

# CNT 4714: Enterprise Computing Summer 2014

## Introduction to PHP – Part 1

Instructor :      Dr. Mark Llewellyn  
                         markl@cs.ucf.edu  
                         HEC 236, 407-823-2790  
                         <http://www.cs.ucf.edu/courses/cnt4714/sum2014>

Department of Electrical Engineering and Computer Science  
Computer Science Division  
University of Central Florida

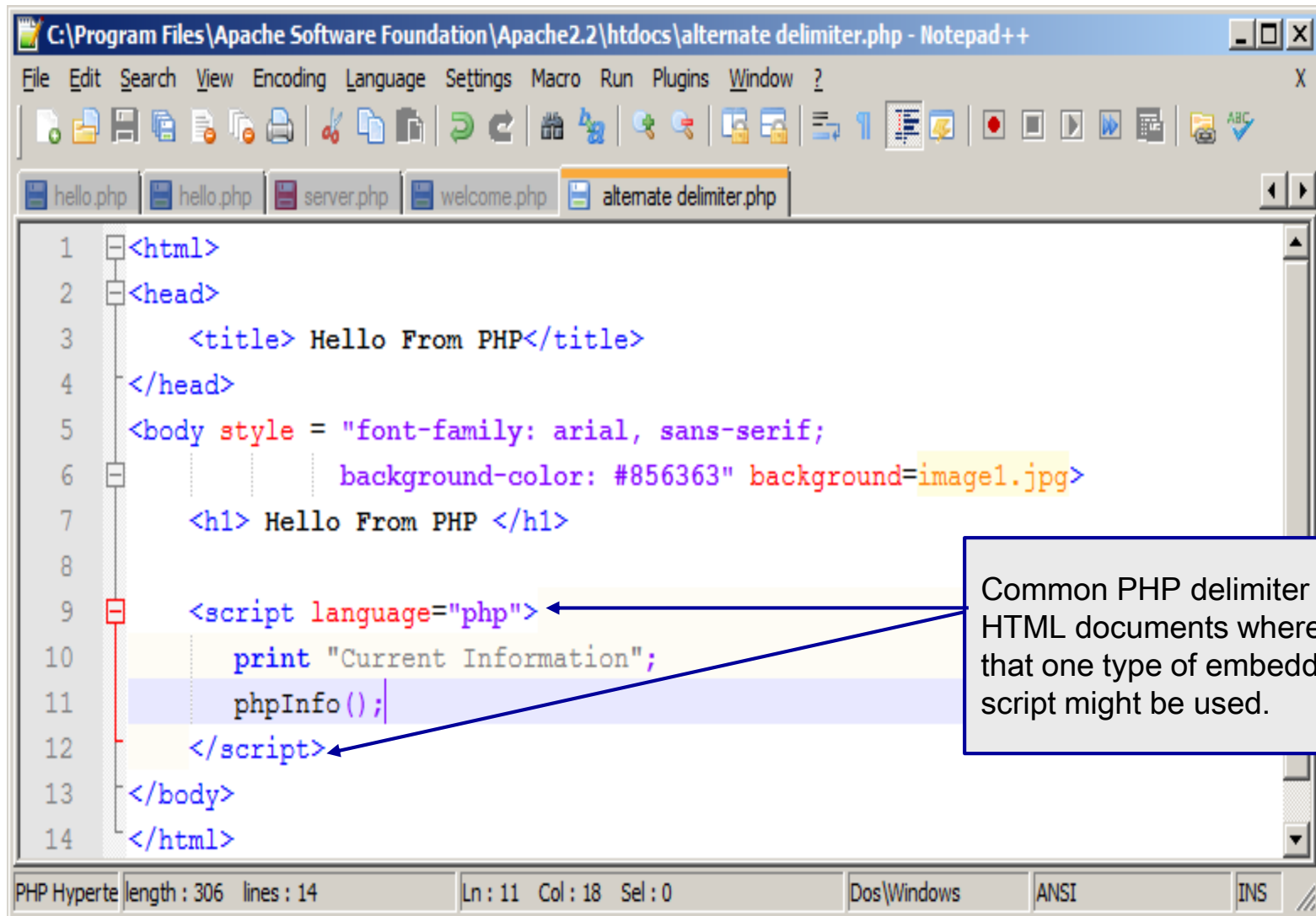


# Introduction to PHP

- We looked at a simple PHP example at the end of the set of notes that covered the installation of the Apache HTTP Server and PHP.
- PHP scripts can be created with any text editor, although Notepad++ is quite convenient for PHP scripting. I'll primarily use it in the examples.
- PHP script files should be saved with a `.php` extension.
- When PHP is embedded inside HTML documents, as it commonly is, several different delimiters can be used. These are illustrated on the next page.



# Introduction to PHP

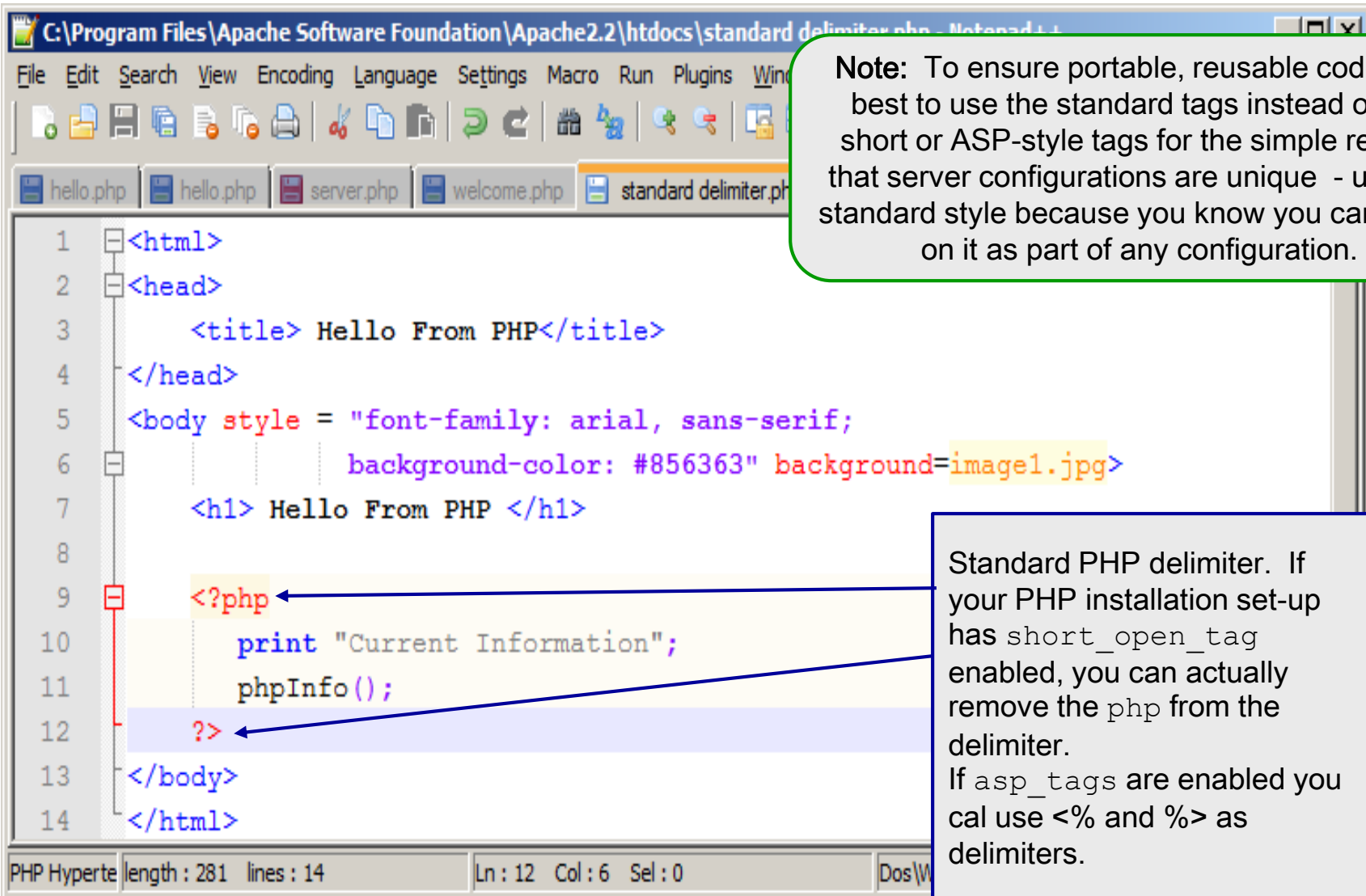


```
1 <html>
2 <head>
3     <title> Hello From PHP</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6     background-color: #856363" background=image1.jpg>
7     <h1> Hello From PHP </h1>
8
9     <script language="php">
10         print "Current Information";
11         phpInfo();
12     </script>
13 </body>
14 </html>
```

Common PHP delimiter in HTML documents where more than one type of embedded script might be used.



# Introduction to PHP



```
1 <html>
2 <head>
3     <title> Hello From PHP</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6     background-color: #856363" background=image1.jpg>
7     <h1> Hello From PHP </h1>
8
9     <?php
10         print "Current Information";
11         phpInfo ();
12     ?>
13 </body>
14 </html>
```

PHP Hypertext length : 281 lines : 14 Ln : 12 Col : 6 Sel : 0 Dos\W

**Note:** To ensure portable, reusable code, it is best to use the standard tags instead of the short or ASP-style tags for the simple reason that server configurations are unique - use the standard style because you know you can count on it as part of any configuration.

Standard PHP delimiter. If your PHP installation set-up has `short_open_tag` enabled, you can actually remove the `php` from the delimiter. If `asp_tags` are enabled you can use `<%` and `%>` as delimiters.



# Introduction to PHP

- As with any programming language, good practice in writing scripts would require comments to be included within the script.
- In-line comments in PHP are indicated with two forward slashes (//).
- Comments can appear any where in the script file and can appear in any position on any line.
- Multiple line comments are delimited with /\* and \*/
- Most PHP implementations also allow # to delimit in-line comments.



# Variables In PHP

- You can select just about any set of characters for a variable name in PHP, but they must:
  - Use a dollar sign (\$) as the first character.
  - Use a letter or an underscore character ( \_ ) as the second character.
- As with any programming/scripting language, good practice would suggest selecting variable names that help describe their function. For example `$counter` is more descriptive than `$c` or `$ctr`.
- You can use the `echo` statement or the `print()` function to output data in PHP. Which you use is more a matter of personal taste or style than anything else.



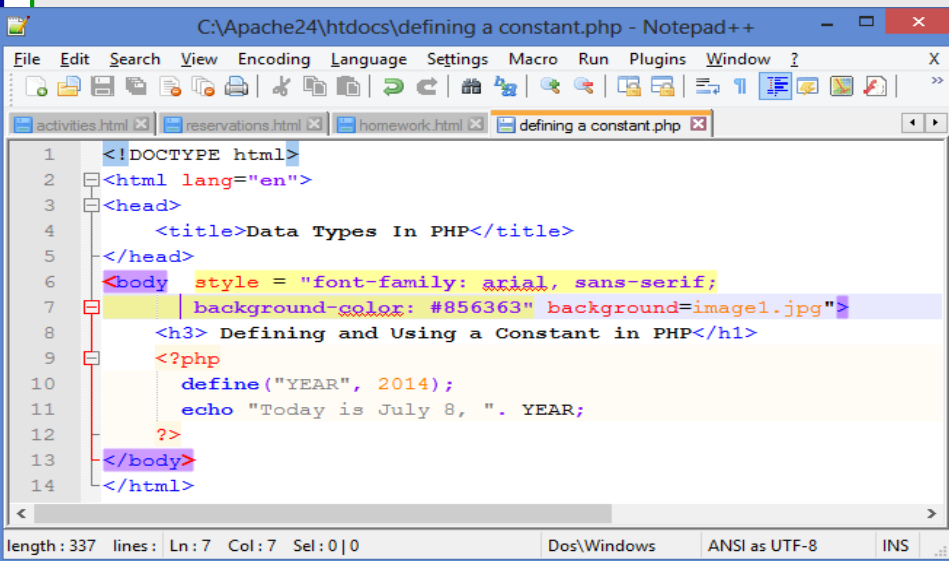
# Variables In PHP

- To print out the value of a variable \$x, write the following PHP statement:  

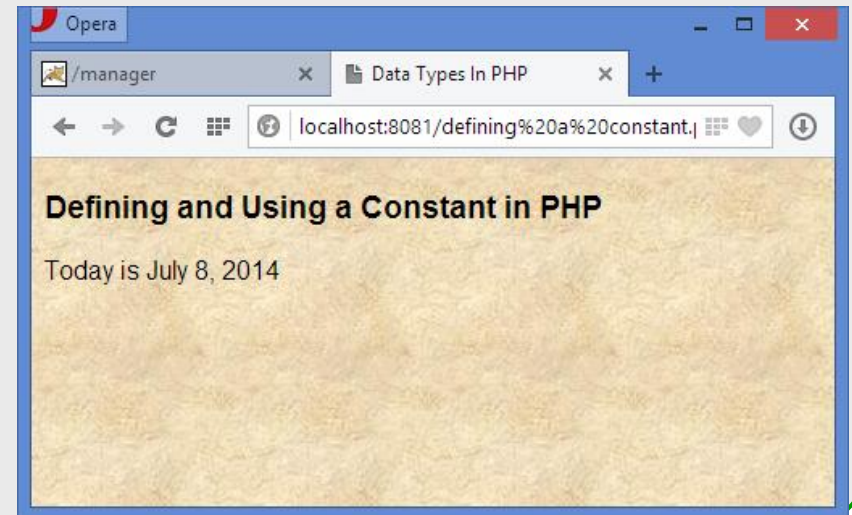
```
print ("$x");
```
- The following code will output “Candice is 26 years old”.  

```
$age=26;  
print ("Candice is $age years old.");
```
- The next page illustrates a full example using PHP variables.

