LAB 4: INTRODUCTION TO PHP

This lab is the fourth in a series. In this lab the student will learn about PHP. Students will learn about PHP syntax and features. The ultimate of objective of this lab is to teach students about PHP. At the end of this lab students will be required to complete a mini assignment.

IT 4530 Senior Capstone Project
Spring 2019
Georgia Southern University

Contents

Introduction

Goals

Objectives

Introduction to PHP

PHP stands for Hypertext Preprocessor. PHP is a scripting language and it can be used for many things. It is mainly used on web servers, as it is open source and free. We will be completing simple coding exercises so that you are able to learn the fundamentals of the programming language.

Goals

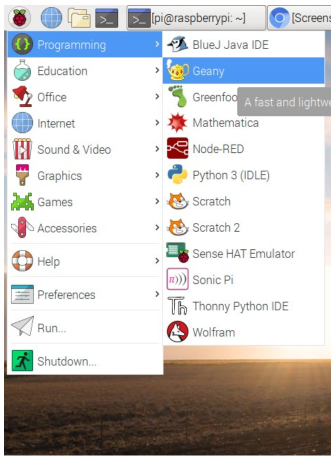
The goal of this lab is for the student to understand the basics of the PHP language. The student will be required to complete a lab to be turned in to the professor.

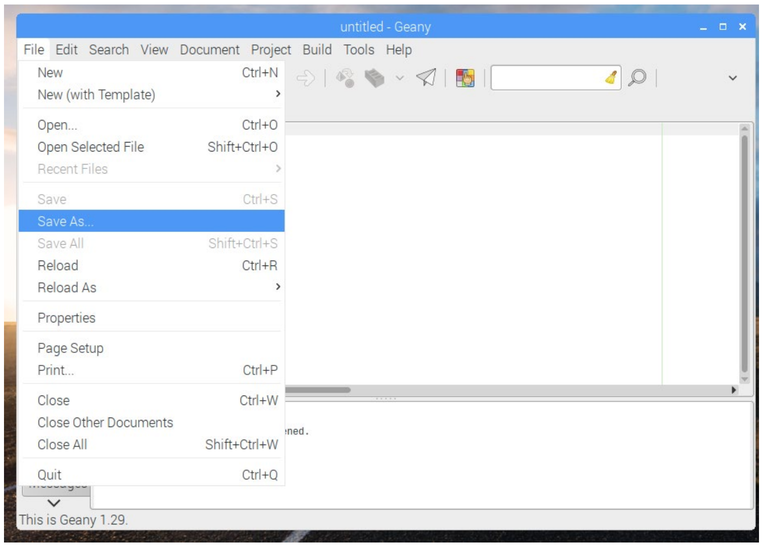
Objective

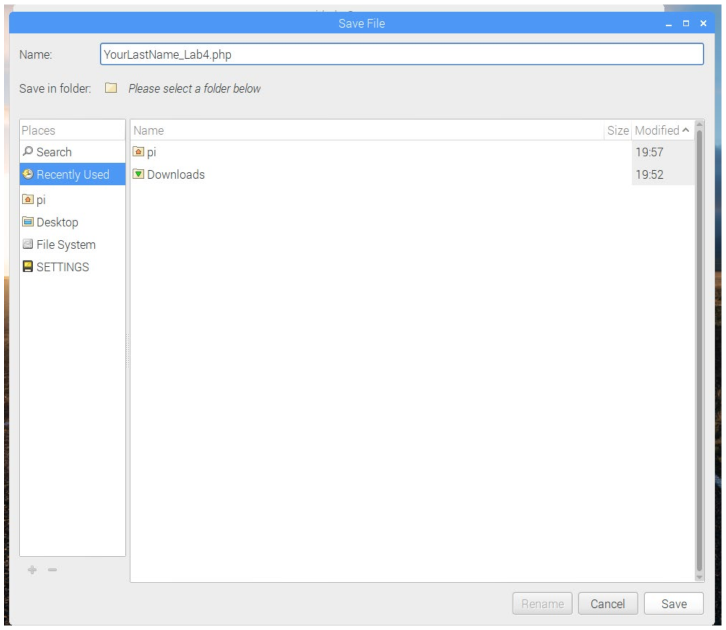
The objective of this lab is to teach the student the fundamentals of the PHP programming language.

File Naming Convention

First we need to name the file. For a PHP file to work correctly on a web server, it must have a .php extension at the end of the file. Open the Geany code editor under Programming under the Applications menu at their upper left corner of the screen, and in the upper left corner Click “File”, then click “Save As”, and navigate to where you would like to save your file. Please name your file as *YourLastName*_Lab4.php with *YourLastName* being your actual last name. Click save. Your file should now be saved on your preferred storage device.



