then type the following code:

```
<html>
<body>
<h3>User Registration Form</h3>
<?

# Only validate form when form is submitted
if(isset($_POST['submit']))
{
    # Get form input
    $username_input = $_POST['username'];
    $password_input = $_POST['password'];
    $password2_input = $_POST['password2'];
    $email_input = $_POST['email'];

    # Declare a variable to hold error message
    $error_msg='';

    # Function to check for email address
    function checkEmail($mail) {
        # check if email is a valid address
        $pattern = "/^[A-z0-9\._-]+". "@". "[A-z0-9][A-z0-9-
        ]*[". "(\\.[A-z0-9_-]+)*". "\\.(\\[A-z]{2,6})$/" ;
        return preg_match ($pattern, $mail);
    }

    # Function to check String for Length
```

```

function checkLength($string, $min, $max) {
    $length = strlen ($string);
    if (($length < $min) || ($length > $max)) {
        return FALSE;
    } else {
        return TRUE;
    }
}

# Password Checking
function checkPass($pass1, $pass2) {
    if ($pass1 == $pass2) {
        return TRUE;
    } else {
        return FALSE;
    }
}

# Call related function
if(!checkLength($username_input,5,10)){
$error_msg.="Please enter a username between 5 to 10
characters long<br>";
}
if(!checkLength($password_input,1,8)){
$error_msg.="Please enter a password maximum 8 characters
long<br>";
}
if(!checkPass($password_input,$password2_input)){
$error_msg.="Both passwords are not matched<br>";
}
if(!checkEmail($email_input)){
$error_msg.="Please enter a valid email address<br>";
}

```

```

# Display error message if any, if not, proceed to other
processing
if($error_msg==""){
echo "<h3><font color=blue>The data you entered were correct.
Thank You!<br>";
}
else {
echo "The following errors are generated : <br>";
echo "<font color=red>$error_msg</font>";
}
}
?>

<form method="POST" action="Validate.php">
<table border="1" cellpadding="5" cellspacing="0" style="border-
collapse: collapse" width="60%">
  <tr>
    <td width="10%" align="left">Username</td>
    <td width="84%">
      <input type="text" name="username" size="20" value="<? echo
      $username_input; ?>"> (between 5 to 10 characters)</td>
  </tr>
  <tr>
    <td width="10%" align="left">Password</td>
    <td width="84%">
      <input type="password" name="password" size="20" value="<?
      echo $password_input; ?>">(must be maximum 8
      characters)</td>
  </tr>
  <tr>
    <td width="10%" align="left">Confirm Password</td>
    <td width="84%">
      <input type="password" name="password2" size="20" value="<?
      echo $password2_input; ?>">(must be same as initial
      password)</td>
  </tr>

```

```

</tr>
<tr>
  <td width="10%" align="left">Email</td>
  <td width="84%">
    <input type="text" name="email" size="20" value="<? Echo
    $email_input; ?>"></td>
</tr>
<tr>
  <td width="10%" align="left"> </td>
  <td width="84%">
    <input type="submit" value="Register" name="submit"></td>
</tr>
</table>
</form>
</body>
</html>

```

4. Save your file as **'Validate.php'** in your work folder. Start your browser, type http://localhost:8080/your_folder_name/Validate.php.
5. The form and validation page resembles figures below respectively.

User Registration Form

Registration form with user's details

Username	<input type="text" value="khchong1234"/>	(between 5 to 10 characters)
Password	<input type="password" value="..."/>	(must be maximum 8 characters)
Confirm Password	<input type="password" value="...."/>	(must be same as initial password)
Email	<input type="text" value="chong.com"/>	
<input type="button" value="Register"/>		

User Registration Form

The following errors are generated :

Please enter a username between 5 to 10 characters long

Both passwords are not matched

Please enter a valid email address

Errors generated by PHP

Username	<input type="text" value="khchong1234"/>	(between 5 to 10 characters)
Password	<input type="password" value="..."/>	(must be maximum 8 characters)
Confirm Password	<input type="password" value="...."/>	(must be same as initial password)
Email	<input type="text" value="chong.com"/>	
<input type="button" value="Register"/>		

User Registration Form

The data you entered were correct. Thank You!