Note: Constants are defined in PHP using the built-in `define()` function. As its name would imply a constant's value cannot be changed once it is set.

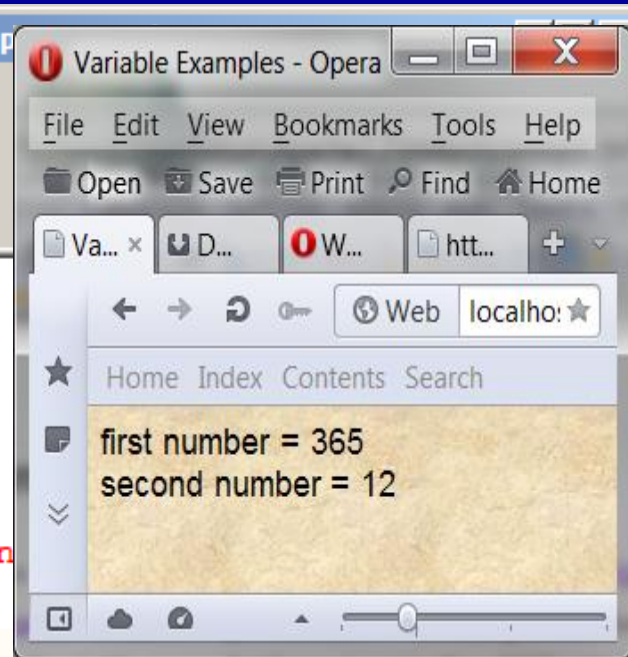


```
1 <!DOCTYPE html>  
2 <html lang="en">  
3 <head>  
4 <title>Data Types In PHP</title>  
5 </head>  
6 <body style = "font-family: arial, sans-serif;  
7 background-color: #856363" background=imagem1.jpg">  
8 <h3> Defining and Using a Constant in PHP</h1>  
9 <?php  
10 define("YEAR", 2014);  
11 echo "Today is July 8, ". YEAR;  
12 ?>  
13 </body>  
14 </html>
```





```
1 <html>
2 <head>
3     <title> Variable Example</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6     background-color: #856363" background
7
8 <?php
9     $firstNum = 12;
10    $secondNum = 365;
11    $temp = $firstNum;
12    $firstNum = $secondNum;
13    $seconNum = $temp;
14    print("first number = $firstNum <br /> second number = $seconNum");
15 ?>
16 </body>
17 </html>
```





# Data Types In PHP

- PHP is a dynamically typed language. This basically means that variables are not assigned a type when the variable is declared. Variable type is determined through assignment.
- The standard data types in PHP are shown in the table below:

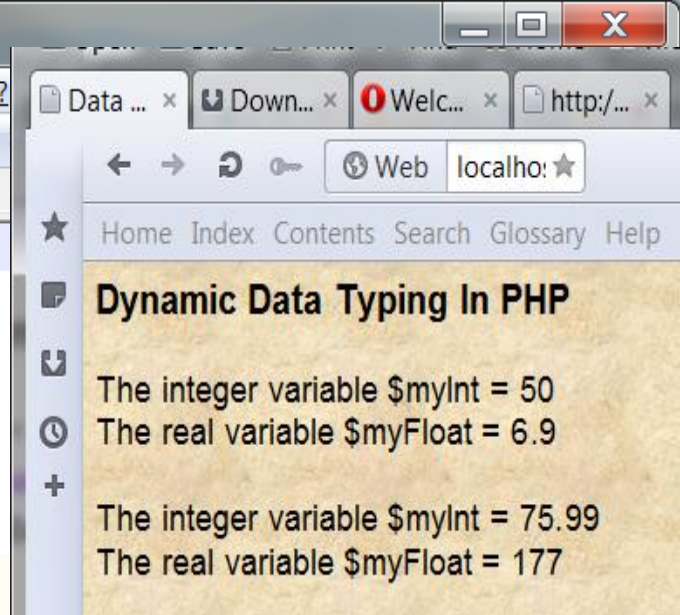
Data Type	Example	Description
Boolean	true	Either <code>true</code> or <code>false</code>
Integer	5	A whole number
Float or Double	3.14159	A floating-point number
String	"Hello"	A collection of characters
Object		An instance of a class
Array		An ordered set of keys and values
Resource		Reference to a 3 <sup>rd</sup> party resource (e.g. a database)
NULL		An uninitialized variable



Dynamic Data Typing Example

```
1 <html>
2 <head>
3 <title>Data Types In PHP</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6     background-color: #856363" background=image1.jpg>
7 <h3> Dynamic Data Typing In PHP</h3>
8 <?php
9     $myInt = 50;
10    $myFloat = 6.9;
11    echo 'The integer variable $myInt = ' . $myInt. "<br />";
12    echo 'The real variable $myFloat = ' . $myFloat. "<br />";
13    echo "<br />";
14    $myInt = 75.99;
15    $myFloat = 177;
16    echo 'The integer variable $myInt = ' . $myInt. "<br />";
17    echo 'The real variable $myFloat = ' . $myFloat. "<br />";
18 ?>
19
20 </body>
21 </html>
```

The concatenation operator in PHP is the period.



# Data Types In PHP

- Technically speaking, there are two types of strings in PHP: parsed and unparsed.
- **Parsed strings** are defined using double quotes and are parsed by PHP.
- **Unparsed strings** are defined using single quotes and are taken as is (they are not parsed).
- What's the difference? Within a parsed string, any references to variables within that string will be automatically replaced with their respective values, whereas within an unparsed string nothing is replaced.
- The example on the next page will clarify the differences.



```

1 <html>
2 <head>
3 <title>Data Types In PHP</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6     background-color: #856363" background=image1.jpg>
7 <h3> Parsed Versus Unparsed Strings In PHP</h3>
8 <?php
9     $myInt = 50;
10    $myString1 = "Hi there";
11    $myString2 = 'The value of $myInt = $myInt';
12    /* The next echo statement includes a reference to $myInt in a parsed string */
13    echo "The integer variable $myInt = ". $myInt. "<br />";
14    /* The next echo statement includes a reference to $myInt in an unparsed string */
15    echo 'The integer variable $myInt = '. $myInt. "<br />";
16    /* The next echo statement prints an unparsed string with no variable references */
17    echo $myString1. "<br />";
18    /* The next echo statement prints an unparsed string with a variable reference */
19    echo $myString2. "<br />";
20    /* The next echo statement contains both parsed and unparsed variable references */
21    echo "The integer variable ". '$myInt'. " = ". $myInt. "<br />";
22 ?>
23
24 </body>
25 </html>

```



# Parsed Versus Unparsed Strings In PHP

The integer variable 50 = 50  
 The integer variable \$myInt = 50  
 Hi there  
 The value of \$myInt = \$myInt  
 The integer variable \$myInt = 50

Parsed

Unparsed



# Arithmetic Operations In PHP

- PHP supports all normal arithmetic operators, with the normal semantic associated with each operator.
- PHP supports automatic increment and decrement operations in both prefix and postfix form, i.e., -- and ++.
- Using an unassigned variable in an expression does not generate an error, the value is simply assumed to be null.

Operator	Effect	Example	Result
+	Addition	<code>\$x = 2 + 2;</code>	<code>\$x</code> is assigned 4.
-	Subtraction	<code>\$y = 3;</code> <code>\$y = \$y - 1;</code>	<code>\$y</code> is assigned 2.
/	Division	<code>\$y = 14 / 2;</code>	<code>\$y</code> is assigned 7.
*	Multiplication	<code>\$z = 4;</code> <code>\$y = \$z * 4;</code>	<code>\$y</code> is assigned 16.
%	Remainder	<code>\$y = 14 % 3;</code>	<code>\$y</code> is assigned 2.

```
<?php
    $y = 3;
    $y = $y + $x + 1;
    print("x=$x y=$y");
?>
```

The output is: x=y=4



# String Variables In PHP

- PHP supports character string variables and this is a widely used aspect of PHP in handling form data.
- Be careful in PHP not to mix numeric and string types together in an expression.
- For example, you might expect the following statements to generate an error message, but they will not. Instead, they will output "y=1".

```
<?php
    $x = "banana";
    $sum = 1 + $x'
    print ("y=$sum");
?>
```



# String Variables In PHP

- The string concatenation operator in PHP is the period as shown below:

```
<?php
    $firstname = "Megan";
    $lastname = "Fox"
    $fullname = $firstname . $lastname;
    print("Full name = $fullname");
?>
```

The output of this script would be: Fullname=MeganFox

You can also use double quotation marks to create concatenation directly. Using the above example you could do the following: `$fullname2 = "$firstname $lastname";` This would have the same effect as: `$fullname2 = $firstname . $lastname;`





# String Variables In PHP

- PHP supports a large variety of string handling functions. A few of the more commonly used ones are illustrated on the next few pages.
- Most string functions require you to send them one or more arguments.
- Arguments are input values that functions use in the processing they do.
- Often functions return a value to the script based on the input arguments. For example:

```
$len = strlen($name);
```

Receives the number of characters in \$name

Name of function

Variable or value to work with



# String Variables In PHP

## strlen() function:

- This function returns the number of characters in the string argument to the function. Consider the following script:

```
<?php
    $comments = "Good Job";
    $len = strlen($comments);
    print ("Length=$len");
?>
```

This PHP script would output "Length=8" .



# String Variables In PHP

## trim() function:

- This function removes any blank characters from the beginning and end of a string. For example, consider the following script:

```
<?php
    $in_name = "    Megan    Fox    ";
    $name = trim($in_name);
    print ("name=$name$name");
?>
```

This PHP script would output "name=Megan FoxMegan Fox".



# String Variables In PHP

## strtolower() and strtoupper functions:

- These functions return the argument string in all uppercase or all lowercase letters, respectively. For example, consider the following script:

```
<?php
    $inquote = "Now Is The Time";
    $lower = strtolower($inquote);
    $upper = strtoupper($inquote);
    print("upper=$upper lower=$lower");
?>
```

This PHP script would output "upper=NOW IS THE TIME lower = now is the time"



# String Variables In PHP

## substr () function:

- This function enables a PHP script to extract a portion of the characters in a string variable. The general syntax is:

Assign the extracted substring into this variable.

```
$part = substr( $name, 0, 5);
```

Extract from this string variable.

Starting position to start extraction from.

Number of characters to extract. (If omitted it will continue to extract until the end of the string.)



# String Variables In PHP

## substr () function:

- The substr() function enumerates character positions starting with 0 (not 1),
  - For example, in the string “Homer”, the “H” would be position 0, the “o” would be position 1, the “m” position 2, and so on.
- For example, the following would output “Month=12 Day=25”.

```
<?php
    $date = "12/25/2002";
    $month = substr($date, 0, 2);
    $day = substr($date, 3, 2);
    print ("Month=$month Day=$day");
?>
```



# String Variables In PHP

## substr () function:

- This example does not include the third argument (and thus returns a substring from the starting position to the end of the search string).

```
<?php
    $date = "12/25/2010";
    $year = substr($date, 6);
    print ("Year=$year");
?>
```

- The above script segment would output "Year=2010".



# Controlling Script Flow In PHP

- PHP contains the normal control statements that handle decision making and iteration within a script.
- Normal logical operators are all supported with their standard semantics.
- As with many modern programming and scripting languages remember to use `==` in a logical comparison operation and not `=`. The single equal sign is an assignment operator and as such is always true. No syntax error is generated.
- The table on the following page illustrates the common logical operators in PHP.

**Note:** PHP also contains a `===` logical comparison operator (called the identical operator). This binary operator returns true iff its two operands are equal in value and also have the same type.





