**Exercise 02\_01\_01 – Step 1**

In this Exercise, we will learn how to code the fundamentals of PHP.

 

1. Create a folder name Exercise 02\_01\_01 and open it with your IDE. Go to ***Exercise 01\_06\_01***, copy the file ***modernizr.custom.65897.js*** and paste it into your new folder.
2. Create a new file called ***PHPCodeBlocks.php***. Scaffold a basic HTML code layout into it, such as the following code, making sure to have the ***modernizr*** <script> linked in. In the <body>, create an ***<h1>*** element for testing::  
   ***<!DOCTYPE html>  
   <html lang="en">  
     
   <head>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width">  
    <title>PHP Code Blocks</title>  
    <script src="modernizr.custom.65897.js"></script>  
   </head>  
     
   <body>  
    <h1>Hello, world!</h1>   
   </body>  
     
   </html>***
3. Go to ***Exercise 01\_06\_01***, and open the file snoot.html. Copy the documentation that is just below the opening <head> element, and paste it below the opening <head> element of your new PHP file. Modify and complete the documentation as appropriate:  
   <head>  
    ***<!--  
    Exercise 02\_01\_01  
     
    Author: Mark J. Buckler  
    Date: august 26, 2017  
     
    Filename: PHPCodeBlocks.php  
    -->***  
   Copy the project folder into the appropriate spot on your Web Server and test it.
4. Open the Developer Tools in your browser. Go to the Network tab and check ***Disable cache***. We will do this for development only, so we can keep copying new files into the server and be sure the new versions are the ones that run.
5. In the PHP file, remove the ***<h1>*** debug element. Add a ***<p>*** element inside the body. Inside it, create a ***PHP code block***, using standard PHP script delimiters:  
    ***<p>  
    <?php  
    ?>  
    </p>***  
   Give this a syntax test on the web server, using the Console tab in your browser.
6. Add a PHP ***echo*** statement into the PHP code block as follows:  
    ***echo "This text is displayed using standard PHP  
    script delimiters. ";***Give this a server test.

**Exercise 02\_01\_01 – Step 2**

 

1. Create a new file named ***PHPTest.php***. Copy the contents of ***PHPCodeBlocks.php*** into it. Remove the ***echo*** statement and replace it with the following function call:  
    <?php  
    ***phpinfo();*** ?>  
   Run a server test on this and you will learn a little bit about your PHP installation.

**Exercise 02\_01\_01 – Step 3**

 

1. Return to the file ***PHPCodeBlocks.php*** into it. Below the ***echo*** statement, enter the code for a ***print*** statement:   
    echo "This text is displayed using standard PHP  
    script delimiters.";  
    ***print "Second string";***  
   Run a server test on this. Notice that there is no new line between the two outputs.
2. Place an HTML ***<br>*** element inside each of the string literals, at the end of each:  
    echo "This text is displayed using standard PHP  
    script delimiters***.<br>***";  
    print "Second string***<br>***";  
   Run a server test on this. Notice that there is now a new line between the two outputs.
3. Below the ***print*** statement, enter in a function style print statement as follows:  
    ***print ("Third string<br>");***  
   Run a server test on this.
4. Below the ***print*** statement, enter in a ***multi-parameter*** echo statement as follows:  
    print "Second string<br>";  
    ***echo "<h1>This is a" , " multiple argument string.</h1>";***  
   Run a server test on this, and everything should be okay.
5. Now change the previous echo statement to function style by adding parentheses:  
    ***echo ("<h1>This is a" , " multiple argument   
    string.</h1>");***  
   Run a server test on this, we can see that function style cannot take ***comma-separated*** parameters. Fix the error by removing the parentheses.
6. Below the fixed ***echo*** statement, enter some function style code that uses concatenation as follows:  
    ***echo("<h1>This is a" . " concatenated argument string.</h1>");***  
   Run a server test on this, and everything should be okay.
7. Below the new ***echo*** statement, enter some ***print*** function code that uses ***concatenation*** and does some ***math*** as follows:  
    ***print("<h2>This is some math: " . (2 + 3) . "</h2>");***  
   Run a server test on this.

