Chapter 4

PHP Fundamentals

4.1 String Delimiters

This section is on dealing with string delimiters (i.e. bits of text data) and specifically on dealing with double quotes and single quotes. When double and single quotes both used within a string, it can be tricky to get it to print out correctly.

Here is an example simple form, where I'm just entered my first name, but before I submit this form, I want you to see what the goal is.

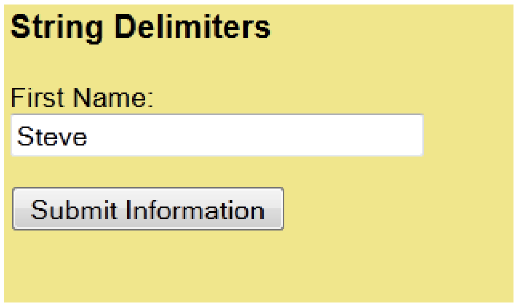


Figure 4.1.1 – Simple Web Form

For every first name that is entered, I want it to display a response in the following format…

"Hey Steve!" I said, "don't do that."

… when I submit it. I want the double quotes to be in the display.

The **print** function, as we have seen, is sent some text (a string) within double or single quotes and that text displays on the browser. So…

```
    print 'Hi there';
```

…or …

```
    print "Hi there";
```

… works the same.

The only rule is that whatever type of quote you start with, you must end with as well.

Assume my PHP program has retrieved the first name entered by a user into a variable named $firstname. What would happen with the following PHP code?

```
    print "<p>"Hey $firstname!", I said. "Don't do that."";
```

From PHP's perspective the **print** function is processed like this…

```
    print "<p>"
```

Since a double quote starts the string, PHP assumes that the next double quote found ends the string, The next thing PHP encounters is the word "Hey" and has no clue what to do with it. Hence, we will get a syntax (i.e. parse) error and the program will stop working.

One way around this is to use the **escape key,** which is a backward slash (\ ). If you put an escape key in front of a double or single quote it tells the print function to treat the quote as a character in the string and not to process it as a delimiter.

So, here is a correct way to do this…

```
  print '<p>"Hey '.$firstname.'!", I said. "Don\'t do that."';
```

Remembering that a dot (.) means to append strings together, here's how PHP "sees" the statement above.

```
    print
    '<p>"Hey '
    .
```

```
$firstname
.
'!", I said. "Don\'t do that."';
```

It processes three separate string appended together with the dot operator.

Let's look closer at that last string…

```
'!", I said. "Don\'t do that."';
```

Notice that this entire string is delimited using single quotes. That is, the string begins after the first single quote and ends just before the last single quote.

There is another single quote in the middle of the string but it is not "seen" as a delimiter by PHP since it is preceded immediately with an **escape key**.

The double quotes in this string don't "confuse" the **print** function either since single quotes are being used as the delimiters in this statement, not double quotes.