# Controlling Script Flow In PHP

Test Operator	Effect	Example	Result
==	Equal to	<pre>if (\$x == 6) {     \$x = \$y + 1;     \$y = \$x + 1; }</pre>	Run the second and third statements if the value of $\$x$ is equal to 6.
!=	Not equal to	<pre>if (\$x != \$y) {     \$x = 5 + 1; }</pre>	Run the second statement if the value of $\$x$ is not equal to the value of $\$y$ .
<	Less than	<pre>if (\$x &lt; 100) {     \$y = 5; }</pre>	Run the second statement if the value of $\$x$ is less than 100.
>	Greater than	<pre>if (\$x &gt; 51) {     print "OK"; }</pre>	Run the second statement if the value of $\$x$ is greater than 51.
>=	Greater than or equal to	<pre>if (16 &gt;= \$x) {     print "x=\$x"; }</pre>	Run the second statement if 16 is greater than or equal to the value of $\$x$ .
<=	Less than or equal to	<pre>if (\$x &lt;= \$y) {     print "y=\$y";     print "x=\$x"; }</pre>	Run the second and third statements if the value of $\$x$ is less than or equal to the value of $\$y$ .



# Controlling Script Flow In PHP

- PHP includes several different forms of logical control statements (decision statements).
- The `if` statement has the form:

```
if (expression) {  
    //code to execute if expression evaluates to true  
}
```

- The `if-else` statement has the form:

```
if (expression) {  
    //code to execute if expression evaluates to true  
} else {  
    //code to execute when expression evaluates to false  
}
```



# Controlling Script Flow In PHP

- There is also an `elseif` clause that can be used with `if` statements for a nested stack of `if` statements. The basic syntax for this clause is:

```
if (expression) {  
    //code to execute if expression evaluates to true  
} elseif (another expression) {  
    //code to execute when expression evaluates to false  
    //and another expression evaluates to true  
} else {  
    //code to execute if all expressions evaluate to false  
}
```



# Controlling Script Flow In PHP

- PHP includes a `switch` statement which allows for multiple options for a single evaluation of an expression. The basic syntax for the `switch` statement is:

```
switch (expression) {
    case result1:
        //code to execute if expression evaluates to result1
        break;
    case result2:
        //code to execute if expression evaluates to result2
        break;
    . . .
    default:
        //code to execute if no break has been encountered
}
```



# Controlling Script Flow In PHP

- The following example uses an input form (HTML) and two values are extracted from the form (`grade1` and `grade2`), passed to a PHP script which determines the average score, the maximum score and assigns a grade to the average for the student's scores.
- We'll get much more into forms and form handling in PHP later, but this simple example will illustrate several of the common threads that appear in form handling in PHP (and server side scripting in general).

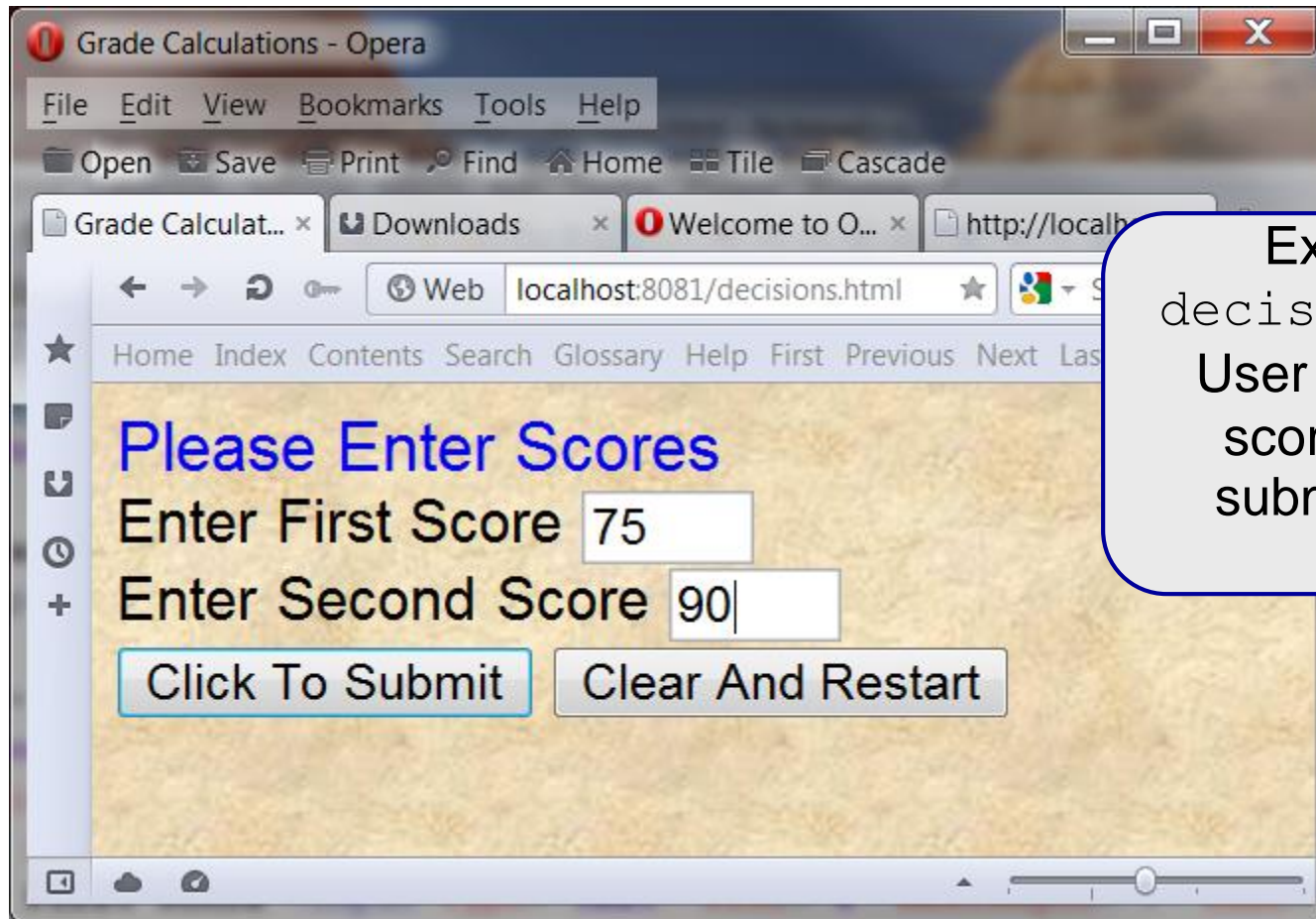


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>Grade Calculations</title>
5     <meta http-equiv="content-type" content="text/html" charset="iso-8859-1" />
6 </head>
7 <body style = "font-family: arial, sans-serif;
8     background-color: #856363" background=image1.jpg>
9 <form action="decisionsWithGlobals.php" method="post" >
10     <font size=4 color=blue>Please Enter Scores</font> <br />
11     Enter First Score <input type="text" size="4" maxlength="7" name="grade1" /> <br />
12     Enter Second Score <input type="text" size="4" maxlength="7" name="grade2" /> <br />
13     <input type="submit" value="Click To Submit" >
14     <input type="reset" value="Clear And Restart" >
15 </form>
16 </body>
17 </html>
```

decisions.html



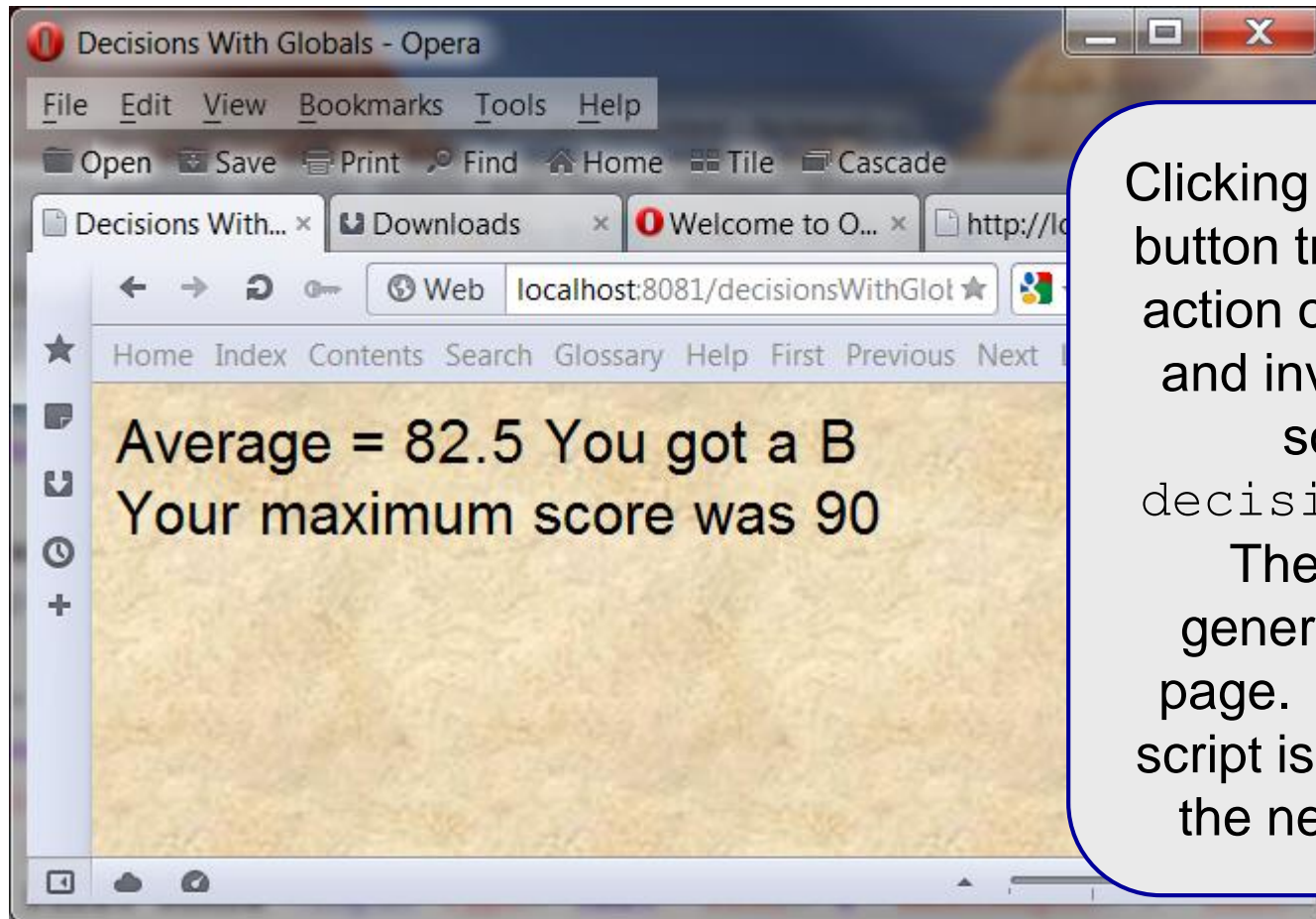
# Controlling Script Flow In PHP



Executing  
decisions.html  
User enters two  
scores, clicks  
submit button.



# Controlling Script Flow In PHP



Clicking the submit button triggers the action of the form and invokes the script

`decisions.php`

The script generates this page. The PHP script is shown on the next page.





```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>Decisions With Globals</title>
5     <meta http-equiv="content-type" content="text/html" charset="iso-8859-1" />
6 </head>
7 <body style = "font-family: arial, sans-serif;
8     background-color: #856363" background=image1.jpg>
9 <?php
10     $grade1 = $_POST["grade1"];
11     $grade2 = $_POST["grade2"];
12     $average = ($grade1 + $grade2)/2;
13     if ($average > 89){
14         print ("Average = $average You got an A");
15     } elseif ($average > 79) {
16         print ("Average = $average You got a B");
17     } elseif ($average > 69) {
18         print ("Average = $average You got a C");
19     } elseif ($average > 59) {
20         print ("Average = $average You got a D");
21     } elseif ($average >= 0) {
22         print ("Average = $average You got an F ");
23     } else {
24         print ("Illegal average less than 0: Average = $average");
25     }
26     $max=$grade1;
27     if ($grade1 < $grade2) {
28         $max = $grade2;
29     }
30     print ("<br /> Your maximum score was $max");
31 >>
32 </body>
```



# Controlling Script Flow In PHP

- PHP supports three types of iterative constructs:
  - the `while` loop (both top and bottom tested versions are supported)
  - the `for` loop
  - and the `foreach` loop.
- The `for` and `while` loops act as you would expect given your knowledge of other programming languages. The `foreach` loop applies specifically to arrays in PHP. We'll look at the `foreach` loop later.
- The next couple of pages show the basic syntax for each of the iterative constructs in PHP.



# Controlling Script Flow In PHP

- The syntax for the top tested version of the `while` loop is:

```
while (expression) {  
    //statements to execute  
}
```

- The syntax for the bottom tested version of the `while` loop is:

```
do {  
    //statements to execute  
} while (expression);
```



# Controlling Script Flow In PHP

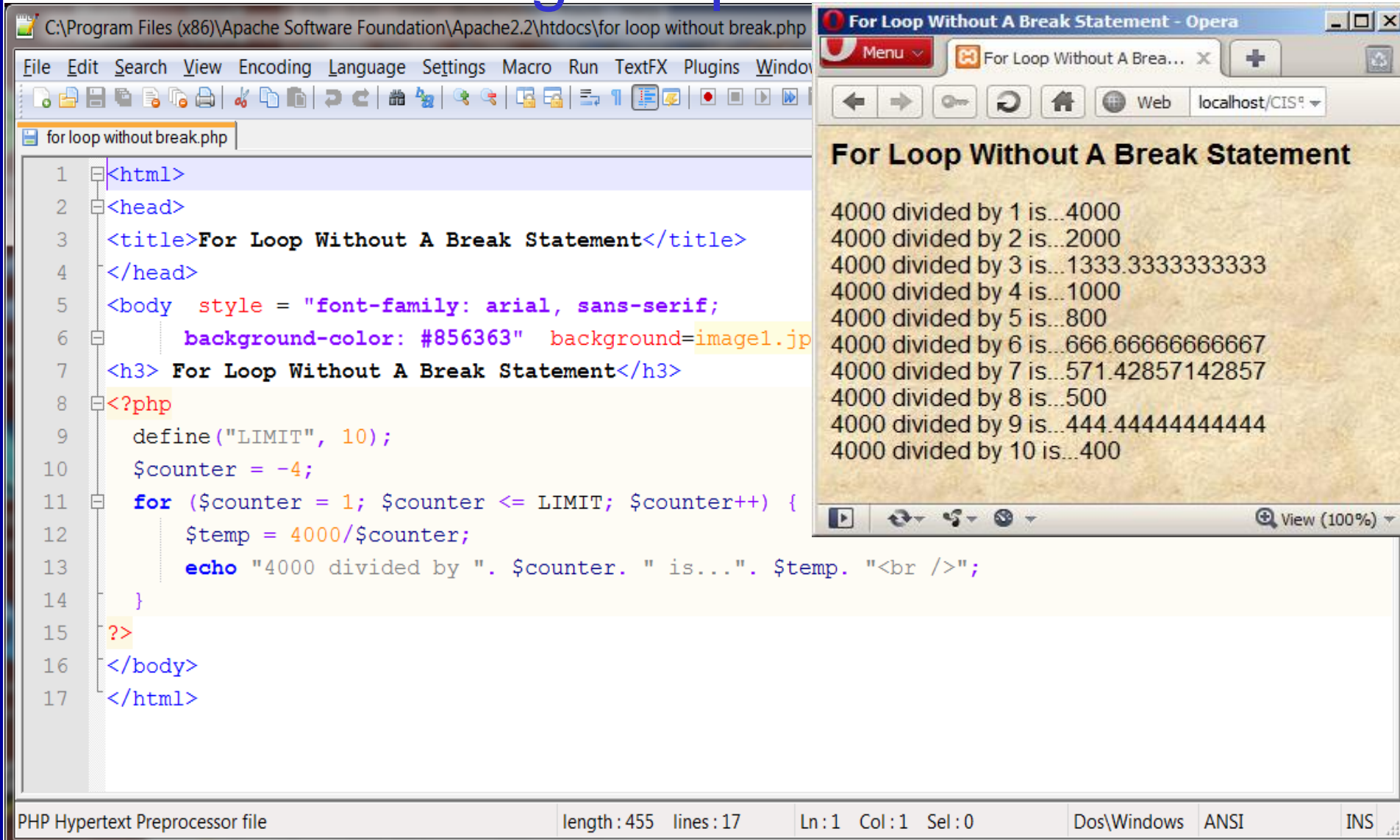
- The basic syntax for the for statement is:

```
for (initialization expr; test expr; modifying expr) {  
    //statements to be executed  
}
```

- The next couple of pages illustrates some of the nuances of dealing with counted loops in PHP.



# Controlling Script Flow In PHP



The image shows a code editor window on the left and a web browser window on the right. The code editor displays a PHP script that uses a for loop to calculate the division of 4000 by integers from 1 to 10. The browser window shows the rendered output of this script, displaying the results of the divisions on a light brown background.

```
1 <html>
2 <head>
3 <title>For Loop Without A Break Statement</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6 background-color: #856363" background=image1.jp
7 <h3> For Loop Without A Break Statement</h3>
8 <?php
9 define("LIMIT", 10);
10 $counter = -4;
11 for ($counter = 1; $counter <= LIMIT; $counter++) {
12     $temp = 4000/$counter;
13     echo "4000 divided by ". $counter. " is...". $temp. "<br />";
14 }
15 ?>
16 </body>
17 </html>
```

**For Loop Without A Break Statement**

4000 divided by 1 is...4000  
4000 divided by 2 is...2000  
4000 divided by 3 is...1333.3333333333  
4000 divided by 4 is...1000  
4000 divided by 5 is...800  
4000 divided by 6 is...666.666666666667  
4000 divided by 7 is...571.42857142857  
4000 divided by 8 is...500  
4000 divided by 9 is...444.444444444444  
4000 divided by 10 is...400

PHP Hypertext Preprocessor file      length : 455    lines : 17    Ln : 1    Col : 1    Sel : 0    Dos\Windows    ANSI    INS



```
1 <html>
2 <head>
3 <title>For Loop Without A Break Statement</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6     background-color: #856363" background=imager.jpg>
7 <h3> For Loop Without A Break Statement</h3>
8 <?php
9     define("LIMIT", 10);
10    $counter = -4;
11    for (; $counter <= LIMIT; $counter++) {
12        $temp = 4000/$counter;
13        echo "4000 divided by ". $counter. " is...". $temp. "<br />";
14    }
15 ?>
16 </body>
17 </html>
```

What do you expect will happen when \$counter == 0?



## For Loop Without A Break Statement

4000 divided by -4 is...-1000  
 4000 divided by -3 is...-1333.3333333333  
 4000 divided by -2 is...-2000  
 4000 divided by -1 is...-4000

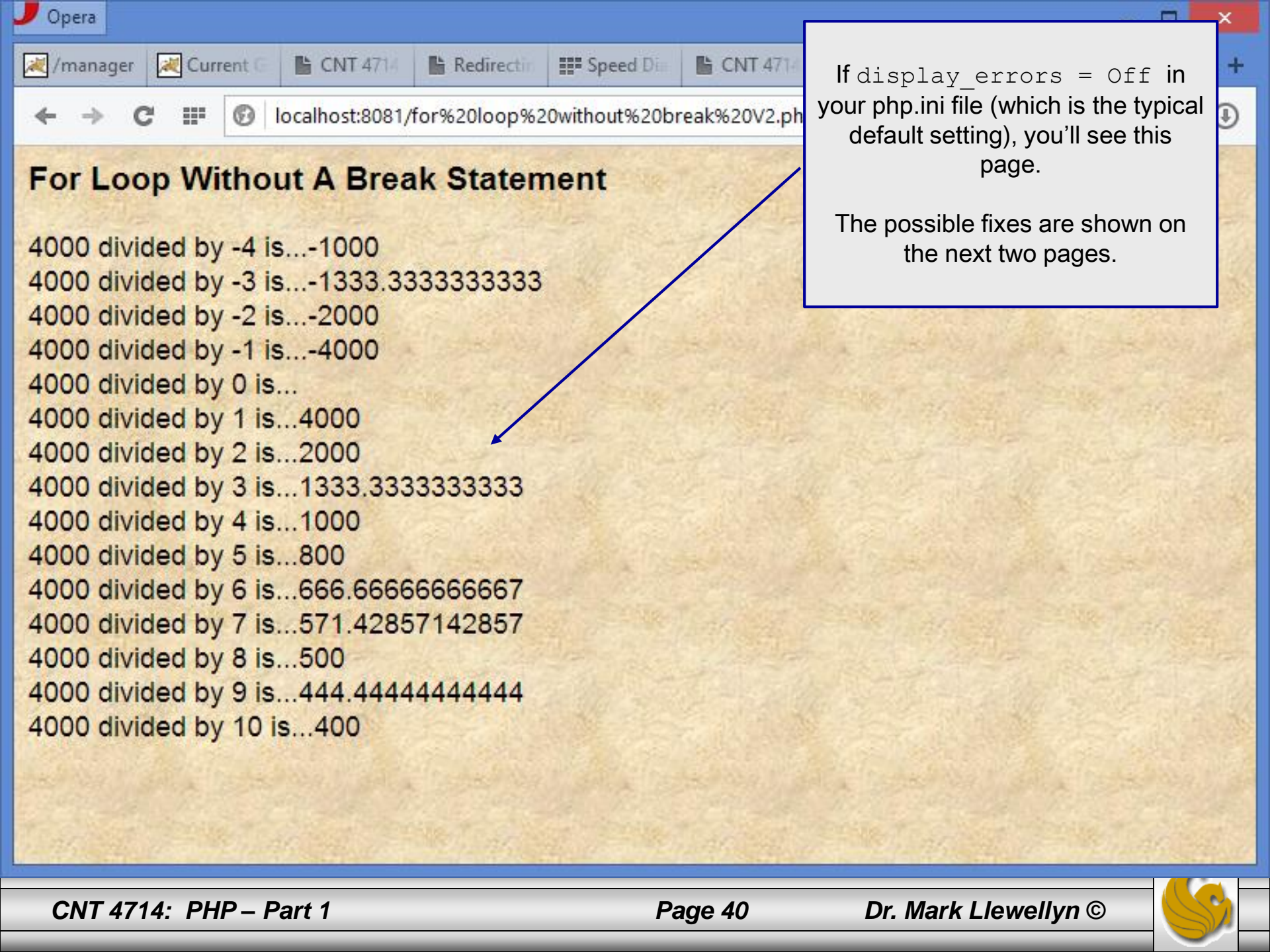
**Warning:** Division by zero in C:\Apache24\htdocs\for loop without break V2.php on line 12

4000 divided by 0 is...  
 4000 divided by 1 is...4000  
 4000 divided by 2 is...2000  
 4000 divided by 3 is...1333.3333333333  
 4000 divided by 4 is...1000  
 4000 divided by 5 is...800  
 4000 divided by 6 is...666.66666666667  
 4000 divided by 7 is...571.42857142857  
 4000 divided by 8 is...500  
 4000 divided by 9 is...444.44444444444  
 4000 divided by 10 is...400

Division by zero is not a fatal error in PHP. Instead a warning is generated an execution continues. (Note that this warning is only displayed if `display_errors = On` in your `php.ini` file. Otherwise it looks like the following page.

The possible fixes are shown on the next two pages.





## For Loop Without A Break Statement

4000 divided by -4 is...-1000  
4000 divided by -3 is...-1333.3333333333  
4000 divided by -2 is...-2000  
4000 divided by -1 is...-4000  
4000 divided by 0 is...  
4000 divided by 1 is...4000  
4000 divided by 2 is...2000  
4000 divided by 3 is...1333.3333333333  
4000 divided by 4 is...1000  
4000 divided by 5 is...800  
4000 divided by 6 is...666.666666666667  
4000 divided by 7 is...571.42857142857  
4000 divided by 8 is...500  
4000 divided by 9 is...444.444444444444  
4000 divided by 10 is...400

If `display_errors = Off` in your `php.ini` file (which is the typical default setting), you'll see this page.

The possible fixes are shown on the next two pages.

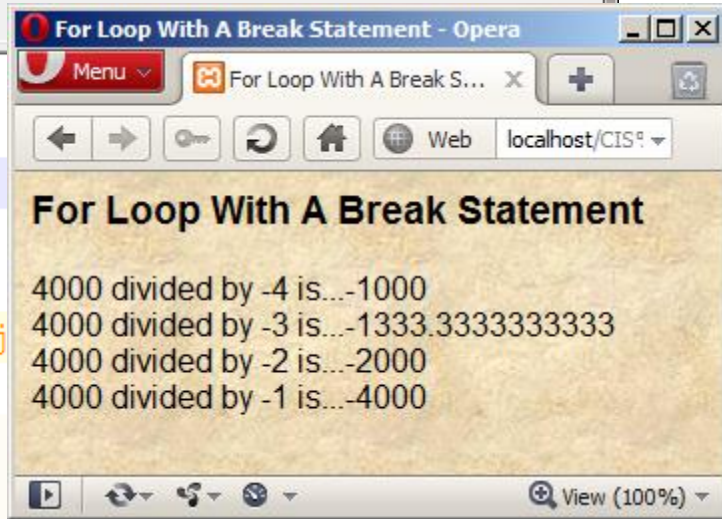




```

File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
for loop without break.php for loop without break V2.php for loop with break.php
1 <html>
2 <head>
3 <title>For Loop With A Break Statement</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6     background-color: #856363" background=image1.j
7 <h3> For Loop With A Break Statement</h3>
8 <?php
9     define("LIMIT", 10);
10    $counter = -4;
11    for (; $counter <= LIMIT; $counter++) {
12        if ($counter == 0) {
13            break;
14        } else {
15            $temp = 4000/$counter;
16            echo "4000 divided by ". $counter. " is...". $temp. "<br />";
17        }
18    }
19    ?>

```



Use a break statement to terminate the loop in a division by zero case.