**Exercise 02\_01\_01 – Step 4**

 

1. Create a new document in your IDE named ***MultipleScripts.php***. Scaffold an HTML template into it. Set the ***<title>*** to be ***PHP Environment Info***. Link in a ***modernizr*** script:   
   ***<!doctype html>  
   <html>  
   <head>  
    <title>PHP Environment Info</title>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="initial-scale=1.0">  
    <script src="modernizr.custom.65897.js"></script>  
   </head>  
     
   <body>  
     
   </body>  
   </html>***Run a server test on this.
2. Place an HTML ***<h1>*** element and content inside the document <body>. Also add a <p> element and content. After the content, place PHP script delimiters, with a function call inside them:  
   <body>  
    ***<h1>PHP Environment Info</h1>  
    <p>This page was rendered with PHP version  
    <?php  
    echo phpversion();  
    ?>  
    </p>***</body>  
   Run a server test on this.
3. Below the previous code, before the closing </body>, place the following HTML element, content, and PHP code block inside the document:  
    ***<p>The PHP code was rendered with Zend Engine version  
    <?php  
    echo zend\_version();  
    ?>  
    </p>***Run a server test on this.
4. Finally, below all of the previous code, before the closing </body>, place the following HTML elements, content, and PHP code blocks inside the document:  
    ***<p>PHP's default MIME type is  
    <?php  
    echo ini\_get("default\_mimetype");  
    ?>  
    <p>The maximum allowable execution time of a PHP script   
    is  
    <?php  
    echo ini\_get("max\_execution\_time");  
    ?>  
    seconds.  
    </p>***Run a server test on this.

**Exercise 02\_01\_01 – Step 5**

 

1. Create a new file called ***HelloWorld.php***. Scaffold a basic HTML code layout into it, such as the following code, making sure to have the ***modernizr*** <script> linked in. In the <body>, create an ***<h1>*** element for testing::  
   ***<!DOCTYPE html>  
   <html lang="en">  
   <head>  
    <meta charset="utf-8">  
    <meta name="viewport" content="width=device-width">  
    <title>Hello World</title>  
    <script src="modernizr.custom.65897.js"></script>  
   </head>  
     
   <body>  
     
   </body>  
   </html>***
2. Open one of your other code files and copy the documentation that is just below the opening <head> element, and paste it below the opening <head> element of your new PHP file. Modify and complete the documentation as appropriate:  
   <head>  
    ***<!--  
    Exercise 02\_01\_01  
     
    Author: Mark J. Buckler  
    Date: September 3, 2017  
     
    Filename: HelloWorld.php  
    -->***
3. Add standard PHP script ***delimiters*** to the document body:  
   <body>  
    ***<?php  
     
    ?>***
4. In the code block, ***declare*** the following ***variables*** containing names of celestial bodies and scientific information about each:  
    ***<?php  
    $worldVar = "World";  
    $sunVar = "Sun";  
    $moonVar = "Moon";  
    $worldInfo = 92897000;  
    $sunInfo = 72000000;  
    $moonInfo = 3456;  
    ?>***Run a server test on this, with browser Developer Tools on the Console tab to check syntax.
5. Add the following statements to the end of the code block. These should display the values stored in each of the variables:  
    ***echo "<p>Hello $worldVar!<br>";  
    echo "The $worldVar is $worldInfo miles from the   
    $sunVar.<br>";  
    echo "Hello ", $sunVar, "!<br>";  
    echo "The $sunVar's core temperature is approximately   
    $sunInfo   
    degrees Fahrenheit.<br>";  
    echo "Hello ", $moonVar, "!<br>";  
    echo "The $moonVar is $moonInfo miles in diameter.</p>";***Run a server test on this.
6. Let’s test out the use of ***curly*** ***braces*** when embedding variables in double quoted strings. Add the following statements to the end of the code block:  
    ***echo "This is a {$moonVar}embedded in double quotes";  
    echo "This is a $moonVarembedded in double quotes";***Run a server test on this. This should clearly show why we have to use curly braces in some instances.

