Controls and Functions

Functions

- Functions allow you to group programming statements into logical units.
- Before you can use a function in a PHP program, you must first define it. The syntax for defining a function is as follows:

```
<?php
function name_of_function(parameters) {
  statement(s);
}
?>
```

- A formal parameter is a variable that receives a value passed to a function when it is called. For example,
  - `function calculateSalesTotal($Subtotal) {]`
  - `function calculateSalesTotal($Subtotal, $SalesTax, $Shipping) {}`
- You can also assign default values to a parameter as follows:

```
function sampleFunction($Num1="100", $Num2="200", $Num3="300") {
  echo ("<p>$Num1</p>");
  echo ("<p>$Num2</p>");
  echo ("<p>$Num3</p>");
}
```

- Unlike variables, function names are case insensitive, which means you can call the displayCompanyName() function with any of the following statements:
  - `displayCompanyName("Course Technology");`
  - `DisplayCompanyName("Course Technology");`
  - `DISPLAYCOMPANYNAME("Course Technology");`
- To actually return a value, the function must include a return statement. Example,

```
function averageNumbers($a, $b, $c) {
  $SumOfNumbers = $a + $b + $c;
  $Result = $SumOfNumbers / 3;
  return $Result;
}
```

Example of Simple Functions

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        function displayMessage($FirstMessage) {
            echo "<p>$FirstMessage</p>";
        }

        function returnMessage() {
            return "<p>This message was returned from a function.</p>";
        }

        displayMessage("This message was displayed from a function.");
        $ReturnValue = returnMessage();
        echo $ReturnValue;
        ?>
    </body>
</html>
```

This message was displayed from a function.

This message was returned from a function.

Passing Parameters by Reference

- “Passed-by-value ” - the value of a variable is passed as the parameter of a function. Any changes made to the parameter’s value within the function are lost when control is passed from the function back to the program.
- “Passed-by-reference” - the actual variable is used within the function. Any changes to that variable made by the function will remain after the function completes.

Example of Passed-by-reference

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Demo of Passed-by-reference</title>
    </head>
    <body>
        <?php

        function IncrementByValue($CountByValue) {
            ++$CountByValue;
            echo "<p>IncrementByValue() value is $CountByValue.</p>";
        }

        function IncrementByReference(&$CountByReference) {
            ++$CountByReference;
            echo "<p>IncrementByReference() value is $CountByReference.</p>";
        }

        $Count = 1;
        echo "<p>Main program starting value is $Count.</p>";
        IncrementByValue($Count);
        echo "<p>Main program between value is $Count.</p>";
        IncrementByReference($Count);
        echo "<p>Main program ending value is $Count.</p>";
        ?>
    </body>
</html>
```

Main program starting value is 1.

IncrementByValue() value is 2.

Main program between value is 1.

IncrementByReference() value is 2.

Main program ending value is 2.

Variable Scope

- The variable' scope decides where in your program a declared variable can be used.
- A variable's scope can be either global or local.
  - A **global variable** is declared outside a function and is available to all parts of your program.
  - A **local variable** is declared inside a function and is only available within that function. Local variables cease to exist when the function ends.
- In PHP, global variables are NOT automatically available to all parts of your program, including functions.
  - you must declare a global variable with the global keyword inside a function definition to make the variable available within the scope of that function.

```
<?php
$GlobalVariable = "Global variable";
function scopeExample() {
   global $GlobalVariable;
   echo "<p>$GlobalVariable</p>";
}
scopeExample();
?>
```

Example of Variable Scope

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Demo of Passed-by-reference</title>
    </head>
    <body>
        <?php
        $GlobalVariable = "Global variable";
        function scopeExample() {
            $LocalVariable = "<p>Local variable</p>";
            echo "<p>$LocalVariable</p>"; // displays successfully
            echo "<p>$GlobalVariable</p>"; // error message
        }
        scopeExample();
        echo "<p>$GlobalVariable</p>";
        echo "<p>$LocalVariable</p>"; // error message
        ?>
    </body>
</html>
```

Decision Making

- Decision making or Flow control is the process of determining the order in which statements execute in a program.
- If statements

```
if (conditional expression)
   statement;
```

- Example 1

```
$ExampleVar = 5;
if ($ExampleVar == 5) // Condition evaluates to 'TRUE'
    echo "<p>The variable is equal to $ExampleVar.</p>";
echo "<p>This text is generated after the 'if' statement.</p>";
```

- Example 2