```
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window
for loop without break.php for loop without break V2.php for loop with break.php for loop with cont

1 <html>
2 <head>
3 <title>For Loop With A Continue Statement</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6 background-color: #856363" background=image1.jp
7 <h3> For Loop With A Continue Statement</h3>
8 <?php
9 define("LIMIT", 10);
10 $counter = -4;
11 for (; $counter <= LIMIT; $counter++) {
12     if ($counter == 0) {
13         continue;
14     } else {
15         $temp = 4000/$counter;
16         echo "4000 divided by ". $counter. " is...". $temp. "<br />";
17     }
18 }
19 ?>
```

For Loop With A Continue Statement - Opera

Menu For Loop With A Continu... X

Web localhost/CIS?

### For Loop With A Continue Statement

4000 divided by -4 is...-1000  
4000 divided by -3 is...-1333.3333333333  
4000 divided by -2 is...-2000  
4000 divided by -1 is...-4000  
4000 divided by 1 is...4000 ←  
4000 divided by 2 is...2000  
4000 divided by 3 is...1333.3333333333  
4000 divided by 4 is...1000  
4000 divided by 5 is...800  
4000 divided by 6 is...666.666666666667  
4000 divided by 7 is...571.42857142857  
4000 divided by 8 is...500  
4000 divided by 9 is...444.444444444444  
4000 divided by 10 is...400

View (100%)

Use a continue statement to skip the division by zero case.

## Nested Loop Example

```
1 <html>
2 <head>
3 <title>For Loop With A Continue Statement</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6     background-color: #856363" background=image1.jpg>
7 <h3> Nested Loops</h3>
8 <?php
9     define("LOWER_LIMIT", 1);
10    define("UPPER_LIMIT", 12);
11    echo "<table style=\"border: 1px solid black; \"> \n";
12    for ($i = LOWER_LIMIT; $i <= UPPER_LIMIT; $i++) {
13        echo "<tr> \n";
14        for ($j = LOWER_LIMIT; $j <= UPPER_LIMIT; $j++) {
15            echo "<td style=\"border: 1px solid black; width:25px; padding:4px; text-align:center;\">";
16            echo ( $i * $j);
17            echo "<td> \n";
18        }
19        echo "<tr> \n";
20    }
21    echo "</table?";
22 ?>
23 </body>
24 </html>
```



For Loop With A Continue Statement - Opera

File Edit View Bookmarks Tools Help

Open Save Print Find Home Tile Cascade

Downloads Welcome to Opera For Loop With A C...

Web localhost:8081/nested%20loops.php Search with Google

Home Index Contents Search Glossary Help First Previous Next Last Up Copyright Author

## Nested Loops

1	2	3	4	5	6	7	8	9	10	11	12
2	4	6	8	10	12	14	16	18	20	22	24
3	6	9	12	15	18	21	24	27	30	33	36
4	8	12	16	20	24	28	32	36	40	44	48
5	10	15	20	25	30	35	40	45	50	55	60
6	12	18	24	30	36	42	48	54	60	66	72
7	14	21	28	35	42	49	56	63	70	77	84
8	16	24	32	40	48	56	64	72	80	88	96
9	18	27	36	45	54	63	72	81	90	99	108
10	20	30	40	50	60	70	80	90	100	110	120
11	22	33	44	55	66	77	88	99	110	121	132
12	24	36	48	60	72	84	96	108	120	132	144

Nested Loop Example -  
Output



# Controlling Script Flow In PHP

- The example on the next couple of pages illustrates a while loop. Again, I've used a form to extract user input. This time the user input sets the lower and upper limit on the loop.





whileloop.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>While Loop Demo</title>
5     <meta http-equiv="content-type" content="text/html' charset=iso-8859-1" />
6 </head>
7 <body style = "font-family: arial, sans-serif;
8     background-color: #856363" background=image1.jpg>
9 <form action="whileloop.php" method="post" >
10     Select Starting Number
11     <select name="start"><option>0</option> <option>1</option> <option>2</option>
12         <option>3</option> <option>4</option> <option>5</option> <option>6</option>
13         <option>7</option> <option>8</option> <option>9</option>
14     </select> <br /><br />
15     Select Ending Number
16     <select name="end"><option>0</option> <option>10</option> <option>11</option>
17         <option>12</option> <option>13</option> <option>14</option> <option>15</option>
18         <option>16</option> <option>17</option> <option>18</option> <option>19</option>
19     </select> <br />
20     <input type="submit" value="Submit" >
21     <input type="reset" value="Clear And Restart" >
22 </form>
23 </body>
24 </html>
```

# Controlling Script Flow In PHP

While Loop Demo - Opera

File Edit View Bookmarks Tools Help

Open Save Print Find Home Tile Cascade

Downloads Welcome to Opera While Loop Demo

localhost:8081/whileloop.html

Home Index Contents Search Glossary Help First Previous Next

Select Starting Number 3

Select Ending Number 12

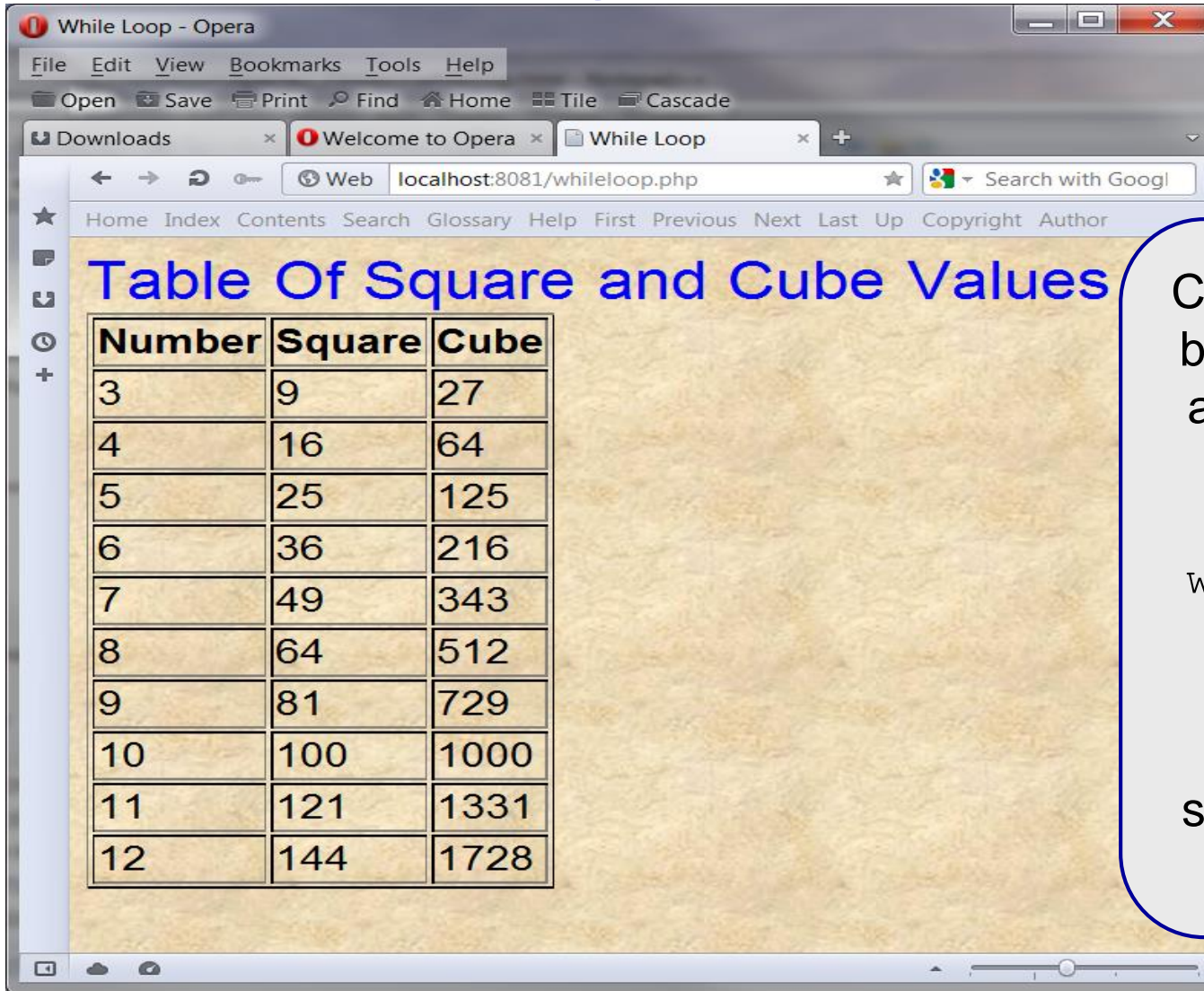
Submit Clear And Restart

0 10 11 12 13

Executing whileloop.html  
User enters lower and upper limits,  
clicks submit button.



# Controlling Script Flow In PHP



The screenshot shows a web browser window titled "While Loop - Opera". The address bar displays "localhost:8081/whileloop.php". The page content includes a navigation menu with links like "Home", "Index", "Contents", "Search", "Glossary", "Help", "First", "Previous", "Next", "Last", "Up", "Copyright", and "Author". The main heading is "Table Of Square and Cube Values". Below the heading is a table with three columns: "Number", "Square", and "Cube". The table contains data for numbers 3 through 12.

Number	Square	Cube
3	9	27
4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000
11	121	1331
12	144	1728

Clicking the submit button triggers the action of the form and invokes the script

`whileloop.php`

The script generates this page. The PHP script is shown on the next page.





whileloop.php

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>While Loop</title>
5     <meta http-equiv="content-type" content="text/html" charset="iso-8859-1" />
6 </head>
7 <body style = "font-family: arial, sans-serif;
8     background-color: #856363" background=image1.jpg>
9 <font size=5 color=blue> Table Of Square and Cube Values </font> <br />
10 <table border=1>
11     <th> Number </th> <th> Square </th> <th> Cube </th>
12     <?php
13         $start = $_POST["start"];
14         $end = $_POST["end"];
15         $i = $start;
16         while ($i <= $end) {
17             $square = $i * $i;
18             $cube = $i * $i * $i;
19             print ("<tr><td>$i</td><td>$square</td><td>$cube</td></tr>");
20             $i++;
21         }
22     ?>
23 </table>
24 </body>
25 </html>
```

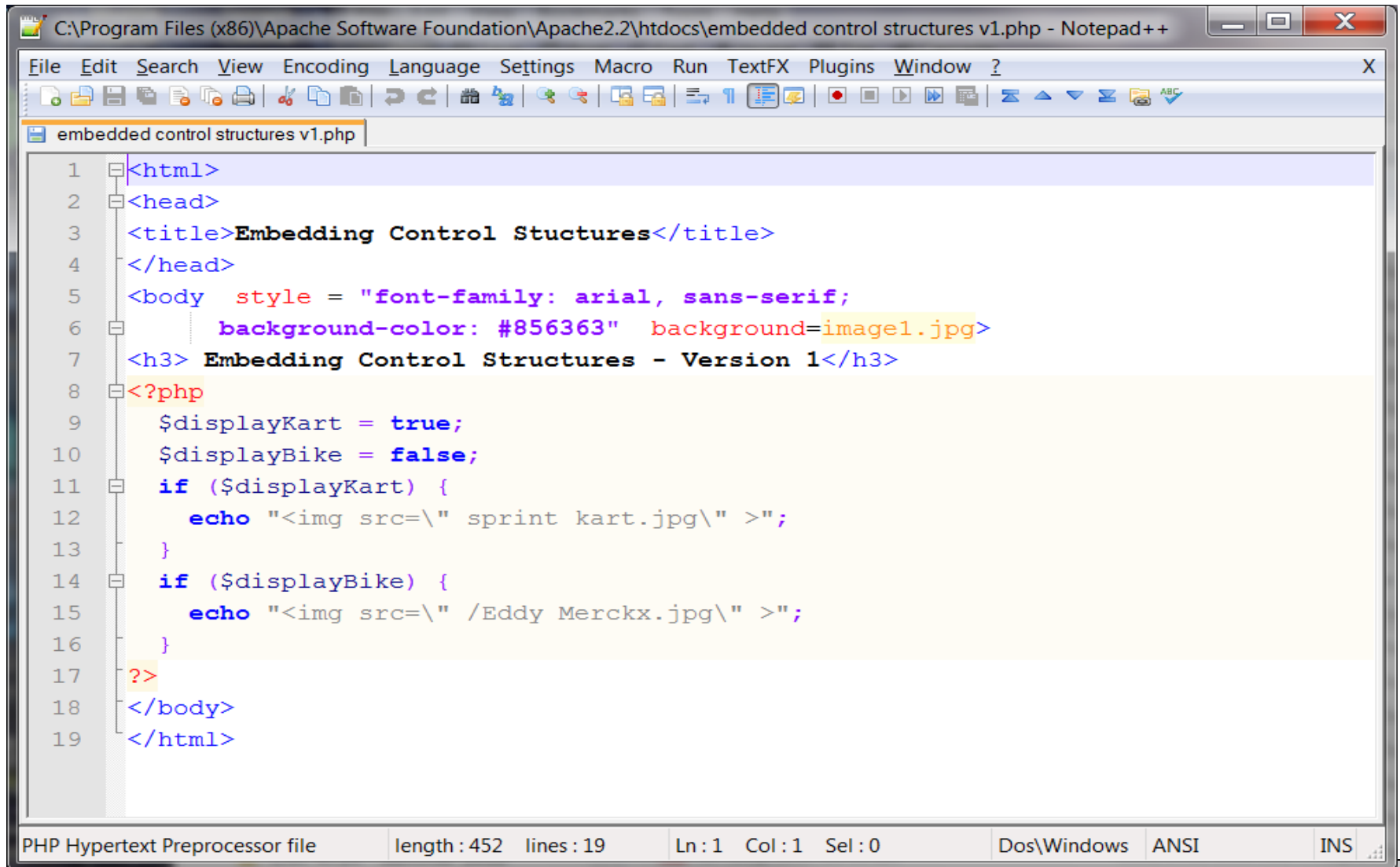


# Embedding Control Structures

- Now that we've seen most of the control structures in PHP, we need to see how these control structures can be used more effectively to produce HTML elements (or any other output).
- PHP is an embedded language that enables you to code both your HTML and the supporting script in the same document.
- PHP takes this concept a bit further by allowing you to “turn off” the PHP parser during a control structure and embed non-PHP output without losing the logic provided by the control structure.
- The following example, illustrates this concept by displaying an image in your HTML document only when a variable is set to true.



# Embedding Control Structures