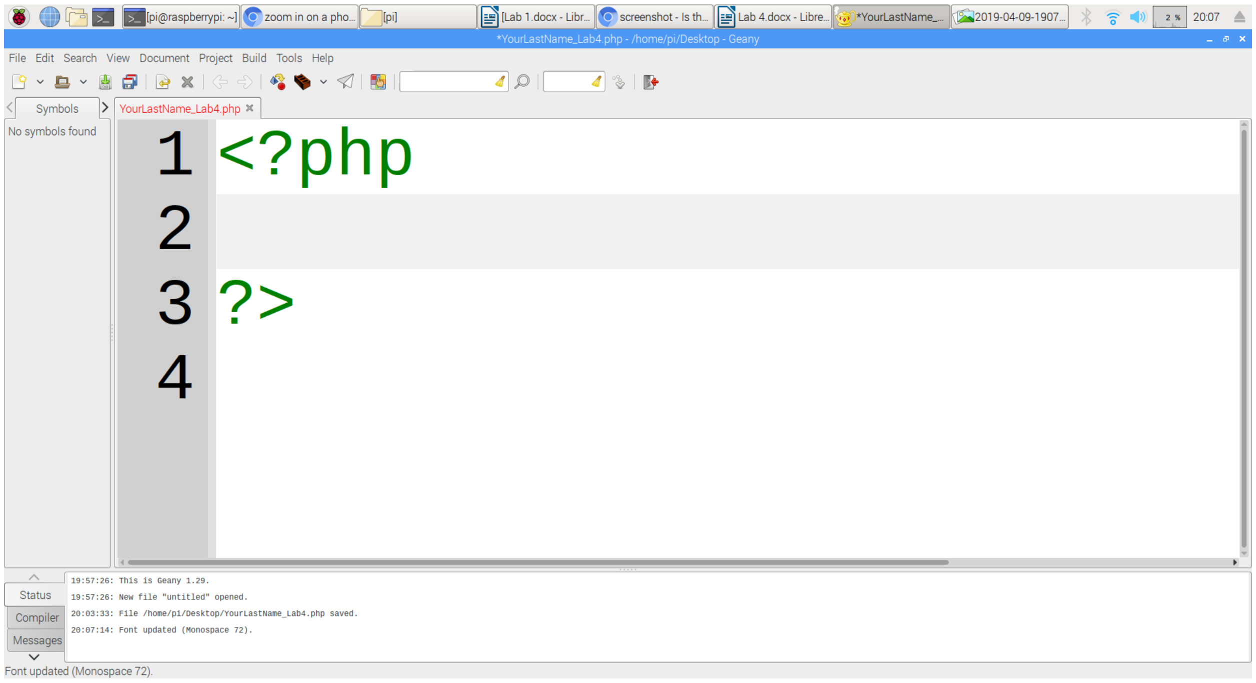




Before we can start coding, we must insert certain code for the php file to work correctly. A php file must always have an opening and closing statement. The opening statement is **`<?php`** and the closing statement is **?>**. These two statements must be present in every file and all code must be placed inside of the two statements. Please insert your opening and closing statement in the file you just created. (Figure 1). You may enter the statements as close or as far apart as you like. However, a good coding practice is for you to insert them close together.

Your file should look similar to the one below. The font is enlarged in order for you to see the code. Your font does not have to be the same size.

Figure 1



*YourLastName_Lab4.php - /home/pi/Desktop - Geany
File Edit Search View Document Project Build Tools Help
Symbols
No symbols found
YourLastName_Lab4.php
Status
Compiler
Messages

```
<?php

?>
```

```
19:57:26: This is Geany 1.29.
19:57:26: New file "untitled" opened.
20:03:33: File /home/pi/Desktop/YourLastName_Lab4.php saved.
20:07:14: Font updated (Monospace 72).
```

Font updated (Monospace 72).