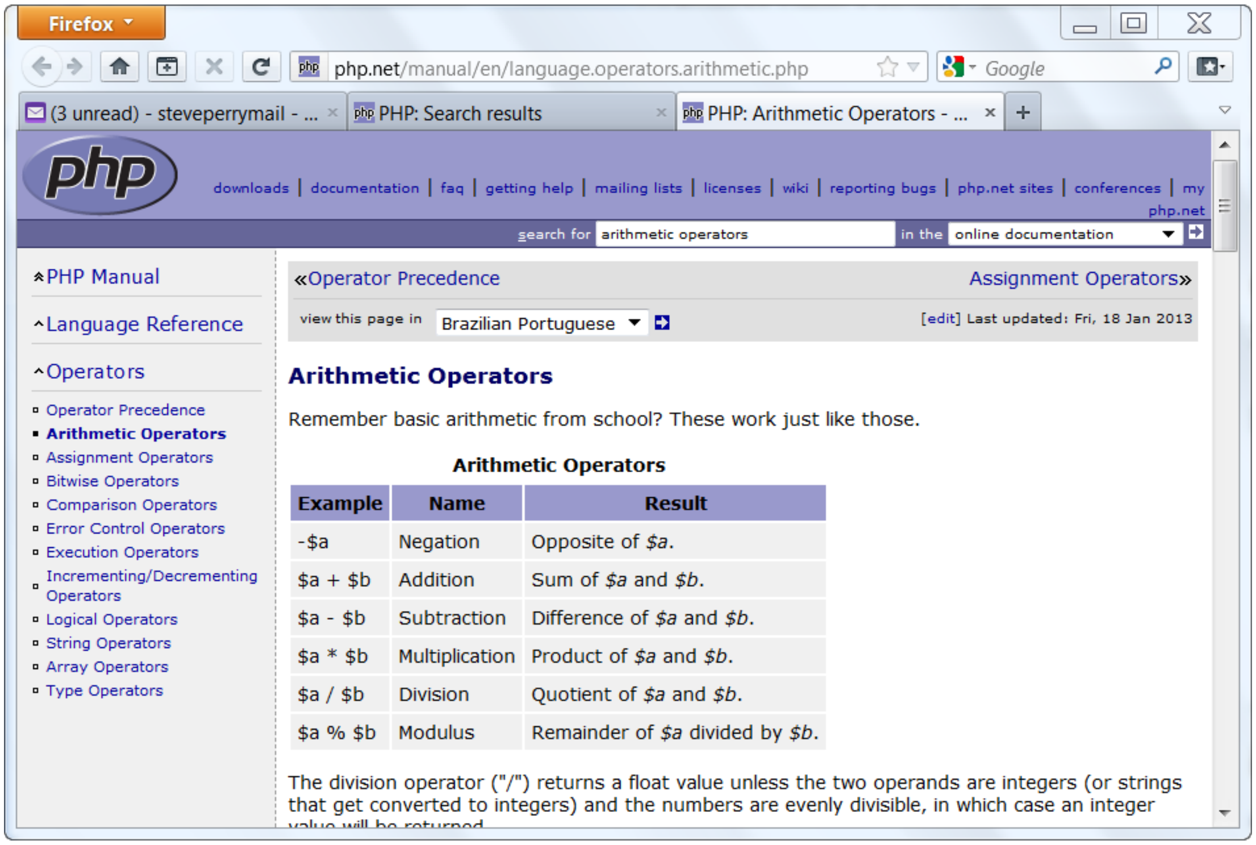
4.2 Simple Math

A very good resource for PHP programmers is PHP.net. This is the most important site for giving you documentation on what is current with PHP.

More often than not, you'll search for functions you need, such as "what does the **strpos** function do, or what does an **array-push** function do?

I want information about the "arithmetic operators" that come with PHP. So I'm going to use the drop-down, and select "online documentation".

Then I click the little arrow to begin the search.


Firefox
php.net/manual/en/language.operators.arithmetic.php
Google
(3 unread) - steveperrymail - ...
PHP: Search results
PHP: Arithmetic Operators - ...
downloads | documentation | faq | getting help | mailing lists | licenses | wiki | reporting bugs | php.net sites | conferences | my php.net
search for arithmetic operators in the online documentation
PHP Manual
Language Reference
Operators
Operator Precedence
Arithmetic Operators
Assignment Operators
Bitwise Operators
Comparison Operators
Error Control Operators
Execution Operators
Incrementing/Decrementing Operators
Logical Operators
String Operators
Array Operators
Type Operators
«Operator Precedence
Assignment Operators»
view this page in Brazilian Portuguese
[edit] Last updated: Fri, 18 Jan 2013

Arithmetic Operators

Remember basic arithmetic from school? These work just like those.

Arithmetic Operators

Example	Name	Result
-$a	Negation	Opposite of *$a*.
$a + $b	Addition	Sum of *$a* and *$b*.
$a - $b	Subtraction	Difference of *$a* and *$b*.
$a * $b	Multiplication	Product of *$a* and *$b*.
$a / $b	Division	Quotient of *$a* and *$b*.
$a % $b	Modulus	Remainder of *$a* divided by *$b*.

The division operator ("/") returns a float value unless the two operands are integers (or strings that get converted to integers) and the numbers are evenly divisible, in which case an integer

Figure 4.2.1 – Arithmetic Operator documentation shown on PHP.net

Presumably, all of you are familiar with basic arithmetic. How to add, subtract, multiply, divide, that sort of thing. But we have to learn how PHP does it. Fortunately, it's very similar to how we do it on a piece of paper.