**Exercise 02\_01\_01 – Step 6**

 

1. Copy the file ***HelloWorld.php*** to a new file called ***HelloWorld2.php***. Update the documentation to indicate the new file name.
2. Replace the variable declarations for ***$worldInfo***, ***$sunInfo***, and ***$moonInfo*** with ***constant*** ***definitions***:  
    ***define("WORLD\_INFO", 92897000);  
    define("SUN\_INFO", 72000000);  
    define("MOON\_INFO", 3456);***
3. Replace the variables ***$worldInfo***, ***$sunInfo***, and ***$moonInfo*** in the ***echo*** statements with the new constants:  
    ***echo "The $worldVar is “, WORLD\_INFO,” miles from the   
    $sunVar.<br>";*** echo "Hello ", $sunVar, "!<br>";  
    ***echo "The $sunVar's core temperature is approximately “,   
    SUN\_INFO, “ degrees Fahrenheit.<br>";*** echo "Hello ", $moonVar, "!<br>";  
    ***echo "The $moonVar is “, MOON\_INFO, “ miles in   
    diameter.</p>";***Run a server test on this.
4. Now let’s format the number output using a ***number\_format()*** function. Convert the output of the constants as follows:  
    ***echo "The $worldVar is ", number\_format(WORLD\_INFO,0), " miles   
    from the $sunVar.<br>";***  
    echo "Hello ", $sunVar, "!<br>";  
    ***echo "The $sunVar's core temperature is approximately ",  
    number\_format(SUN\_INFO,2), " degrees Fahrenheit.<br>";***  
    echo "Hello ", $moonVar, "!<br>";  
    ***echo "The $moonVar is ", number\_format(MOON\_INFO,0), " miles   
    in diameter.</p>";***  
   Run a server test on this.

**Exercise 02\_01\_01 – Step 7**

 

1. Copy the file ***HelloWorld.php*** to a new file called ***Concerts.php***. Update the documentation to indicate the new file name. Update the ***<title>*** content to ***Concerts***. Remove all of the PHP code from the script delimiters.
2. Add an ***<h1>*** and an ***<h2>*** element to the ***<body>***, just above the PHP script delimiters:  
    ***<h1>Central Valley Civic Center</h1>  
    <h2>Summer Concert Season</h2>  
    <?php  
      
    ?>***
3. Add an array constructor to the script code block as follows:  
    ***<?php  
    $concerts = array("Jimmy Buffett", "Chris Isaak", "Bonnie   
    Raitt", "James Taylor", "Alicia Keys");  
    ?>***
4. Append more elements to the array using assignment statements as follows:  
    ***$concerts[] = "Bob Dylan";  
    $concerts[] = "Ryan Cabrera";***Run a server test on this for syntax.
5. Make use of the count() function to use the array length at the end of the code block:  
    ***echo "<p>The following ", count($concerts), " concerts are   
    scheduled:</p><p>";***Run a server test on this.
6. Output the contents of the array with echo statements:  
    ***echo "$concerts[0]<br>";  
    echo "$concerts[1]<br>";  
    echo "$concerts[2]<br>";  
    echo "$concerts[3]<br>";  
    echo "$concerts[4]<br>";  
    echo "$concerts[5]<br>";  
    echo "$concerts[6]</p>";***  
   Run a server test on this.
7. Make use of an informational output function, ***print\_r()***. The output of this function does not have any HTML tags in it. To get decent output, we will insert it in a ***<pre>*** tag to keep its format exactly as received:  
    ***echo "<pre>";  
    print\_r($concerts);  
    echo "</pre>";***  
   Run a server test on this.
8. Let’s use another informational output function, ***var\_dump()***. The output of this function does not have any HTML tags in it. To get decent output, we will insert it in a ***<pre>*** tag to keep its format exactly as received:  
    ***echo "<pre>";  
    var\_dump($concerts);  
    echo "</pre>";***  
   Run a server test on this.

**Exercise 02\_01\_01 – Step 8**

 