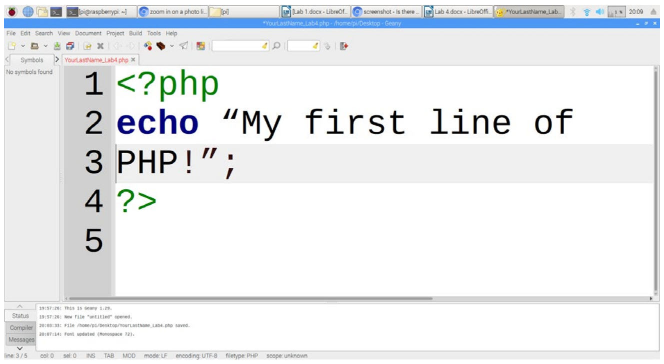
For easier reference the code is also listed below:

```
<?php
?>
```

Echo Statement

We will now begin to add code to the file. Echo is a function that is used to print messages out to the user so they are able to see it. You use parentheses to place the text you would like to print to the user. You may use either double quotes (“) or single quotes (‘) to place text inside of. However, you must be consistent with your quotes or otherwise your code will not work. Now, inside of the opening statements please type: `echo “My first line of PHP!”;` (Figure 2). This is considered a **string**. In PHP, a string is anything that is placed inside of quotes. As you will notice, what we type in the parentheses is what will print out to the user. You may notice at the end of the line we placed a semicolon (;). This is to let the server know this is the end of a line. If you fail to place a semicolon at the end of a line, you code will produce an error and the code will not run in a server. To ensure that your code is running correctly throughout this lab we will place sections of your code into a validator. In a web browser, please proceed to **phpfiddle.org.** PHP Fiddle is a free web site to help you check your code and ensure that it is working correctly. Place your first line of code into the validator and click run. If there are any errors, the validator will produce error messages of what line your error is on and why it is producing an error (Figure 4). In Figure 5 the error produced is because we forgot to include a closing quote.

Figure 2

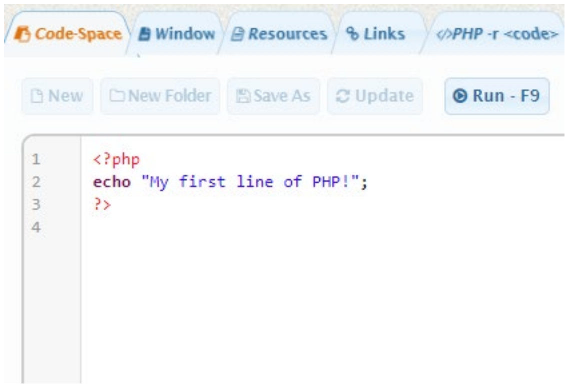


```
<?php
echo “My first line of
PHP!”;
?>
```

```
19:57:26: This is Geany 1.29.
19:57:26: New file "untitled" opened.
20:03:33: File /home/pi/Desktop/YourLastName_Lab4.php saved.
20:07:14: Font updated (Monospace 72).
```

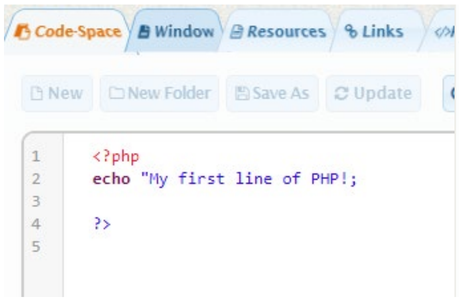
```
line: 3 / 5   col: 0   sel: 0   INS   TAB   MOD   mode: LF   encoding: UTF-8   filetype: PHP   scope: unknown
```

Figure 3

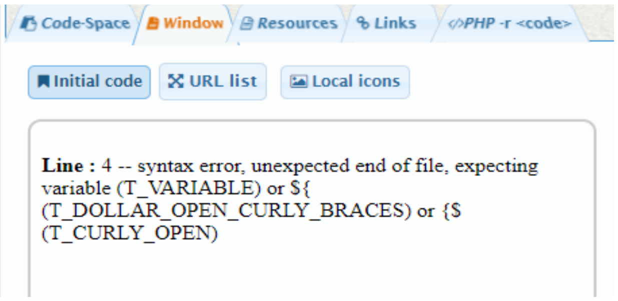


```
<?php
echo "My first line of PHP!";
?>
```

Figures 4 & 5



```
<?php
echo "My first line of PHP!;

?>
```



Line : 4 -- syntax error, unexpected end of file, expecting variable (T_VARIABLE) or ${ (T_DOLLAR_OPEN_CURLY_BRACES) or {$ (T_CURLY_OPEN)