```
C:\Program Files (x86)\Apache Software Foundation\Apache2.2\htdocs\embedded control structures v1.php - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
embedded control structures v1.php
1 <html>
2 <head>
3 <title>Embedding Control Structures</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6     background-color: #856363" background=imagem1.jpg>
7 <h3> Embedding Control Structures - Version 1</h3>
8 <?php
9     $displayKart = true;
10    $displayBike = false;
11    if ($displayKart) {
12        echo "<img src=\" sprintf kart.jpg\" >";
13    }
14    if ($displayBike) {
15        echo "<img src=\" /Eddy Merckx.jpg\" >";
16    }
17    ?>
18 </body>
19 </html>
PHP Hypertext Preprocessor file length : 452 lines : 19 Ln : 1 Col : 1 Sel : 0 Dos\Windows ANSI INS
```



# Embedding Control Structures



# Embedding Control Structures

- Although the previous solution works and for novice PHP programmers it seems to be the most obvious technique, PHP provides an alternate syntax that actually allows the embedding of the control structure into the markup.
- This alternative syntax is:

```
<?php . . .  
    if (conditional): ?>  
        - text/whatever that should be output but not parsed  
<?php endif;  
?>
```

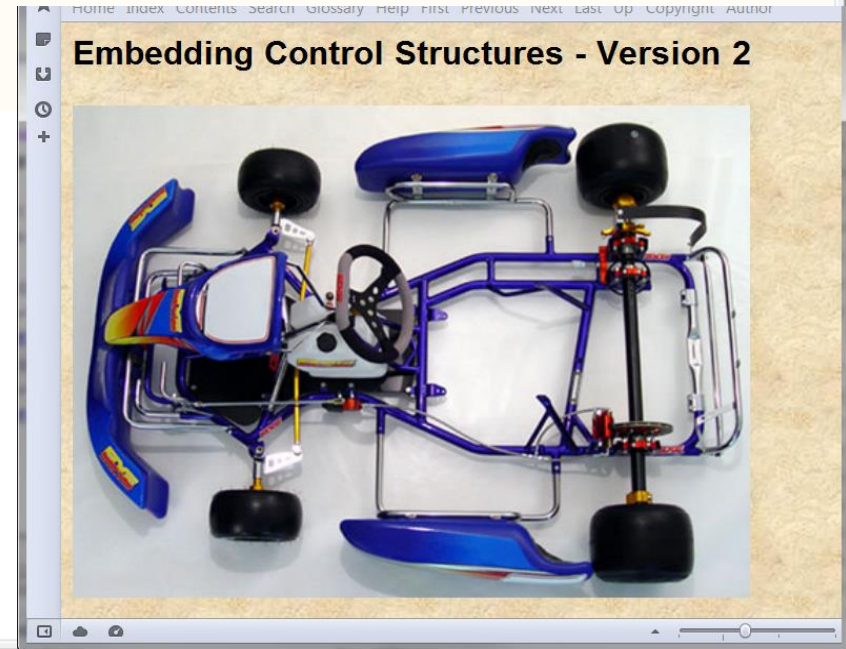
- This is shown in the next version of this example on the following page.



```

1 <html>
2 <head>
3 <title>Embedding Control Structures</title>
4 </head>
5 <body style = "font-family: arial, sans-serif;
6     background-color: #856363" background=image1.jpg>
7 <h3> Embedding Control Structures - Version 2</h3>
8 <?php
9     $displayKart = true;
10    $displayBike = false;
11 ?>
12 <?php if ($displayKart): ?>
13     
14 <?php endif; ?>
15 <?php if ($displayBike): ?>
16     
17 <?php endif; ?>
18
19 </body>
20 </html>

```

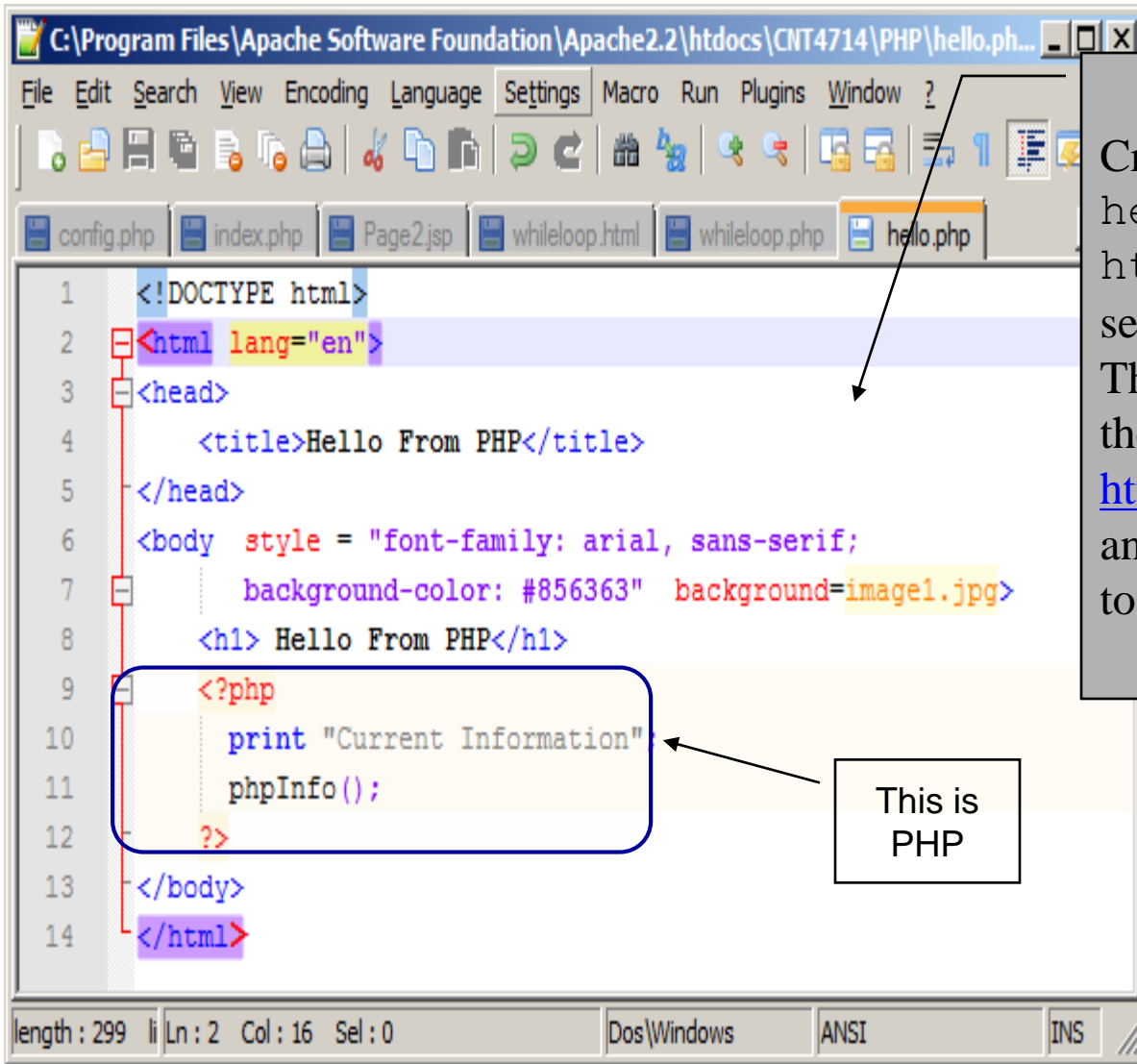


# Using PHP With REGISTER\_GLOBBALS OFF

- Since PHP 4.2.0, PHP is shipped with the REGISTER\_GLOBBALS configuration variable set to OFF. Prior to version 4.2.0 this variable was set to ON, but represented a fairly large security concern, so since that time the default setting is OFF. While this setting can be overridden by local system administrators, it is wise not to do so.
- Current versions of PHP register only a fixed set of global variables: GET, POST, COOKIE, and SERVER. By default ENV is not registered as is not recommended on production servers (you should use the getenv() function to handle this). You can tell your PHP site's status of GLOBBALS by running the phpInfo() function that was shown in the hello.php script in the setting up PHP section of notes (repeated here)...



# A PHP Test Example



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>Hello From PHP</title>
5 </head>
6 <body style = "font-family: arial, sans-serif;
7     background-color: #856363" background=imagem1.jpg>
8     <h1> Hello From PHP</h1>
9     <?php
10         print "Current Information"
11         phpInfo ();
12     ?>
13 </body>
14 </html>
```

This is PHP

Create this file named hello.php and save it to the htdocs folder in the Apache server.

Then start your browser and enter the URL:

<http://localhost:8081/hello.php>

and you should see output similar to that shown on the next slide.





sys_temp_dir	no value	no value
track_errors	Off	Off
unserialize_callback_func	no value	no value
upload_max_filesize	2M	2M
upload_tmp_dir	no value	no value
user_dir	no value	no value
user_ini.cache_ttl	300	300
user_ini.filename	.user.ini	.user.ini
variables_order	GPCS	GPCS
windows.show_crt_warning	Off	Off
xmlrpc_error_number	0	0
xmlrpc_errors	Off	Off
zend.detect_unicode	On	On
zend.enable_gc	On	On
zend.multibyte	Off	Off
zend.script_encoding	no value	no value

**ctype**

ctype functions	enabled
-----------------	---------

**date**

Scan down the output listing until you get to the Core settings for PHP and in this table you'll find the setting for variables\_order. This server is registered for GET, POST, COOKIE, and SERVER globals. First column is the local value and the second column is the master value (originally set implementation value)



# Using PHP With REGISTER\_GLOBS OFF

- You must receive HTML form input data using the `$_POST`, or `$_GET` associative arrays.
- Go back and look at the PHP scripts on pages 33 and 49 and you will see the `$_POST` associative array has been used in both examples to extract the input form data.
- We'll deal with this in more detail later, but for now these two examples should give you a good idea of how form data extraction is handled in PHP scripts.
- There are many other associative arrays utilized in PHP. The remainder of this set of notes is devoted to some of these arrays.