1. Copy the file ***HelloWorld.php*** to a new file called ***ArithmeticOperators.php***. Update the documentation to indicate the new file name. Update the ***<title>*** content to ***ArithmeticOperators***. Remove all of the PHP code from the script delimiters.
2. Add the code for addition as follows:  
    ***<?php  
    // Addition  
    $x = 100;  
    $y = 200;  
    $returnValue = $x + $y;  
    echo "<h2>Addition</h2>";  
    echo '<p>$returnValue after addition expression: ', $returnValue, "</p>";  
    ?>***Run a server test on this.
3. Add the code for subtraction at the bottom of the code block as follows:  
    ***// Subtraction  
    $x = 10;  
    $y = 7;  
    $returnValue = $x - $y;  
    echo "<h2>Subtraction</h2>";  
    echo '<p>$returnValue after subtraction expression: ',   
    $returnValue, "</p>";***Run a server test on this.
4. Add the code for division at the bottom of the code block as follows:  
    ***// Division  
    $x = 24;  
    $y = 3;  
    $returnValue = $x / $y;  
    echo "<h2>Division</h2>";  
    echo '<p>$returnValue after division expression: ',   
    $returnValue, "</p>";***  
   Run a server test on this.
5. Add the code for modulus at the bottom of the code block as follows:  
    ***// Division  
    $x = 24;  
    $y = 3;  
    $returnValue = $x / $y;  
    echo "<h2>Division</h2>";  
    echo '<p>$returnValue after division expression: ',   
    $returnValue, "</p>";***  
   Run a server test on this.
6. Now let’s do some code for the ***unary*** operators. Let’s start off with unary increment using ***prefix*** notation. Add the code at the bottom of the code block:  
    ***echo "<h2>Unary Increment: Prefix Notation</h2>";  
    $studentID = 100;  
    $curStudentID = ++$studentID;  
    echo "The first student ID is: ", $curStudentID, "<br>";  
    $curStudentID = ++$studentID;  
    echo "The second student ID is: ", $curStudentID, "<br>";  
    $curStudentID = ++$studentID;  
    echo "The third student ID is: ", $curStudentID, "<br>";***  
   Run a server test on this.
7. Finally let’s do some code for the ***unary*** operators using ***postfix*** notation and see the difference. Add the code at the bottom of the code block:  
    ***echo "<h2>Unary Increment: Postfix Notation</h2>";  
    $studentID = 100;  
    $curStudentID = $studentID++  
    echo "The first student ID is: ", $curStudentID, "<br>";  
    $curStudentID = $studentID++;  
    echo "The second student ID is: ", $curStudentID, "<br>";  
    $curStudentID = $studentID++;  
    echo "The third student ID is: ", $curStudentID, "<br>";***  
   Run a server test on this.

**Exercise 02\_01\_01 – Step 9**

 

1. Copy the file ***HelloWorld.php*** to a new file called ***AssignmentOperators.php***. Update the documentation to indicate the new file name. Update the ***<title>*** content to ***AssignmentOperators***. Remove all of the PHP code from the script delimiters.
2. Add the code for compound addition assignment as follows:  
    <?php  
    ***echo "<p>";  
    // Compound Addition Assignment  
    echo "<h2>Compound Addition Assignment</h2>";  
    $changingVar = 100;  
    $changingVar += 50;  
    echo '$changingVar = 100<br>';  
    echo 'After $changingVar += 50: ', $changingVar, "<br>";*** ?>Run a server test on this.
3. Add the rest of the code for compound assignments at the bottom of the code block as follows:  
    ***// Compound Subtraction Assignment  
    echo "<h2>Compound Subtraction Assignment</h2>";  
    $changingVar -= 30;  
    echo 'After $changingVar -= 30: ', $changingVar, "<br>";  
      
    // Compound Division Assignment  
    echo "<h2>Compound Division Assignment</h2>";  
    $changingVar /= 3;  
    echo 'After $changingVar /= 3;: ', $changingVar, "<br>";  
     
    // Compound Multiplication Assignment  
    echo "<h2>Compound Multiplication Assignment</h2>";  
    $changingVar \*= 8;  
    echo 'After $changingVar \*= 8: ', $changingVar, "<br>";  
     
    // Compound Modulus Assignment  
    echo "<h2>Compound Modulus Assignment</h2>";  
    $changingVar %= 300;  
    echo 'After $changingVar %= 300: ', $changingVar, "<br>";  
     
    // Compound Concatenation Assignment  
    echo "<h2>Compound Modulus Assignment</h2>";  
    $changingVar = "Those who do not remember the past are   
    condemned to repeat it.";  
    $changingVar .= ' - Santayana';  
    echo '$changingVar = Those who do not remember the past   
    are condemned to repeat it.<br>';  
    echo 'After $changingVar .= " - Santayana": ', $changingVar,   
    "</p>";***Run a server test on this.