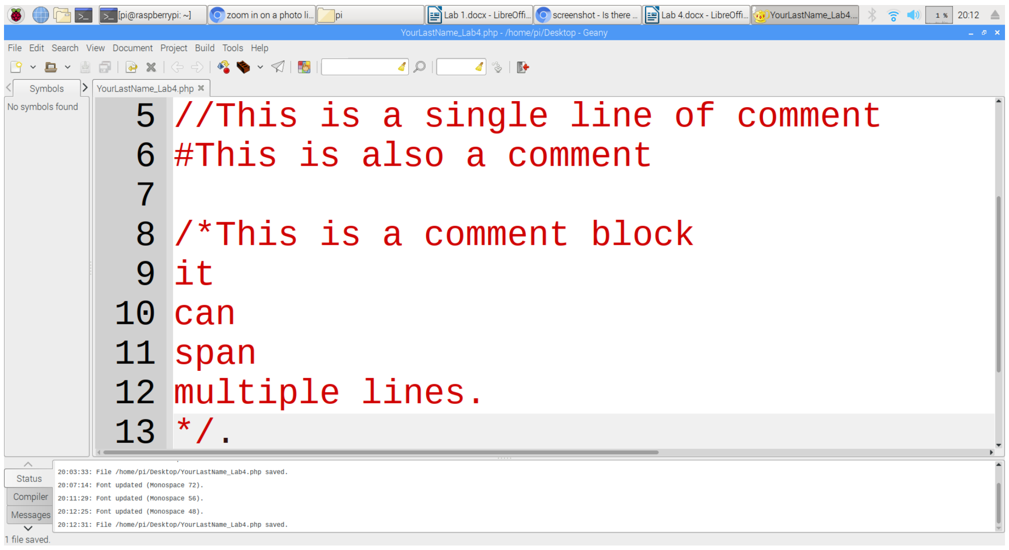
Comments

In any programming language it is important for you to be able to insert comments for both you and others to be able to understand the code. In PHP you can insert comments into your code in two different ways. You can either insert comments using double forward slashes (`//`) or using combined forward slash (`/`) and Asterisk (`*`). The double forward slashes (`//`) are for single line comments. The combined forward slash (`/`) and Asterisk (`*`) are for multi-line comments. You can also use a single pound (`#`) sign to insert a comment. Comment lines will be shown in green, and anything in comments will not run in the program, including code. Comments can be helpful in debugging your code by disabling code. In your file under the echo statement enter the following code: `//This is a single line of comment.` Under that line also enter the following: `#This is also a comment.` Comments can also span multiple lines. Enter the following code into your file:

```
/*This is a comment block
it
can
span
multiple lines.
*/.
```

Comments that span multiple lines must be formatted correctly. Always start multi line comments with a forward slash (/) and Asterisk (*) next to each other (/*). Ensure you close the comment block correctly with an Asterisk (*) and forward slash (/). The closing part must be (*/) with the Asterisk first. Consult Figure 6 for a visual reference.

Figure 6


YourLastName_Lab4.php - /home/pi/Desktop - Geany
File Edit Search View Document Project Build Tools Help
Symbols
YourLastName_Lab4.php
No symbols found
Status
Compiler
Messages

```
//This is a single line of comment
#This is also a comment

/*This is a comment block
it
can
span
multiple lines.
*/.
```

```
20:03:33: File /home/pi/Desktop/YourLastName_Lab4.php saved.
20:07:14: Font updated (Monospace 72).
20:11:29: Font updated (Monospace 56).
20:12:25: Font updated (Monospace 48).
20:12:31: File /home/pi/Desktop/YourLastName_Lab4.php saved.
```

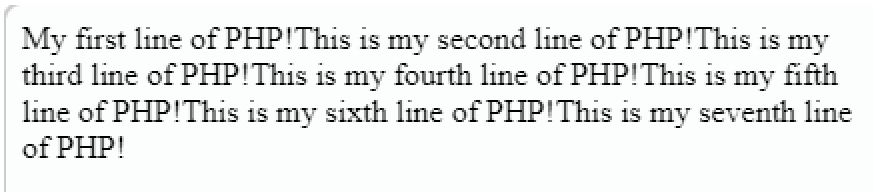
1 file saved.

Break Tags

In PHP a compiler will output your code exactly as it reads it. This means your code may look like it ends on every line but the compiler will bunch up your code making it difficult to read. An example of this is shown in Figure 7. With break tags (<br>) you can separate individual lines so that it is easier for the user to read. To add break tags insert `echo "<br>";` after each line you wish to be separated. This helps organize and separate lines for the user to be able to read. After inserting break tags the output will look similar to Figure 8. After your comment block please enter a break tag.

