So PHP code is written for computer to understand what we want to do, so what about us? That is where comments come in. Comments are basically just a line or two that explains what's going on in a portion of your code. It can also be used to void out a portion of your code that you don't want to run. There are two types of comments: one for single line short comments, and another for longer comments that span multiple lines.

The short single-line comments are made by adding **//** in front of the comment. Multi-line comments are started with **/\*** and ended with **\*/**. Below are examples of both single and multi line comments:

// This is a comment

/\* This is a   
 comment that spans  
 multiple lines \*/

It’s a pretty simple concept, but it takes practice to get it right. You don’t want to necessarily put comments on every line, but maybe every few dozens of line explaining what that section of the code does. Take a look at the next example and try it out on your website:

<html>

<body>

<?php

//This prints “Hello World” to the screen.

echo "Hello World";

/\* This is another type of comment

that can span

multiple lines.\*/

?>

</body>

</html>

See how the single line comment explains what the line below it does? We will cover proper commenting in other tutorials later.

Note: Anything that is “commented out” will be ignored by your server at not run. For example, if you put **//** before the echo statement in the above code, nothing will be displayed on the screen.

2nd Note: You don’t need to put a semicolon(;) at the end of the comment.

Just printing out text with echo is not going to do us that good, because we can just as easily do that in HTML. The real power comes from something called variables. Just as you probably used the variable X for math equations. Just as you used them to store numbers, programming variables can store much more including text and sets of data. There are two types of variables that you need to know about right now, though there are many more:

* Integers – Stores numbers like 1, 39, 519.
* Strings – Stores text like “Hello World”, “I love PHP,” “WildPHP is cool.”

Unlike some of the other scripting languages, if you have programmed before, PHP automatically converts variables types for us. Take a look at the code below and then we will break it down.

<html>

<body>

<?php

//An integer that stores numbers

$integer = 1;

//An string that stores text

$string = "Hello World";

?>

</body>

</html>

The way variables work is you put the dollar sign ($) and then put the variable name after it. Something like this: $variablename. Now there are some rules that you must follow:

\* Must start with a letter or an underscore "\_"

\* A variable name can only contain alpha-numeric characters and underscores (a-z, A-Z, 0-9, and \_ )

\* A variable name should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($my\_string), or with capitalization ($myString)

We set the value of the variable by adding an equal sign (=) and putting the value we want to set it to on the right. Like the echo statement, we must end this with a semicolon (;). To make sure this works correctly lets print our variables by putting them after the echo statement.

<html>

<body>

<?php

//An integer that stores numbers

$integer = 1;

//An string that stores text

$string = "Hello World ";

//Print out the variables

echo $string;

echo $integer;

?>

</body>

</html>

This code should output “Hello Word 1.” If you are getting any errors, make sure that you have that semicolon, and your variable name does not violate any naming rules.

In the previous tutorial we stored a number in a variable and printed it. Let take this another level with PHP operators. Basically operators is just a technical way of saying adding , subtracting, multiplying, etc. **PLEASE NOTE THAT I HAVE REMOVED THE HTML TAGS. YOU STILL NEED THEM!**

<?php

//Adds 5+7. Variable value is 12.

$add = 5+7;

//Subtract 9-3. Variable value is 6.

$subtract = 9-3;

//Multiply 10\*10. Variable value is 100.

$multiply = 10\*10;

//Divide 32/2. Variable value is 18.

$divide = 32/2;

//Modulus(Also known as remainder) 10/3. Variable value is 1.

$modulus = 10%3;

?>

Pretty simple, huh? Well I can get even simpler. You can combine the operator with the equal sign to get what is called an assignment operator. You use it when you want to modify the existing variables value. You get the same result as the code above, but with less keystrokes.

<?php

//Adds 5+7. Variable value is 12.

$add += 5+7;

//Subtract 9-3. Variable value is 6.

$subtract = 9-3;

//Multiply 10\*10. Variable value is 100.

$multiply = 10\*10;

//Divide 32/2. Variable value is 18.

$divide = 32/2;

//Modulus(Also known as remainder) 10/3. Variable value is 1.

$modulus = 10%3;

?>