Addition, uses the plus sign (+) and subtraction is just like you expect with a minus sign (-), which is the same as the hypen key.

Multiplication with most programming languages done with an asterisk (*) between the two variables, not a little "x" like you might put on a piece of paper.

Division is done with a forward slash (/).

Here's something you may or may not be familiar with, it's called modulus and it is used to find the remainder of something. This can actually be very useful on routines where we want to do an alternate of something like say, in an HTML table, we want to display a different color background on each row.

The percent (%) symbol is used between the two integers. For example…

x = 8 % 5

… here x would equal 3, since 5 goes into 8 once with a remainder of 3.

To format a number with commas, decimal places, etc. use the **number_format** function. Look it up on PHP.net

You will only need simple arithmetic throughout as we build up the assignments, so you do not have to be a Math Wizard to take this class.

4.3 Using HTML Tables

This section is on using HTML tables in your webpage and outputting an HTML table from your PHP.

Look at this form, it looks similar to something we've seen before but things line up kind of nice.

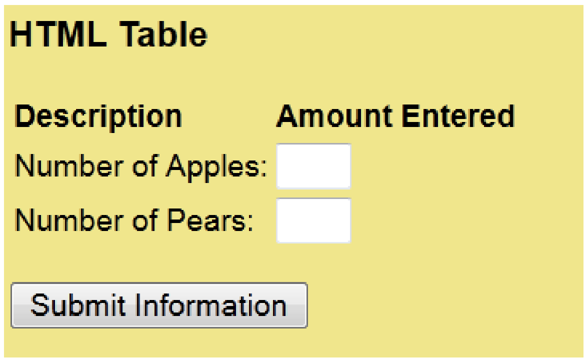


Figure 4.3.1 – Simple Web Form formatted using an HTML table

I've got the number of apples and the number of pears in boxes that line up right below each other. Even the titles line up nicely.

Let's look at the HTML code I used for this…

```
<body>

<h3>HTML Table</h3>
```

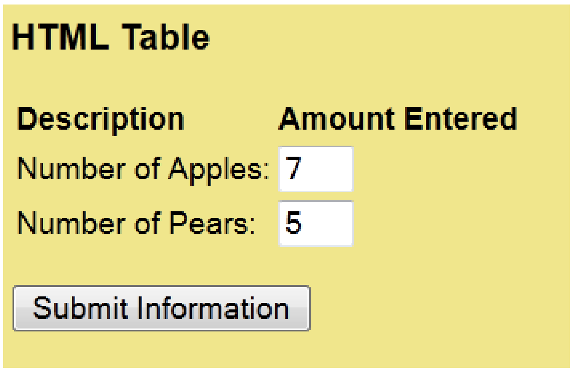
```
<form method="post" action="0403_HTML_Table.php">

<table>
     <tr align="left">
          <th>Description</th>
          <th>Amount Entered</th>
     </tr>

     <tr>
          <td>Number of Apples:</td>
          <td>
               <input type="text" name="apples" size="2" />
          </td>
     </tr>

     <tr>
          <td>Number of Pears:</td>
          <td>
               <input type="text" name="pears" size="2" />
          </td>
     </tr>
</table>

<p>
     <input type="submit" value="Submit Information" />
</p>

</form>

</body>
```

Figure 4.3.2 – HTML code for Simple Web Form formatted using an HTML table

Notice the key tag pairs here, <table> and </table>, <tr> and </tr>, and <td> and </td> that are used to define a table. This will force the data in the table to be aligned nicely in simple grid.

NOTE: Nowadays, HTML tables are generally not used for Web page formatting, but for a simple form like this I, personally, think it is OK to use. A professional web page designer would, instead, use CSS code to position these elements.