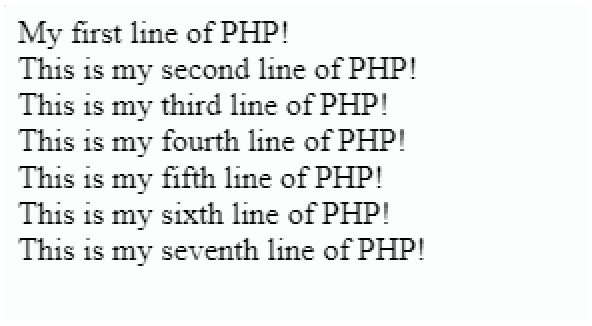
```
echo "<br>";
```

Figure 7



```
My first line of PHP!This is my second line of PHP!This is my
third line of PHP!This is my fourth line of PHP!This is my fifth
line of PHP!This is my sixth line of PHP!This is my seventh line
of PHP!
```

Figure 8

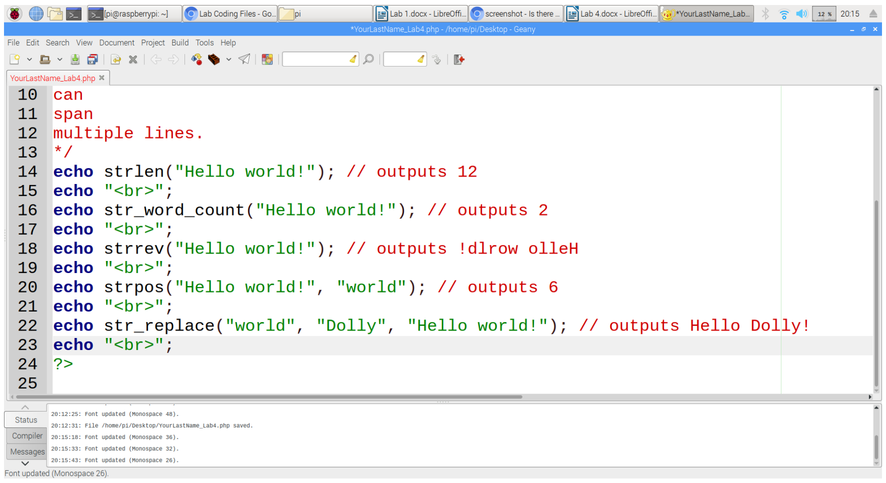


```
My first line of PHP!
This is my second line of PHP!
This is my third line of PHP!
This is my fourth line of PHP!
This is my fifth line of PHP!
This is my sixth line of PHP!
This is my seventh line of PHP!
```

String Functions

PHP includes numerous string functions that we can use. String functions always work with strings. A string is anything placed inside of quotes in PHP. The first line of code you inserted was a string, the "My first line of PHP" code. With string functions, you can do anything from counting the number of characters in a string, to replacing the letters, to even reversing characters in a string. We will now insert more strings. After the break tag please enter `echo strlen ("Hello world!);.` This string function will output the length of the string, in this case 12 characters. Enter a break tag (<br>) using an echo statement. Now enter `echo str_word_count ("Hello world!");`. This string function outputs the number of words in a string, in this case 2. You can also reverse a string. Enter a break tag (<br>) using an echo statement. If you enter echo `strrev("Hello World!");` the compiler will produce a mirror image of the string you have input. PHP also allows you to find and search for characters in a string. Enter a break tag (<br>) using an echo statement. You can use `echo strpos` to search for words or characters in a string. In your file enter `echo strpos("Hello World!". "world");`. In this code, you are determining the location of world in the string. The compiler will output 6 because that is the instance within the code where world begins. The strpos function does not count any spaces. The stringpos function always starts counting from the left. Now enter a break tag (<br>) using an echo statement. You can also replace words in a string with the string function str_replace. In your code enter echo str_replace("world", "Dolly", "Hello World!");. This function will replace the word "world" with the word "Dolly". In this function you insert the word you want to change first in quotes, and the word you want to replace the word with second in quotes. Enter a break tag (<br>) using an echo statement. Your code should look similar to Figure 9. Your code output should look similar to Figure 10.

Figure 9

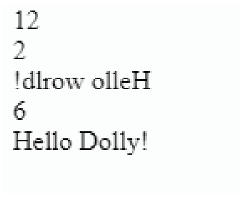

[pi@raspberrypi: ~]
Lab Coding Files - Go...
pi
Lab 1.docx - LibreOffi...
screenshot - Is there ...
Lab 4.docx - LibreOffi...
*YourLastName_Lab...
12 %
20:15
*YourLastName_Lab4.php - /home/pi/Desktop - Geany
File Edit Search View Document Project Build Tools Help
YourLastName_Lab4.php
Status
Compiler
Messages

```
can
span
multiple lines.
*/
echo strlen("Hello world!"); // outputs 12
echo "<br>";
echo str_word_count("Hello world!"); // outputs 2
echo "<br>";
echo strrev("Hello world!"); // outputs !dlrow olleH
echo "<br>";
echo strpos("Hello world!", "world"); // outputs 6
echo "<br>";
echo str_replace("world", "Dolly", "Hello world!"); // outputs Hello Dolly!
echo "<br>";
?>
```

```
20:12:25: Font updated (Monospace 48).
20:12:31: File /home/pi/Desktop/YourLastName_Lab4.php saved.
20:15:18: Font updated (Monospace 36).
20:15:33: Font updated (Monospace 32).
20:15:43: Font updated (Monospace 26).
```

Font updated (Monospace 26).