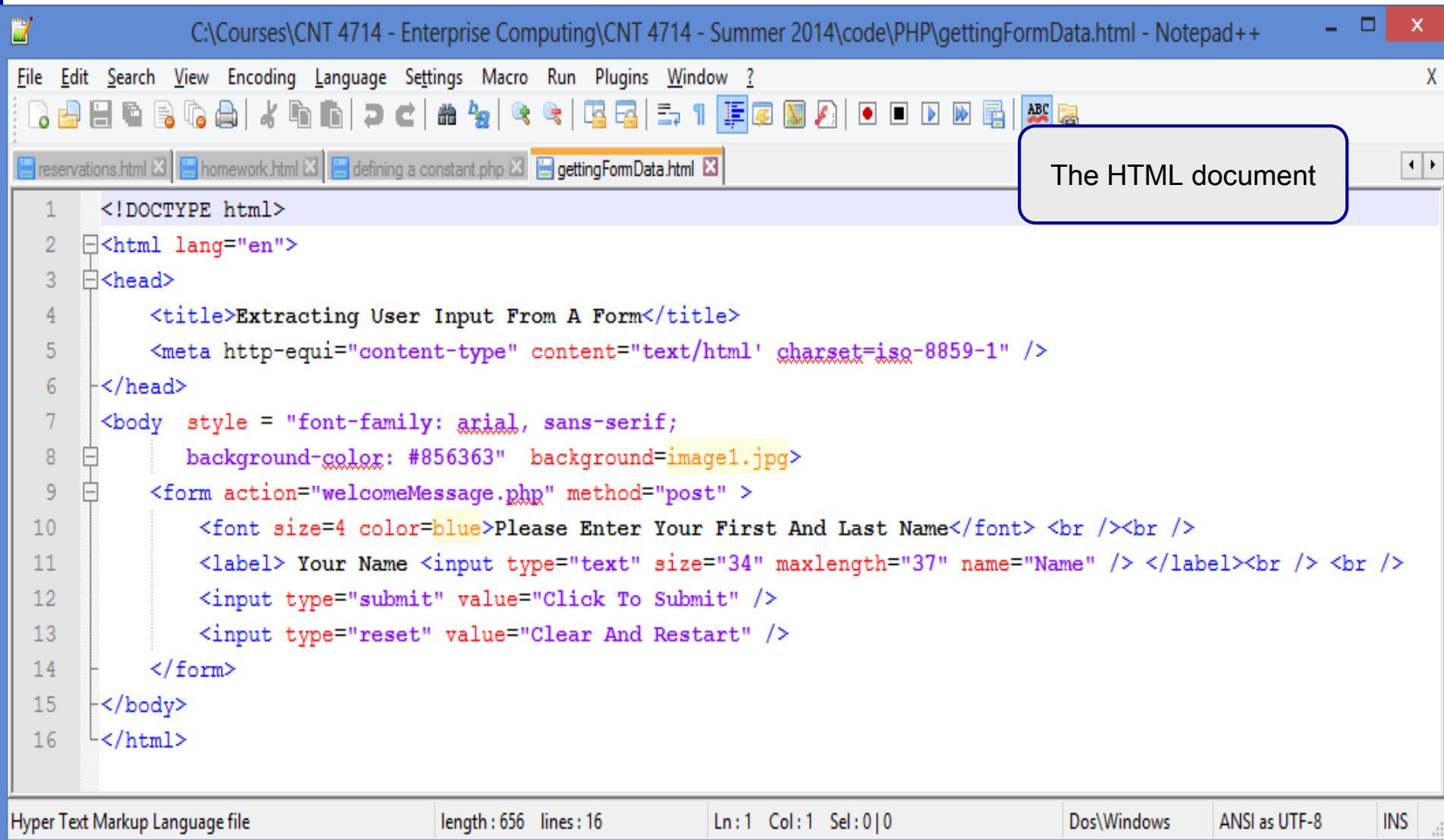


# A More Complete Example Using Form Data

- Several of the previous examples illustrated extracting user input from form variables. Let's look at one more example, where we prompt a user for their first and last name, and then use that information to display on every page of our website.
- First, let's write the HTML form and PHP script necessary to get the user's name.
- The HTML document is shown on page 60 and the PHP script is shown on page 61, with an example shown on page 62.



# A More Complete Example Using Form Data



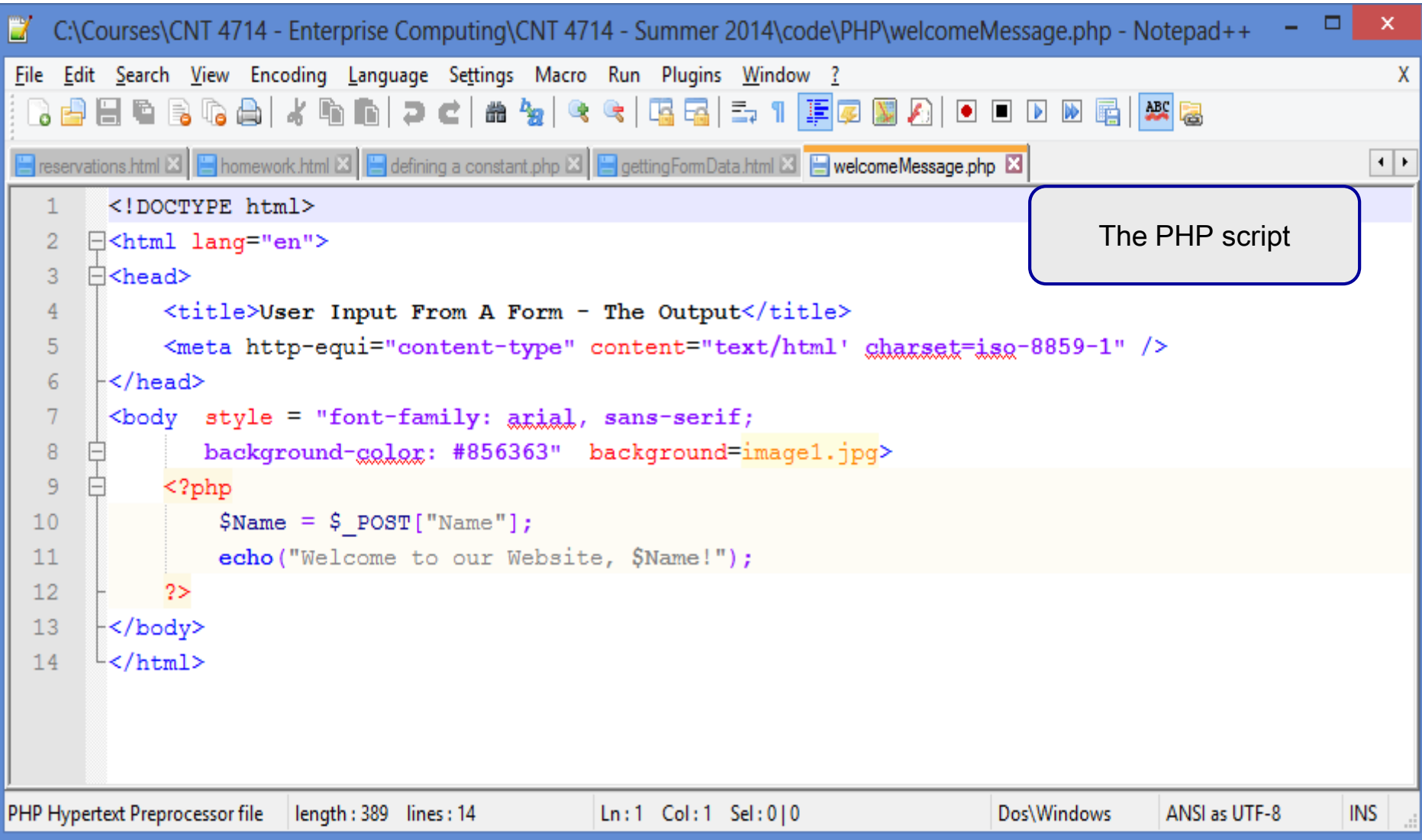
The HTML document

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>Extracting User Input From A Form</title>
5     <meta http-equiv="content-type" content="text/html" charset="iso-8859-1" />
6 </head>
7 <body style = "font-family: arial, sans-serif;
8     background-color: #856363" background=imageUrl.jpg>
9 <form action="welcomeMessage.php" method="post" >
10     <font size=4 color=blue>Please Enter Your First And Last Name</font> <br /><br />
11     <label> Your Name <input type="text" size="34" maxlength="37" name="Name" /> </label><br /> <br />
12     <input type="submit" value="Click To Submit" />
13     <input type="reset" value="Clear And Restart" />
14 </form>
15 </body>
16 </html>
```

Hyper Text Markup Language file      length : 656   lines : 16      Ln : 1   Col : 1   Sel : 0 | 0      Dos\Windows      ANSI as UTF-8      INS



# A More Complete Example Using Form Data



The screenshot shows a Notepad++ window with the following code:

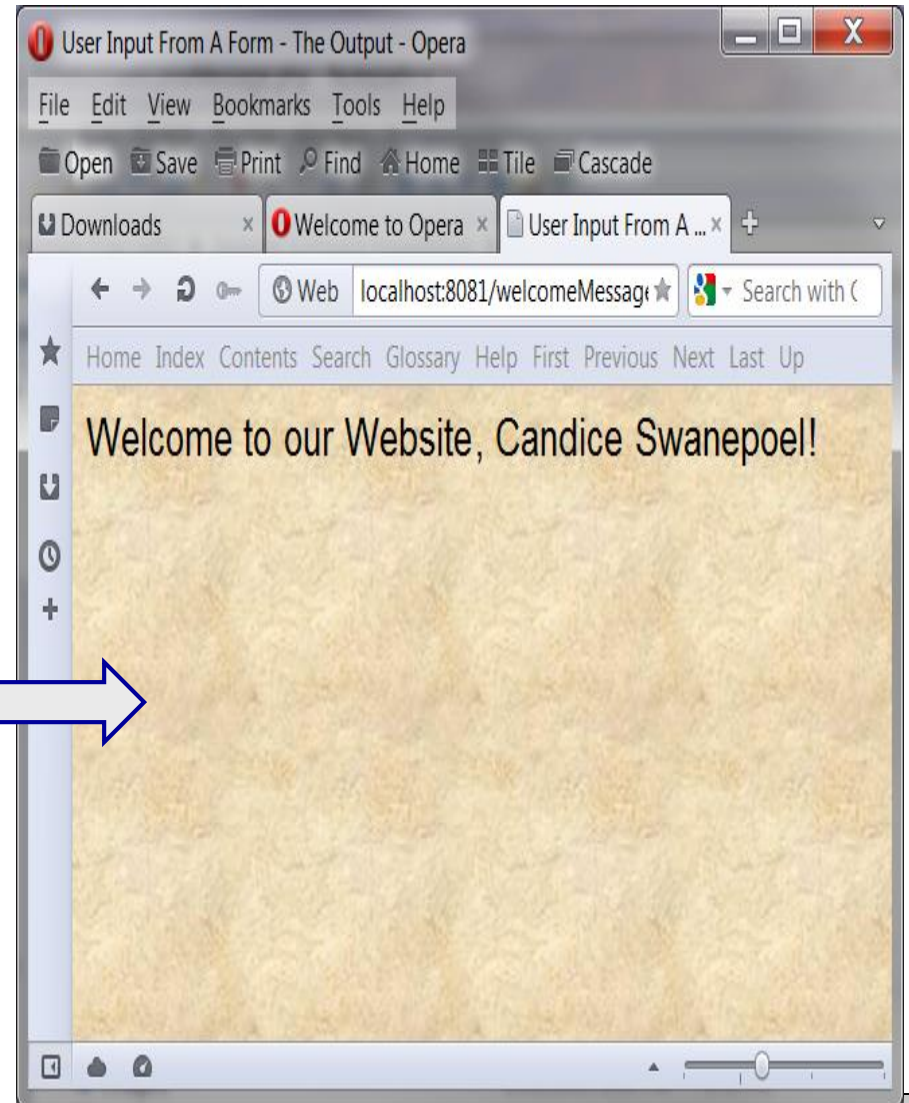
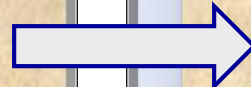
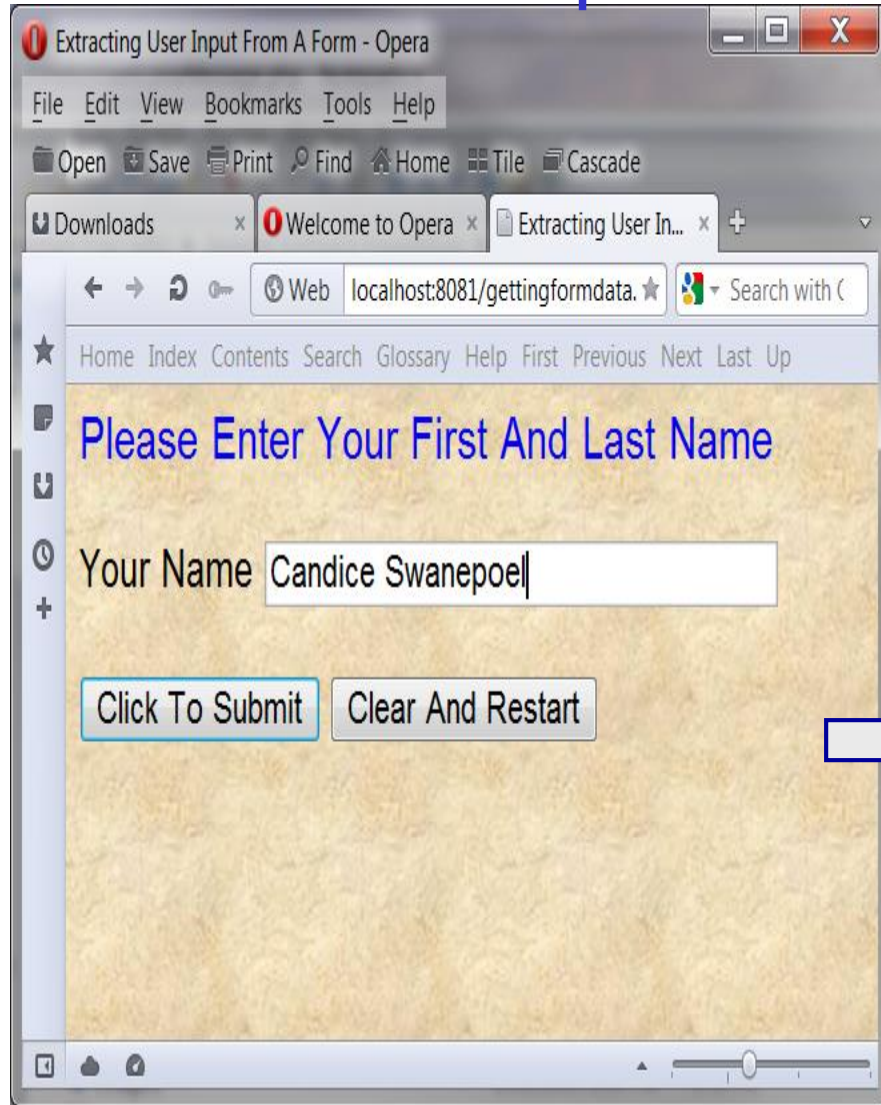
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>User Input From A Form - The Output</title>
5     <meta http-equiv="content-type" content="text/html" charset="iso-8859-1" />
6 </head>
7 <body style = "font-family: arial, sans-serif;
8     background-color: #856363" background="image1.jpg">
9     <?php
10         $Name = $_POST["Name"];
11         echo("Welcome to our Website, $Name!");
12     ?>
13 </body>
14 </html>
```

A callout box on the right side of the code editor contains the text: "The PHP script".