Now I will enter some data…



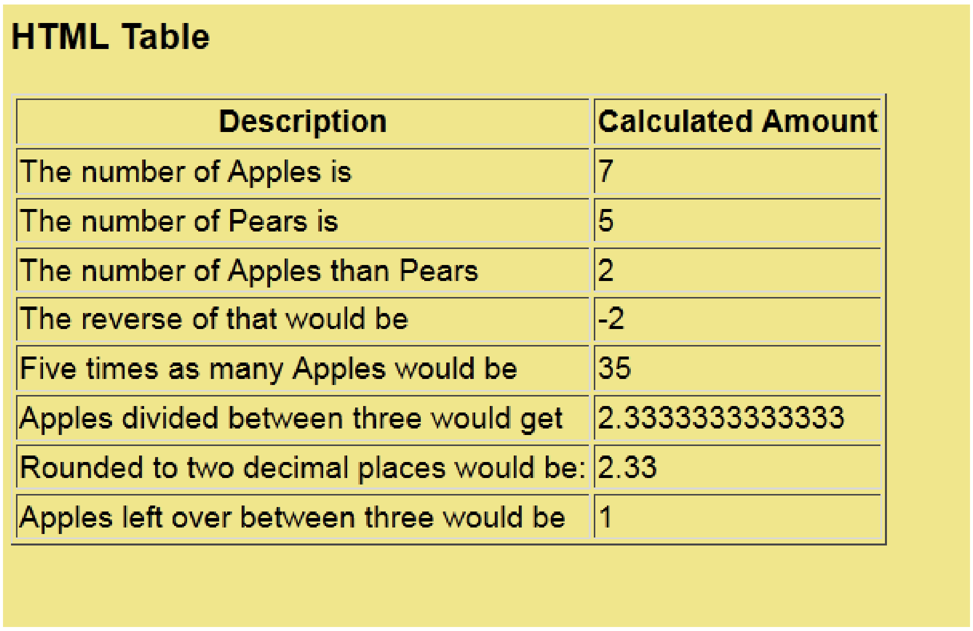
HTML Table

Description	Amount Entered
Number of Apples:	7
Number of Pears:	5

Submit Information

Figure 4.3.3 – Simple Web Form formatted using an HTML table (with data)

… and click the Submit button to see….



HTML Table

Description	Calculated Amount
The number of Apples is	7
The number of Pears is	5
The number of Apples than Pears	2
The reverse of that would be	-2
Five times as many Apples would be	35
Apples divided between three would get	2.3333333333333
Rounded to two decimal places would be:	2.33
Apples left over between three would be	1

Figure 4.3.4 – Response to Simple Web Form formatted using an HTML table

Notice we have a lot of information here! Even though I only had two pieces of data (i.e. the number of apples and pears) I did quite a lot with them.

Also, you can more clearly see that the output is presented in an HTML table since I am displaying a table border so that the user can see the grid.

Here is the code for the PHP file that displayed the page above…

```
<body>

<h3>HTML Table</h3>

<?php
     $apples = $_POST['apples'];
     $pears = $_POST['pears'];

     $totalFruit = $apples + $pears;

     $diff = $apples - $pears;

     $reverseDiff = -$diff;

     $multpliedApples = $apples * 5;

     $dividedApples = $apples / 3;

     $moduloApples = $apples % 3;

?>

<table border='1'>
     <tr>
          <th>Description</th>
          <th>Calculated Amount</th>
     </tr>

<?php

     print "<tr><td>The number of Apples is
          </td><td>$apples</td></tr>\n";

     print "<tr><td>The number of Pears is
          </td><td>$pears</td></tr>\n";

     print "<tr><td>The number of Apples than Pears
          </td><td>$diff</td></tr>\n";

     print "<tr><td>The reverse of that would be
          </td><td>$reverseDiff</td></tr>\n";

     print "<tr><td>Five times as many Apples would be
          </td><td>$multpliedApples</td></tr>\n";
     print "<tr><td>Apples divided between three would get
          </td><td>";

     print $dividedApples;

     print "</td></tr>\n";

     print "<tr><td>Rounded to two decimal places would be:
          </td><td>";

     print number_format($dividedApples, 2);

     print "</td></tr>\n";

     print "<tr><td>Apples left over between three would be
          </td><td>";

     print $moduloApples;

     print "</td></tr>\n";
?>

</table>

</body>
```

Figure 4.3.5 – PHP code that processed the Web form

Notice that I am using more than one PHP block (i.e. <?php and ?> tags) and that everything gets placed in an HTML table.

4.4 The IF Statement

This section is on using the **IF** statement.