Figure 10

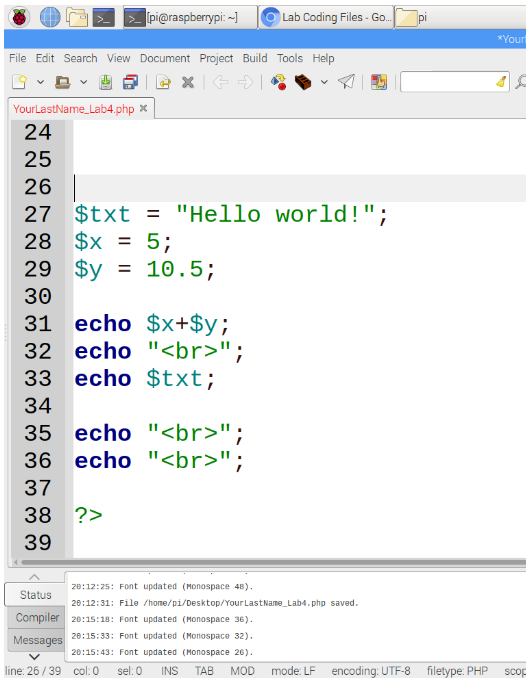


```
12
2
!dlrow olleH
6
Hello Dolly!
```

Variables

PHP also allows you to create variables inside the code. Variables allow you to store information. These variables can be utilized to store anything from numbers, to entire strings. Variables make coding in PHP easier because you can substitute the variable instead of coding out all of your information. All variables in PHP start with a dollar sign ($). You assign information to a variable using an equal sign (=). You can name your variables anything but a good coding practice is to name them something relevant to your code or use single letters. In your code on a new line enter `$txt = "Hello World!";`. On the next line enter `$x = 5;`, and on one more line enter `$y = 10.5;`. You do not have to enter a break tag with variables as they will not be visible to the user, only you can see variables that are defined. On a new line enter echo `$x+y`;. In this instance we are using stored variables to add two numbers. With variables this is simplified as we can use these variables over and over again. The output of these two variables should be 15.5. Enter a break tag (<br>) using an echo statement. In your code enter `echo $txt`. This will out put the information stored in the variable txt. As you can see, variables can be useful in coding information that needs to be coded continuously, such as a name or phone number. Now please enter two separate break tags (<br>) using an echo statement on separate lines. Your code should look similar to Figure 11. Your code output should look similar to Figure 12.

Figure 11

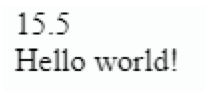


```


$txt = "Hello world!";
$x = 5;
$y = 10.5;

echo $x+$y;
echo "<br>";
echo $txt;

echo "<br>";
echo "<br>";

?>

```

```
20:12:25: Font updated (Monospace 48).
20:12:31: File /home/pi/Desktop/YourLastName_Lab4.php saved.
20:15:18: Font updated (Monospace 36).
20:15:33: Font updated (Monospace 32).
20:15:43: Font updated (Monospace 26).
```

[pi@raspberrypi: ~]
Lab Coding Files - Go...
pi
*Your
File Edit Search View Document Project Build Tools Help
YourLastName_Lab4.php
Status
Compiler
Messages
line: 26 / 39 col: 0 sel: 0 INS TAB MOD mode: LF encoding: UTF-8 filetype: PHP scop

Figure 12

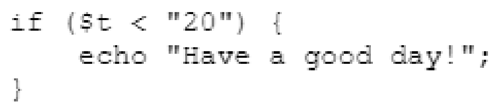


```
15.5
Hello world!
```

If Statements

In PHP you can use logic statements to determine what will be output to the user. We will be using a logical if statement to output certain information to the user at different times of the day. We will be using a predefined function, the date function within PHP. In your code define a variable with the current hour using `$t = date(“H”);`. Ensure you are using a capital “H” inside the date function as a capital H is used for 24-hour format, and a lower case “h” is used for 12-hour format. In any coding language it is good practice to use 24-hour format. We will now enter the if statement. If statements include opening and closing braces. It is very important for you to close your braces correctly as if you do not, it can wreak havoc on other parts of your code. The code may also not run if you do not close the braces correctly. In your code please enter the code exactly as shown below in Figure 13 as it is a good practice to enter the code below as shown in Figure 13. This code will output the message “Have a good day” if the hour is less than 20 (8 P.M.). Right now if the hour is more than 20 nothing will be output to the user. We can solve this with an If Else statement.