```
ExampleVar = 5;
if ($ExampleVar == 5) { // Condition evaluates to 'TRUE'
    echo "<p>The condition evaluates to true.</p>";
    echo '<p>$ExampleVar is equal to ', "$ExampleVar.</p>";
    echo "<p>Each of these lines will be displayed.</p>";
}
echo "<p>This statement always executes after the 'if' statement.</
p>";
```

Example of Dice Rolls

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Roll Dice</title>
    </head>
    <body>
        <?php
        $DiceNamesSingle = array("one", "two", "three", "four", "five", "six");
        $DiceNamesDouble = array("ones", "twos", "threes", "fours", "fives", "sixes");
        function CheckForDoubles($Die1, $Die2) {
            global $DiceNamesSingle;
            global $DiceNamesDouble;
            if ($Die1 == $Die2) // Doubles
                echo "The roll was double ", $DiceNamesDouble[$Die1-1], ".<br />";
            if ($Die1 != $Die2) // Not Doubles
                echo "The roll was a ", $DiceNamesSingle[$Die1-1], " and a ",
                    $DiceNamesSingle[$Die2-1], ".<br />";
        }
        function DisplayScoreText($Score) {
            if ($Score == 2)
                echo "You rolled TWO!<br />";
            if ($Score == 3)
                echo "You rolled THREE!<br />";
            if ($Score == 5)
                echo "You rolled FIVE!<br />";
            if ($Score == 7)
                echo "You rolled SEVEN!<br />";
            if ($Score == 9)
                echo "You rolled NINE!<br />";
            if ($Score == 11)
                echo "You rolled ELEVENTH!<br />";
            if ($Score == 12)
                echo "You rolled TWELVE!<br />";
        }
        $Dice = array();
        $Dice[0] = rand(1,6);
        $Dice[1] = rand(1,6);
        $Score = $Dice[0] + $Dice[1];
        echo "<p>";
        echo "The total score for the roll was $Score.<br />";

        CheckForDoubles($Dice[0],$Dice[1]);
        DisplayScoreText($Score);
        echo "</p>";
        ?>
    </body>
```

The total score for the roll was 9.
The roll was a six and a three.
You rolled NINE!

If-else and if-else-if statements

- if else statements

```
if (conditional expression)
  statement;
else
  statement;
```

Example,

```
$name = "Peter";
if ($name == "Peter")
      echo "<p>name is Peter</p>";
else
      echo "<p>name is not Peter</p>";
```

- if else if statements

```
if (conditional expression)
  statement;
else if (conditional expression)
  statement;
```

Example,

```
$name = "Peter";
if ($name == "Peter")
      echo "<p>name is Peter</p>";
else if ($name == "Lily")
      echo "<p>name is not Lily</p>";
```

Example of Enhanced Dice Rolls

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Roll Dice</title>
    </head>
    <body>
        <?php
        $DiceNamesSingle = array("one", "two", "three", "four", "five", "six");
        $DiceNamesDouble = array("ones", "twos", "threes", "fours", "fives", "sixes");
        function CheckForDoubles($Die1, $Die2) {
            global $DiceNamesSingle;
            global $DiceNamesDouble;
            $ReturnValue = false;
            if ($Die1 == $Die2) {// Doubles
                echo "The roll was double ", $DiceNamesDouble[$Die1-1], ".<br />";
                $ReturnValue = true;
            }
            else { // Not Doubles
                echo "The roll was a ", $DiceNamesSingle[$Die1-1], " and a ",
                    $DiceNamesSingle[$Die2-1], ".<br />";
                $ReturnValue = false;
            }
            return $ReturnValue;
        }
        function DisplayScoreText($Score, $Double) {
            if ($Double) { // Doubles were rolled
                if ($Score == 2)
                    echo "You rolled TWO!<br />";
                if ($Score == 12)
                    echo "You rolled TWELVE!<br />";
            }
            else {
                if ($Score == 3)
                    echo "You rolled THREE!<br />";
                if ($Score == 5)
                    echo "You rolled FIVE!<br />";
                if ($Score == 7)
                    echo "You rolled SEVEN!<br />";
                if ($Score == 9)
                    echo "You rolled NINE!<br />";
                if ($Score == 11)
                    echo "You rolled ELEVENTH!<br />";
            }
        }
        $Dice = array();
        $Dice[0] = rand(1,6);
        $Dice[1] = rand(1,6);
        $Score = $Dice[0] + $Dice[1];
        echo "<p>";
        echo "The total score for the roll was $Score.<br />";
        $Double = CheckForDoubles($Dice[0],$Dice[1]);
        DisplayScoreText($Score, $Double);
        echo "</p>";
        ?>
    </body>
</html>
```

switch statements

- The switch statement compares the value of an expres- sion to a value contained within a special statement called a case label.
- A case label represents a specific value and contains one or more statements that execute if the value of the case label matches the value of the switch statement's expression.