The **IF** statement is probably the most important statement you'll ever learn in any programming language. It allows you to make a decision about something and do something different based on the value of some data. Without it, you really aren't programming. So far, we've been able to just get information into our programs and display them in different ways, but now, we're taking that first step to using logical paths to get something different to happen based on data.

Here's simple form with some data keyed into it.

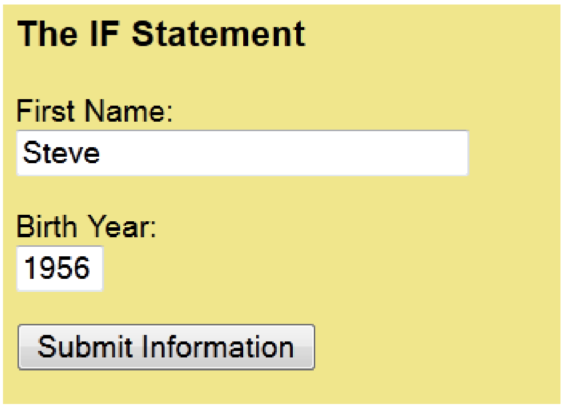
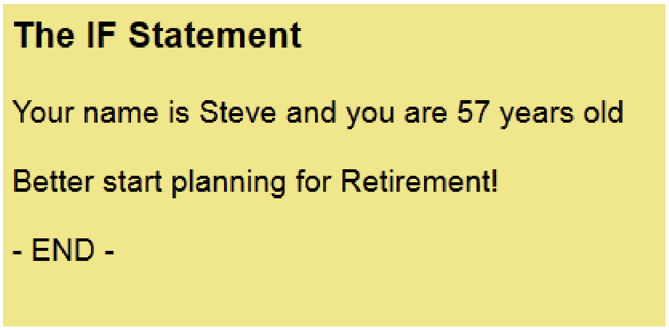


Figure 4.4.1 – Simple Web form

When I click the submit button it returns…



The IF Statement

Your name is Steve and you are 57 years old

Better start planning for Retirement!

- END -

Figure 4.4.2 – Simple Web form response

Let's take a look at the code.

```
<body>

<h3>The IF Statement</h3>

<?php
     $firstname = $_POST['firstname'];
     $birthyear = $_POST['birthyear'];

     $current_year = date('Y');

     $age = $current_year - $birthyear;

     print "<p>Your name is $firstname
          and you are $age years old</p>";

     if ($age > 50)
     {
          print "<p>Better start planning
               for Retirement!</p>";
     }

     print "<p> - END - </p>";
?>

</body>
```

Figure 4.4.3 – PHP code that processed the Web form

We're getting two pieces of data, the first name and the birth year this time.

I'm getting the date so I can get just the current year. Notice that I passed the parameter with a capital 'Y', that brings back a 4-character year into the current year of the variable I've set up.

Now, I'm going to calculate my age. This is approximate age, of course. I'm just taking the current year, subtracting the birth year that I entered.

So, just like we've done before the form displays the user's first name and how old they are.

Look at the **IF** statement.

This is the first time we're going to make a decision and we're going to do something different on the web page based on a value based on a comparison.

Is the age greater than 50?

There can only be two answers from the standpoint of programming. The comparison is either TRUE, if it's greater than 50, or FALSE, if age is not greater than 50.

If I am older than 50 the code, that comes after the **IF** statement inside the curly braces, will run. If I am not older than 50, it will just skip pass the last curly brace and print the last line of the program.

Now, we're always going to print this last line. The only question is, are we going to print this paragraph about starting for retirement?

Now, technically speaking, you don't have to use opening and closing curly braces if you have just one line that you want to execute after the comparison. However, I find it a good habit to always use them whether you need them or not. I find it easy to look at the structure of the program to follow what's going on, and if I'd make a change later to my program, maybe I want to add another line not just start planning for retirement, but I want to do something else as well that I want to print out. It's very easy to forget to go back and put those curly braces back in. But when I see them there, it's really easy to remember to put the code within the curly braces that I want to run when my age is greater than 50.

4.5 The Else Clause

This section is on using the **else** clause of the **if** statement to know what to do when something is FALSE in the comparison.