Data validation is successful

Username	<input type="text" value="khchong"/>	(between 5 to 10 characters)
Password	<input type="password" value="..."/>	(must be maximum 8 characters)
Confirm Password	<input type="password" value="..."/>	(must be same as initial password)
Email	<input type="text" value="chong@yahoo.com"/>	
<input type="button" value="Register"/>		

6. The following table explains the three custom functions included in the validation file for error checking.

Function	Description
<pre># Function to check String for Length function checkLength(\$string, \$min, \$max) { \$length = strlen (\$string); if ((\$length < \$min) (\$length > \$max)) { return FALSE; } else { return TRUE; } }</pre>	<p>This function will check the length of a string and return FALSE if the length is not within a range we specify. For this task, the <code>strlen()</code> function is used to determine the length of the string and then compare it to the specified minimum and maximum length.</p>
<pre># Password checking function checkPass(\$pass1, \$pass2) { if (\$pass1 == \$pass2) { return TRUE; } else { return FALSE; } }</pre>	<p>This function will identify whether the two passwords are matched and return FALSE if they are not matched.</p>
<pre># Function to check for email address function checkEmail(\$mail) { # check if email is a valid address \$pattern = "/^[A-z0-9\._-]+" . "@" . "[A-z0-9][A-z0-9-]*" . "(\.[A-z0-9_-]+)*" . "\.([A-z]{2,6})\$/"; return preg_match (\$pattern, \$mail); }</pre>	<p>This function tries to find the match for the regular expression pattern in the supplied email string. The regular expression output is assigned as return value of the function. The function invokes the <code>'preg_match'</code> function in order to match the objects in the target file with the desired validation parameters.</p> <p>The slashes <code>/</code> are delimiters, <code>^</code> marks the start of string or line and the Dollar sign <code>\$</code> is the end of the string, or line. The plus-symbol <code>+</code> means required. Then, <code>[A-z0-9\._-]</code> is any character A-Z, a-z, 0-9 and <code>_</code> or <code>-</code>.</p>

1.2 Self-Review Questions

1. Which of the following is **NOT** a valid variable name in PHP?
 - A. `$number-in-class`
 - B. `$number_in_class`
 - C. `$nic`
 - D. `$NumberInClass`

2. In PHP, the word 'AND' is used as
 - A. a regular variable
 - B. a function
 - C. it is reserved, but unused
 - D. an operator

3. When would you use double quotes rather than single quotes to define a string of text?
 - A. When you wanted to include variables and special sequences such as `\n` within the string
 - B. When you wanted the string to be more than one character long
 - C. When you wanted every character to be taken as a literal
 - D. When you wanted to include special characters such as `@` and `%` within the string

4. PHP server scripts are surrounded by which delimiters?
 - A. `<script>...</script>`
 - B. `<?php...?>`
 - C. `<>...</>`
 - D. `<%...%?>`

5. How do you get information from a form that is submitted using the 'get' method?
- A. `Request.QueryString;`
 - B. `$_GET[];`
 - C. `Request.Form;`
 - D. `Request.Get[];`
6. What is the correct way to include the file 'external.inc' ?
- A. `<? php include ('external.inc'); ?>`
 - B. `<? php require ('external.inc'); ?>`
 - C. `<!--include file ('external.inc') -->`
 - D. `<% include file= ('external.inc') %>`
7. Order the following statements to describe the flow of the following regular expression pattern to verify email address:

```
preg_match('/^[A-z0-9_-]+[@][A-z0-9_-]+([.][A-z0-9_-]+)+[A-z]{2,4}$/', $email)
```

1	Begin with a delimiter /
	Adding a () around the next subset at the additional [.] tells it to look for more text following the. (.com, .net, .etc)
	Then, just add a '@' after the plus to look for the @ symbol in the e-mail address
	Now all you need is to repeat your previous criteria for matching (text between A and Z or the numbers 0 - 9)
	Finally, the \$ indicates the end of the target string
	And the expression is closed with our ending delimiter /.
	Then adding a minimum/maximum bracket {2, 4} tells it to look for an ending that falls within those values (for example, .my, .hk, etc)
	Then indicate the beginning of the line with ^
	Then, [A-Za-z0-9_-] is any character A-Z, a-z, 0-9 and _ or - .
	Then, indicate that this pattern is one or more with the + symbol.