```
switch (expression) {
     case label:
          statement(s);
          break;
     case label:
          statement(s);
          break; ...
     default:
         statement(s);
         break;
}
```

- You can use a variety of data types as case labels within the same switch statement. The following code shows examples of four case labels:

```
case $ExampleVar:  // variable name
     statement(s);
    break;
case "text string": // string literal
     statement(s);
     break;
case 75: // integer literal
     statement(s);
     break;
case -273.4: // floating-point literal
     statement(s);
     break;
```

Loop Statements

- while statements

```
while (conditional expression) {
        statement(s);
}
```

- Example

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        $Count = 1;
        while ($Count <= 5) {
            echo "$Count<br />";
            ++$Count;
        }
        echo "<p>You have displayed 5 numbers.</p>";
        ?>
    </body>
</html>
```

```
1
2
3
4
5
You have displayed 5 numbers.
```

Example of using while loop in Dice Roll

```
$DoubleCount = 0;
$RollNumber = 1;
$Dice = array();
while ($RollNumber <= 5) {
    $Dice[0] = rand(1,6);
    $Dice[1] = rand(1,6);
    $Score = $Dice[0] + $Dice[1];
    echo "<p>";
    echo "The total score for roll $RollNumber was $Score.<br />";
    $Double = CheckForDoubles($Dice[0],$Dice[1]);
    DisplayScoreText($Score, $Double);
    echo "</p>";
    if ($Double)
       ++$DoubleCount;
    ++$RollNumber;
}
```

```
The total score for roll 1 was 8.
The roll was a five and a three.
You rolled an easy 8!

The total score for roll 2 was 5.
The roll was a two and a three.
You rolled FIVE!

The total score for roll 3 was 9.
The roll was a three and a six.
You rolled NINE!

The total score for roll 4 was 7.
The roll was a five and a two.
You rolled SEVEN!

The total score for roll 5 was 7.
The roll was a five and a two.
You rolled SEVEN!
```

Other Loop Statements

- do while statements

```
do {
        statement(s);
} while (conditional expression);
```

- Example

```
$DaysOfWeek = array("Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday", "Sunday");
$Count = 0;
do {
      echo $DaysOfWeek[$Count], "<br />";
      $Count++;
} while ($Count < 7);
```

- for loop statements

```
for (counter declaration and initialization; condition;
post statement) {
       statement(s); }
```

- Example

```
$FastFoods = array("pizza", "burgers", "french fries",
"tacos", "fried chicken");
for ($Count = 0; $Count < 5; $Count++) {
      echo $FastFoods[$Count], "<br />";
}
```

foreach Statements

- The foreach statement is used to iterate or loop through the elements in an array. And you do not need to include any counters within a foreach statement. Instead, you specify an array expression.

```
foreach ($array_name as $variable_name) {
    statement(s);
}
```

- Example

```
$DaysOfWeek = array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday");
foreach ($DaysOfWeek as $Day) {
    echo "<p>$Day</p>";
}
```

- The more advanced form of the foreach statement allows you to retrieve both the index (or key) and the value of each array element.

```
foreach ($array_name as $index_name => $variable_name {
    statement(s);
}
```

- Example

```
$DaysOfWeek = array("Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday");
foreach ($DaysOfWeek as $DayNumber => $Day) {
    echo "<p>Day $DayNumber is $Day</p>";
}
```

Day 0 is Monday

Day 1 is Tuesday

Day 2 is Wednesday

Day 3 is Thursday

Day 4 is Friday

Day 5 is Saturday

Day 6 is Sunday

Example of switch statements

```
function DisplayScoreText($Score, $Double) {
            switch ($Score) {
                case 2:
                    echo "You rolled TWO!<br />";
                    break;
                case 3:
                    echo "You rolled THREE!<br />";
                    break;
                case 5:
                    echo "You rolled FIVE!<br />";
                    break;
                case 7:
                    echo "You rolled SEVEN!<br />";
                    break;
                case 9:
                    echo "You rolled NINE!<br />";
                    break;
                case 11:
                    echo "You rolled ELEVENTH!<br />";
                    break;
                case 12:
                    echo "You rolled TWELVE!<br />";
                    break;
                default:
                    if ($Score % 2 == 0) { /* An even number */
                      if ($Doubles) {
                           echo "You rolled a hard $Score!<br />";
                      }
                      else { /* Not doubles */
                            echo "You rolled an easy $Score!<br />";
                      }
                    }
        }
}
```

The total score for the roll was 8.
The roll was a six and a two.
You rolled an easy 8!