Here is a form with data…

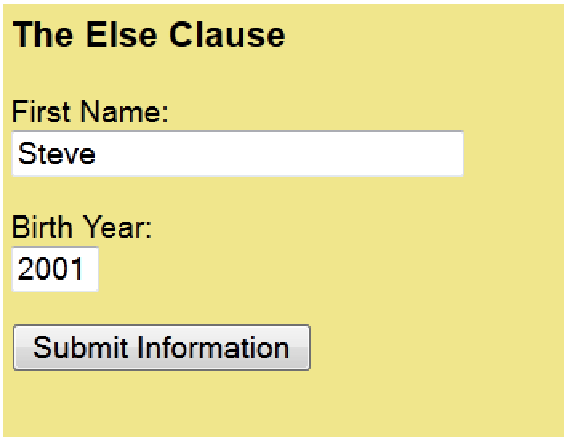
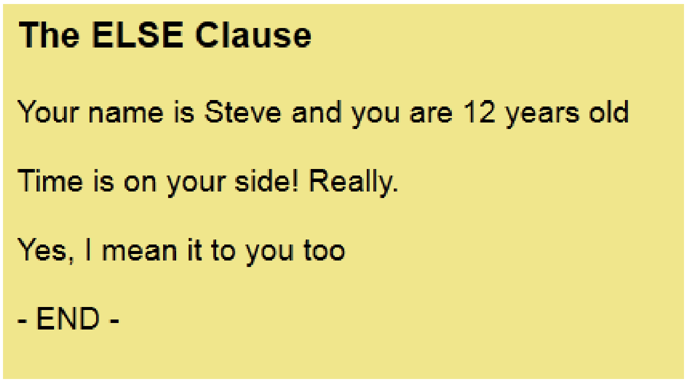


Figure 4.5.1 – Simple Web form

Here's what is returned when I click the Submit button…



The ELSE Clause

Your name is Steve and you are 12 years old

Time is on your side! Really.

Yes, I mean it to you too

- END -

Figure 4.5.2 – Simple Web form response

This time it says, "Your name is Steve, you're 12 years old".

When the age was not greater than 50 and we didn't do anything.

This time, I added some code to put the words "Time is on your side", etc on the response page.

The else clause says, "Well, what do I do when this comparison in the **if** statement is FALSE?" In other words, when the age is less than or equal to 50.

Instead of writing a whole another if statement, I put the key word "**else**" and some code in between a new set of curly braces below it.

Here's the code…

```
<body>

<h3>The ELSE Clause</h3>

<?php
     $firstname = $_POST['firstname'];
     $birthyear = $_POST['birthyear'];

     $current_year = date('Y');

     $age = $current_year - $birthyear;

     print "<p>Your name is $firstname and you
          are $age years old</p>";

     if ($age > 50)
     {
          print "<p>Better start planning
               for Retirement!</p>";
     }
     else
     {
          print "<p>Time is on your side!</p>";
     }

     print "<p> - END - </p>";

?>
</body>
```

Figure 4.5.3 – PHP code that processed the Web form

Now, notice how I have everything lined up very nicely, I've got the "I" for the if, then opening curly brace and closing braces lined up. The **else** clause is under the "I" as well.

Any code that's between a set of curly braces is called a PHP code block.

Now, the way this works is if age is not greater than 50 like when I'm 13, instead of dropping all the way down passed the final curly brace, it takes and else branch and displays "Time is on your side". And then get out of there.

4.6 The Nested IF Statement

Please view videos on class Web site for information on this topic.

4.7 Code Formatting Rules

Please view videos on class Web site for information on this topic.

4.8 Comparison Operators

Please view videos on class Web site for information on this topic.

4.9 Using AND

Please view videos on class Web site for information on this topic.

4.10 Using OR

Please view videos on class Web site for information on this topic.

4.11 Using NOT

Please view videos on class Web site for information on this topic.

4.12 Field Validations

Please view videos on class Web site for information on this topic.

4.13 Using DIVs

Please view videos on class Web site for information on this topic.

4.14 Using DIVs and Forms

Please view videos on class Web site for information on this topic.

4.15 Debugging Techniques

Please view videos on class Web site for information on this topic.