Figure 13



```
if ($t < "20") {
    echo "Have a good day!";
}
```

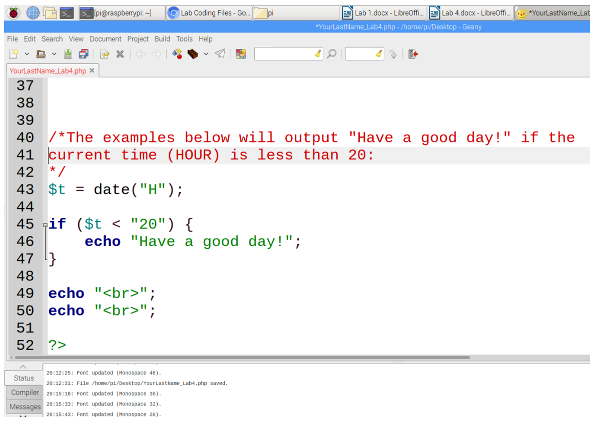
If Else Statements

Sometimes an If statement is not enough in a program. You may need to add more functionality if there are more variables to a task. In the previous example we used an If statement to only output “Have a good day” if the hour was less than 20 (8 P.M.). We will now add more variables with an If Else statement. First please insert two separate break tags on separate lines after your previous code as shown in Figure 14. We will now add the If Else statements. On a new line under the Break Tags insert the following code:

```
if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
```

Please ensure you correctly place and match your brackets, otherwise the code will produce errors and not run. In this code, if the hour is less than 20 (8 P.M.) the code will output “Have a good day!”. If the hour is more than 20 the code will output “Have a good night!”. Your code should look similar to Figure 15. Your If statement output should look similar to Figure 16. Your else statement should look similar to Figure 17. Your $t variable should have already been defined earlier. If not please define now so that the code will not produce errors. You can define it by typing this code: $t = date(“H”);.

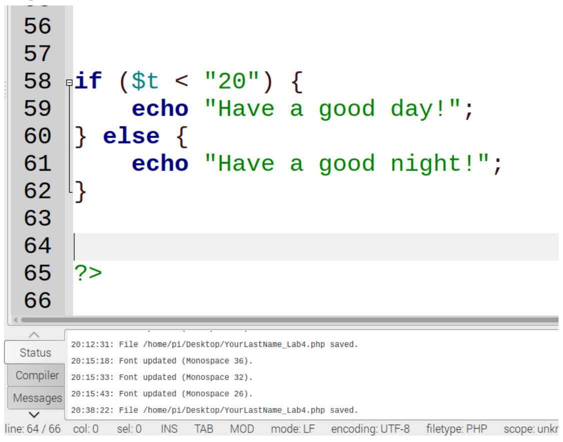
Figure 14


[pi@raspberrypi: ~]
Lab Coding Files - Go...
pi
Lab 1.docx - LibreOffi...
Lab 4.docx - LibreOffi...
*YourLastName_Lab
*YourLastName_Lab4.php - /home/pi/Desktop - Geany
File Edit Search View Document Project Build Tools Help
YourLastName_Lab4.php
Status
Compiler
Messages

```



/*The examples below will output "Have a good day!" if the
current time (HOUR) is less than 20:
*/
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}

echo "<br>";
echo "<br>";

?>
```

```
20:12:25: Font updated (Monospace 48).
20:12:31: File /home/pi/Desktop/YourLastName_Lab4.php saved.
20:15:18: Font updated (Monospace 36).
20:15:33: Font updated (Monospace 32).
20:15:43: Font updated (Monospace 26).
```

Figure 15



```


if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}


?>

```

```
20:12:31: File /home/pi/Desktop/YourLastName_Lab4.php saved.
20:15:18: Font updated (Monospace 36).
20:15:33: Font updated (Monospace 32).
20:15:43: Font updated (Monospace 26).
20:38:22: File /home/pi/Desktop/YourLastName_Lab4.php saved.
```

line: 64 / 66 col: 0 sel: 0 INS TAB MOD mode: LF encoding: UTF-8 filetype: PHP scope: unkr

Figure 16

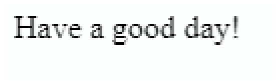
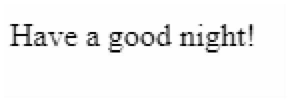

Have a good day!

Figure 17


Have a good night!