**Exercise 02\_01\_01 – Step 10**



1. Copy the file ***HelloWorld.php*** to a new file called ***ComparisonExamples.php***. Update the documentation to indicate the new file name. Update the ***<title>*** content to ***ComparisonExamples***. Remove all of the PHP code from the script delimiters.
2. Add the code for a ***relational*** operator into the code block. Notice that we are going to use it in the conditional expression of the ***conditional*** ***operator*** as follows:  
    ***echo "<p>";  
    echo "<h2>Relational Equal Operator</h2>";  
    $value1 = "First text string";  
    $value2 = "Second text string";  
    $returnValue = ($value1 == $value2 ? "true" : "false");  
    echo '$value1: ', $value1, ' $value2: ', $value2, '<br>';  
    echo 'After $value1 == $value2 $returnValue is: ',   
    $returnValue, "<br>";***  
   Run a server test on this.
3. Add the rest code for ***relational*** and ***conditional*** operators into the code block as follows:  
    ***echo "<h2>Relational Equal Operator</h2>";  
    $value1 = 50;  
    $value2 = 75;  
    $returnValue = ($value1 == $value2 ? "true" : "false");  
    echo '$value1: ', $value1, ' $value2: ', $value2, '<br>';  
    echo 'After $value1 == $value2 $returnValue is: ',   
    $returnValue, "<br>";  
     
    echo "<h2>Relational Not Equal Operator</h2>";  
    $returnValue = ($value1 != $value2 ? "true" : "false");  
    echo '$value1: ', $value1, ' $value2: ', $value2, '<br>';  
    echo 'After $value1 != $value2 $returnValue is: ',   
    $returnValue, "<br>";  
     
    echo "<h2>Relational Alternative Not Equal   
    Operator</h2>";  
    $returnValue = ($value1 <> $value2 ? "true" : "false");  
    echo '$value1: ', $value1, ' $value2: ', $value2, '<br>';  
    echo 'After $value1 <> $value2 $returnValue is: ',   
    $returnValue, "<br>";  
     
    echo "<h2>Relational Greater Than Operator</h2>";  
    $returnValue = ($value1 > $value2 ? "true" : "false");  
    echo '$value1: ', $value1, ' $value2: ', $value2, '<br>';  
    echo 'After $value1 > $value2 $returnValue is: ',   
    $returnValue, "<br>";  
     
    echo "<h2>Relational Less Than Operator</h2>";  
    $returnValue = ($value1 < $value2 ? "true" : "false");  
    echo '$value1: ', $value1, ' $value2: ', $value2, '<br>';  
    echo 'After $value1 < $value2 $returnValue is: ',   
    $returnValue, "<br>";  
     
    echo "<h2>Relational Greater Than or Equal   
    Operator</h2>";  
    $returnValue = ($value1 >= $value2 ? "true" : "false");  
    echo '$value1: ', $value1, ' $value2: ', $value2, '<br>';  
    echo 'After $value1 >= $value2 $returnValue is: ',   
    $returnValue, "<br>";  
     
    echo "<h2>Relational Less Than or Equal Operator</h2>";  
    $returnValue = ($value1 <= $value2 ? "true" : "false");  
    echo '$value1: ', $value1, ' $value2: ', $value2, '<br>';  
    echo 'After $value1 <= $value2 $returnValue is: ',   
    $returnValue, "<br>";  
     
    echo "<h2>Relational Equal Values and Same Data Type   
    Operator</h2>";  
    $value1 = 25;  
    $value2 = 25;  
    $returnValue = ($value1 === $value2 ? "true" : "false");  
    echo '$value1: ', $value1, ' $value2: ', $value2, '<br>';  
    echo 'After $value1 === $value2 $returnValue is: ',   
    $returnValue, "<br>";  
     
    echo "<h2>Relational Equal Values and Not Same Data Type   
    Operator</h2>";  
    $returnValue = ($value1 !== $value2 ? "true" : "false");  
    echo '$value1: ', $value1, ' $value2: ', $value2, '<br>';  
    echo 'After $value1 !== $value2 $returnValue is: ',   
    $returnValue, "<br>";  
     
    echo "<h2>The gettype() Function</h2>";  
    echo 'After gettype($returnValue) call: ',   
    gettype($returnValue), "</p>";***Run a server test on this.