At the bottom of the Notepad++ window, the status bar displays: "PHP Hypertext Preprocessor file | length : 389 | lines : 14 | Ln : 1 | Col : 1 | Sel : 0 | 0 | Dos\Windows | ANSI as UTF-8 | INS".



# A More Complete Example Using Form Data



# A More Complete Example Using Form Data

- Now suppose that we wanted to construct our website so that it showed the visitor's name at the top of every page.
- While the script we've just constructed gets us the required data, there are a couple of problems we need to overcome, if we're going to do this right.
  1. We need the name at the top of every page, not just the first one.
  2. We have no control over which page at our site the user might first enter.
- The first problem is fairly easy to handle, once the visitor's name is in a variable on one page, we simply pass it as part of any request for another page by adding the name to the query string of all links.



# A More Complete Example Using Form Data

- To pass the name as part of a query string of a link, we'll embed PHP code right into the middle of an HTML tag.
- This is perfectly legal and will work just fine.
- The revised link would look like the following:

```
<a href="newpage.php?name=<?php echo urlencode($_GET["name"]);?> />
```

- The `urlencode` function takes special characters in the string, e.g., spaces, and converts them into the special codes they need to be in order to appear in the query string.
- For example, if the `$name` variable had a value of “Mark Llewellyn”, then, since spaces are not allowed in a query string, the output of `urlencode` would be “Mark+Llewellyn”. PHP would then convert it back automatically when it created the `$_GET` variable in `newpage.php`.





# A More Complete Example Using Form Data

- Now that we know how to handle the first problem, we now need to figure out how to get the name in the first place.
- We constructed a special HTML document that contained a form for the visitor to enter their name. The problem is that we don't want to force the user to enter our site by that page every time they visit the site.
- The solution is to have every page in the site check to see if a name has been specified and prompt the user for one if necessary.
- Pages in websites that can decide whether to display one thing or another are referred to as **multipurpose pages**. A basic template for a multipurpose page is shown on the following slide. With examples for our specific case shown on the pages following the template.



# A More Complete Example Using Form Data

The screenshot shows a web editor window with the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>Multipurpose Web Page Template</title>
5   <meta http-equiv="content-type" content="text/html" charset="iso-8859-1" />
6 </head>
7 <body style = "font-family: arial, sans-serif;
8   background-color: #856363" background=image1.jpg>
9   <?php if (condition) { ?>
10    <!-- HTML content to display if condition is true goes here -->
11   <?php } else { ?>
12    <!-- HTML content to display if condition is false goes here -->
13   <?php } ?>
14 </body>
15 </html>
```

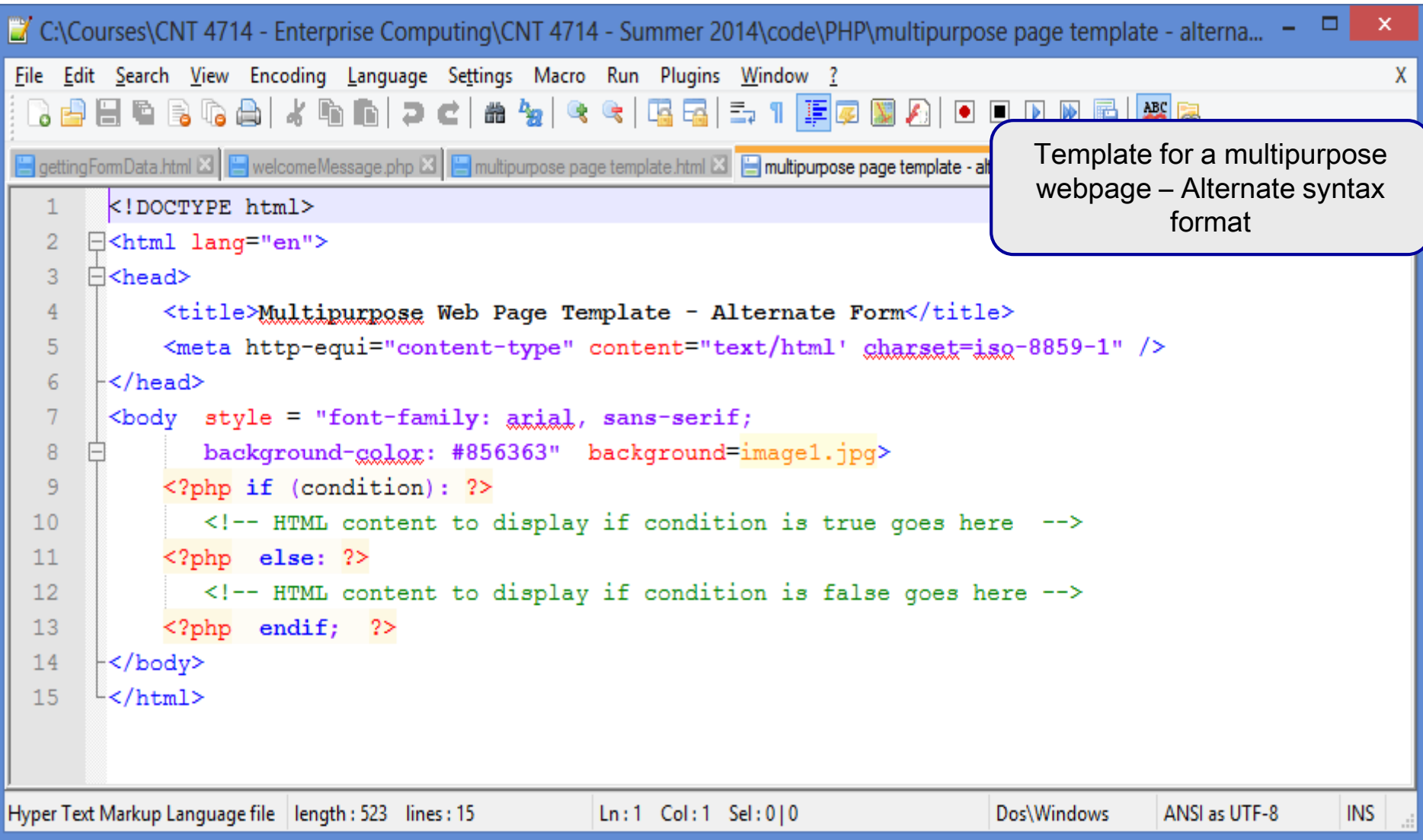
Annotations in the image:

- A box labeled "Enter PHP mode" with an arrow pointing to the opening PHP tag on line 9.
- A box labeled "Exit PHP mode" with an arrow pointing to the closing PHP tag on line 13.
- A box labeled "Template for a multipurpose webpage" pointing to the overall code structure.

Editor status bar: Hyper Text Markup Language file | length : 503 | lines : 15 | Ln : 1 | Col : 1 | Sel : 0 | 0 | Dos\Windows | ANSI as UTF-8 | INS



# A More Complete Example Using Form Data



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <title>Multipurpose Web Page Template - Alternate Form</title>
5     <meta http-equiv="content-type" content="text/html" charset="iso-8859-1" />
6 </head>
7 <body style = "font-family: arial, sans-serif;
8     background-color: #856363" background=image1.jpg>
9     <?php if (condition): ?>
10         <!-- HTML content to display if condition is true goes here -->
11     <?php else: ?>
12         <!-- HTML content to display if condition is false goes here -->
13     <?php endif; ?>
14 </body>
15 </html>
```

Template for a multipurpose webpage – Alternate syntax format

Hyper Text Markup Language file | length : 523 | lines : 15 | Ln : 1 | Col : 1 | Sel : 0 | 0 | Dos\Windows | ANSI as UTF-8 | INS



# Viewing Client/Server Environment Variables

- Knowledge of a client's execution environment is useful to system administrators who want to provide client-specific information.
- Environment variables contain information about a script's environment, such as the client's web browser, the HTTP host and the HTTP connection.
  - The table on the next page summarizes some of the superglobal arrays defined by PHP.
- The HTML document on page 70 displays the values of the server's environment variables in a table. PHP stores the server variables and their values in the `$_SERVER` array. Iterating through the array allows one to view all of the server's environment variables.



# Some Superglobal Environment Arrays

Variable Name	Description
<code>\$_SERVER</code>	Data about the currently running server.
<code>\$_ENV</code>	Data about the client's environment.
<code>\$_GET</code>	Data posted to the server by the <code>get</code> method.
<code>\$_POST</code>	Data posted to the server by the <code>post</code> method.
<code>\$_COOKIE</code>	Data contained in cookies on the client's computer.
<code>\$GLOBALS</code>	Array containing all global variables.



# server.php Example

C:\Courses\CNT 4714 - Enterprise Computing\CNT 4714 - Summer 2014\code\PHP\server.php - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?



gettingFormData.html x welcomeMessage.php x multipurpose page template.html x multipurpose page template - alternate form.html x server.php x

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <!-- server.php -->
4 <!-- Program to display $_SERVER variables -->
5 <head>
6 <title>SERVER Variables Display</title>
7 </head>
8 <body style = "font-family: arial, sans-serif;
9 background-color: #856363" background=imageUrl.jpg>
10 <table border = "0" cellpadding = "2" cellspacing = "0"
11 width = "100%">
12 <?php
13 // print the key and value for each element
14 // in the $_SERVER array
15 foreach ( $_SERVER as $key => $value )
16 print( "<tr><td bgcolor = \"#11bbff\">
17 <strong>$key</strong></td>
18 <td>$value</td></tr>" );
19 ?>
20 </table>
21 </body>
22 </html>
```

Iterate through the \$\_SERVER array to list all of the SERVER variables for the current server on which PHP is running.



Opera

/manager x SERVER Variables Display x +

localhost:8081/server.php

```

HTTP_HOST localhost:8081
HTTP_CONNECTION keep-alive
HTTP_ACCEPT text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
HTTP_USER_AGENT Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36 OPR/22.0.1471.70
HTTP_ACCEPT_ENCODING gzip,deflate,lzma,sdch
HTTP_ACCEPT_LANGUAGE en-US,en;q=0.8
PATH c:\Program Files (x86)\AMD APP\bin\x86_64;c:\Program Files (x86)\AMD APP\bin\x86;c:\Program Files (x86)\Intel\CLS Client\;C:\Program Files\Intel\CLS Client\;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Program Files\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files\Intel\Intel(R) Management Engine Components\IPT;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\DAL;C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\IPT;c:\Program Files (x86)\ATI Technologies\ATIACE\Core-Static;C:\Program Files\Dell\Direct\Protection\Access\Advanced\Wave\Gemalto\Access Client\v5\;C:\Program Files (x86)\Security Innovation\SI TSS\bin\;C:\Program Files (x86)\Windows Live\Shared;C:\Users\Mark Llewellyn\AppData\Local\Smartbar\Application\;C:\Users\Mark Llewellyn\AppData\Local\Smartbar\Application\;C:\Program Files (x86)\MySQL\MySQL Utilities 1.3.6\;C:\Program Files (x86)\Symantec\Symantec Communications Security\SSH Secure Shell
SystemRoot C:\Windows
COMSPEC C:\Windows\system32\cmd.exe
PATHEXT .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
WINDIR C:\Windows
SERVER_SIGNATURE
SERVER_SOFTWARE Apache/2.4.6 (Win64) mod_jk/1.2.37 PHP/5.5.5
SERVER_NAME localhost
SERVER_ADDR ::1
SERVER_PORT 8081
REMOTE_ADDR ::1
DOCUMENT_ROOT C:/Apache24/htdocs
REQUEST_SCHEME http

```

