Episode 9

Functions

Introduction

A function is a block of reusable code that can be executed by being called from another part of the script. Functions help keep your code simple and organized. Instead of having to write a block of code that performs a particular task in several places within your script, you can write it once and call it whenever needed.

Goals

In this section of the course your goals are:

- [ ] To learn how to define and call a function.
- [ ] To learn how to use function arguments.
- [ ] To learn about the return keyword.
- [ ] To learn about variable scope.

Defining and Calling a function

Before you can call a function to be executed, you must define it first. The definition of a function doesn't have to happen before it is called, except when the definition is done inside a conditional statement.
Let's now look at an example:

```
<?php
function myMessage () {
    echo "My name is Julio Quevedo.<br>";
    echo "I am a PHP Developer.<br>";
}
?>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Functions</title>
        <style> body {margin: 30px;}</style>
    </head>
    <body>
        <h3>
        <?php
        myMessage();
        ?>
        </h3>
    </body>
</html>
```

The definition of a function starts with the **function** keyword followed by the name of the function. Function names are **not** case sensitive, which, in our example means that you can call **myMessage( )** by writing **mymessage( )** or **MYMESSAGE( )**. Going back to code writing conventions, function names are usually written in **lowerCaseCamel( )**, but it may vary from company to company. It is also a best practice to name a function to reflect the task that it performs.
After the name of the function comes a set of parentheses which may have zero or more arguments. The function in our example doesn't have any arguments. Next is a set of curly brackets. Everything inside these curly brackets is part of the function, and it will be executed when the function is called. Finally, to call a function, you simply have to write the name of the function followed by a set of parenthesis.

Function Arguments

In the last example, we saw a function with no arguments. Functions can also be called with any number of arguments. Let's take a look at the next example:

```php
function sum ($s1, $s2) {
    echo $s1 + $s2;
}

function ulist ($t, $l1, $l2, $l3) {
    echo "This is your list of $t:";
    echo "<ul>".
         "<li>$l1</li>".
         "<li>$l2</li>".
         "<li>$l3</li>".
         "</ul>";
}

$var1 = 4;
$var2 = 8;
sum ($var1, $var2);

$listTitle = "groceries";
$item1 = "milk";
$item2 = "bread";
$item3 = "eggs";
ulist ($listTitle, $item1, $item2, $item3);
```

The function **sum** takes two arguments and uses an echo statement to display their sum on the screen. The **ulist** function takes four arguments. The first is the title for a list, and the next three are the list items. Notice how we were able to create an unordered list with this function.

Using the Return keyword

In the last examples, we have used functions to do something for us and then display the result on the screen. We can also use the **return** keyword to get the result back from the function to our main program. Let’s modify the last two examples to do that:

```
function sumR ($s1, $s2) {
    return $s1 + $s2;
}

function ulistR ($t, $l1, $l2, $l3) {
    $output = "This is your list of ".$t.":".
              "<ul>".
              "<li>".$l1."</li>".
              "<li>".$l2."</li>".
              "<li>".$l3."</li>".
              "</ul>";
    return $output;
}

$mySum = sumR ($var1, $var2);
echo $mySum."<br><br>";

$myList = ulistR ($listTitle, $item1, $item2, $item3);
echo $myList."<br><br>";
```

In these cases the functions perform a task and send back the results to us. Doing it this way gives us better control over what to display on the screen and perhaps use those results in more than one place.

Variable Scope

Where you declare a variable makes a difference as to where you can use that variable in your program. Variables created inside a function can only be accessed by the function in which it was created. Conversely, a variable created outside any function cannot be accessed by those functions. Let's look at the following example:

```
function addThree ($number) {
    $number = $number + 3;
    echo "I am the number from the function: ".$number."<br>";
}

$number = 5;
addThree ($number);
echo "I am the number from the main script: ".$number."<br>";
```

The echo statement inside the addThree function will output 8. However, the echo statement outside the function will output 5. This is because the $number variable inside the function and the $number variable outside the function are two separate variables. The scope of the $number variable declared in the function is only within the function itself. Whereas the scope of the $number variable declared outside the function is only outside any functions.

Debugging Code

Examine the following code. Find and fix the errors.

```
<?php
$number = 100;

function piApprox($number){
    $t = 0;
    for($i = 1; $i <= $n; $i++){
        $t += ((-1)**($i+1))/(2 * $i - 1);
    }
    return 4 * $t;
}
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Debugging</title>
        <style>body{margin: 30px;}</style>
    </head>
    <body>
        <?php
        echo piApprox($number);
        ?>
    </body>
</html>
```

Lab Exercises

1. Write a function that will take three numbers and will display them in increasing order.
2. Write a function that will take three arrays of strings, and will display an HTML table with the first array as headers and the second and third as data.
3. Write two functions, one that will convert Celcius to Fahrenheit and one that will convert Fahrenheit to Celcius and use their return values to display the following HTML table:

Fahrenheit	Celcius	Celcius	Fahrenheit
-40	-40	-40	-40
-30	-34.44	-35	-31
-20	-28.89	-30	-22

Fahrenheit	Celcius	Celcius	Fahrenheit
-10	-23.33	-25	-13
0	-17.78	-20	-4
...	...	...	...
...	...	...	...
110	43.33	35	95
120	48.89	40	104