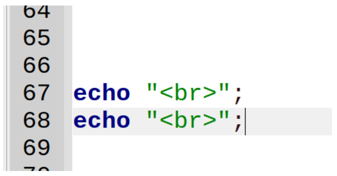
Else If Statement

Sometimes you need even more than an If Else statement, you also need to include an Else If statement. These are usually reserved for multi part coding statements. You can have as many Else If statements as you would like in you code to help you with solving a coding problem. In your code on separate lines place two more break tags as shown in Figure 18. On a new separate line place the following code:

```
if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
```

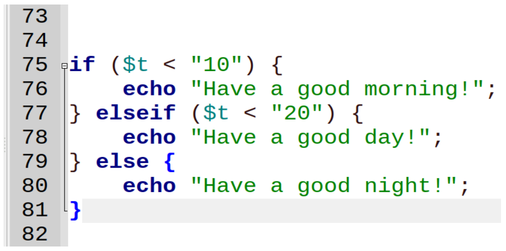
This code has three separate parts. The first part will produce the output “Have a good morning!” if the hour is less than 10. The second part, the elseif will produce an output of “Have a good day!” if the hour is less than 20. The third part, the else statement will produce an output of “Have a good night!” if the hour is more than 20. It is important to mention that you can change the hours or the output of any part of these If statements. Just remember that you need to change it back to the original before submitting. Your code should look similar to Figure 19. Ensure that your brackets are placed and closed correctly; otherwise your code will produce errors and not run. Your If statement output should look similar to Figure 20. Your elseif statement output should look similar to Figure 21. Your else statement output should look similar to Figure 22.

Figure 18



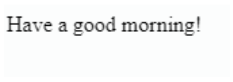
```
echo "<br>";
echo "<br>";
```

Figure 19



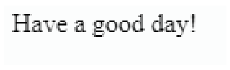
```
if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
```

Figure 20



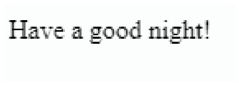
Have a good morning!

Figure 21



Have a good day!

Figure 22



Have a good night!

Functions

Functions can be useful in PHP, as they are stored instructions. You can put code into a function once and then call it numerous times. Functions are massive time savers in coding. We will now create a simple function titled "writeMsg()". When creating your function, you always choose a name that is relevant. If the function is more than one word, you put them together as in “writeMsg”. The next step is to add opening and closing parentheses “()”. You can always place an argument inside the parentheses; we will do that in the next section. The next step is to add opening and closing braces. Your function so far should look similar to Figure 23. We will now add the coding inside the function for the function to run. Inside the braces add the following code:

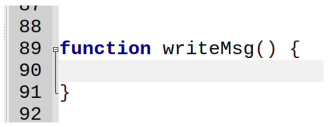
```
echo "I'm writing this on a Raspberry Pi!";
```

Now that we have code inside the function, it still will not run. Your code should look similar to Figure 24. You have to call the function for it to run. On a new line enter the following code after the closing braces:

```
writeMsg();
```

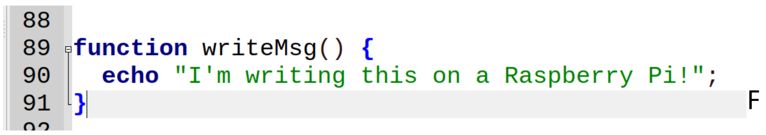
Your code should look similar to Figure 25. Your output should look similar to Figure 26.

Figure 23



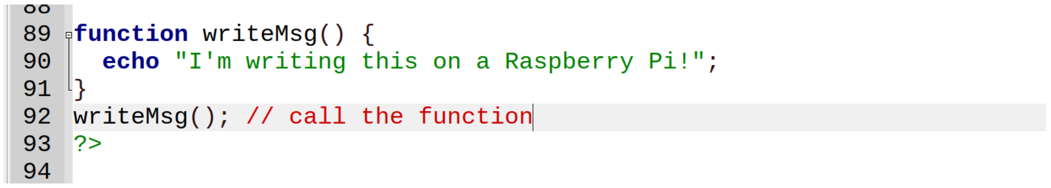
```
function writeMsg() {

}
```

Figure 24



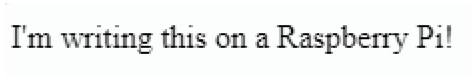
```
function writeMsg() {
  echo "I'm writing this on a Raspberry Pi!";
}
```

Figure 25



```
function writeMsg() {
  echo "I'm writing this on a Raspberry Pi!";
}
writeMsg(); // call the function
?>
```

Figure 26



```
I'm writing this on a Raspberry Pi!
```