Session 5

Conditional and Flow Control Statements and Arrays

Session Overview

In this session, you will learn to:

- List and explain different types of conditional statements and their usage
- Distinguish between `if`, `if…else`, and `if…elseif…else` statements
- Define and use switch statement
- List and explain different types of loop statements and their usage
- Distinguish between `while`, `do…while` and `for` loops
- Identify use of `foreach` loop statement
- Identify use of break and continue statement
- Outline different types of Arrays
- List the uses of different types of Sorting functions

PHP Conditional Statements

Conditional statements in PHP help the user to arrive at a decision based on certain conditions. When a user writes code in PHP, there will be scenarios where different actions must be performed for different conditions.

Conditional statements are used in such scenarios. These conditions are defined by a set of conditional statements which contain expressions that are evaluated to a Boolean value of true or false.

PHP language supports the following conditional statements:

- `if`
- `if...else`
- `If...elseif...else`
- `switch`

`if` Statement [1-2]

An `if` statement allows you to execute one or more statements after evaluating a specified logical condition.

It starts with the `if` keyword and is followed by the condition.

If the condition evaluates to *true*, the block of statements following the `if` statement is executed.

If the condition evaluates to *false*, the block of statements following the `if` statement is ignored and the statement after the block is executed.

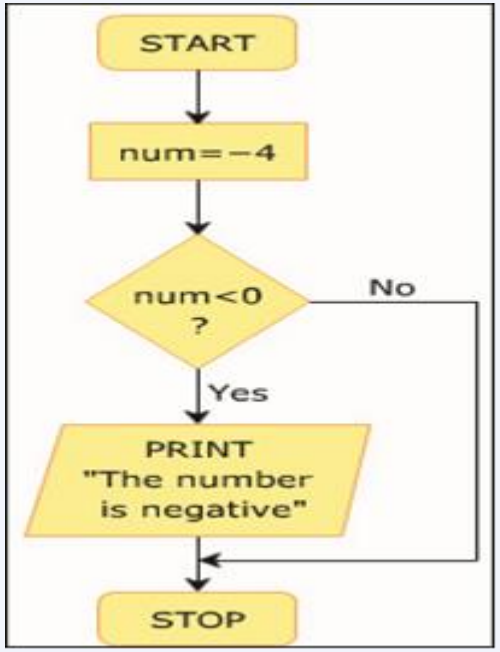

Figure 1:
Flowchart Representing if Statement

`if` Statement [2-2]

Code Snippet:

```
<?php
$timeofday = date("H");
if ($timeofday < "12") {
  echo "Happy Morning!";
}
?>
```

Output

```
Happy Morning!
```

Code Snippet shows a sample program that uses `if` statement to display **'Happy Morning'** if the current time is less than 12 o'clock.

In Code Snippet, the program is executed when current time is less than 12 o'clock. Hence, the output is 'Happy Morning!'

`if...else` Statement [1-2]

The `if` statement executes a block of statements only if the specified condition is true.

However, in some situations, it is required to define an action for a false condition. This is done using an `if...else` construct.

The `if...else` construct starts with `if` block followed by an `else` block.

The `else` block starts with `else` keyword followed by a block of statements.

If the condition specified in the `if` statement evaluates to false, the statements in the `else` block are executed.

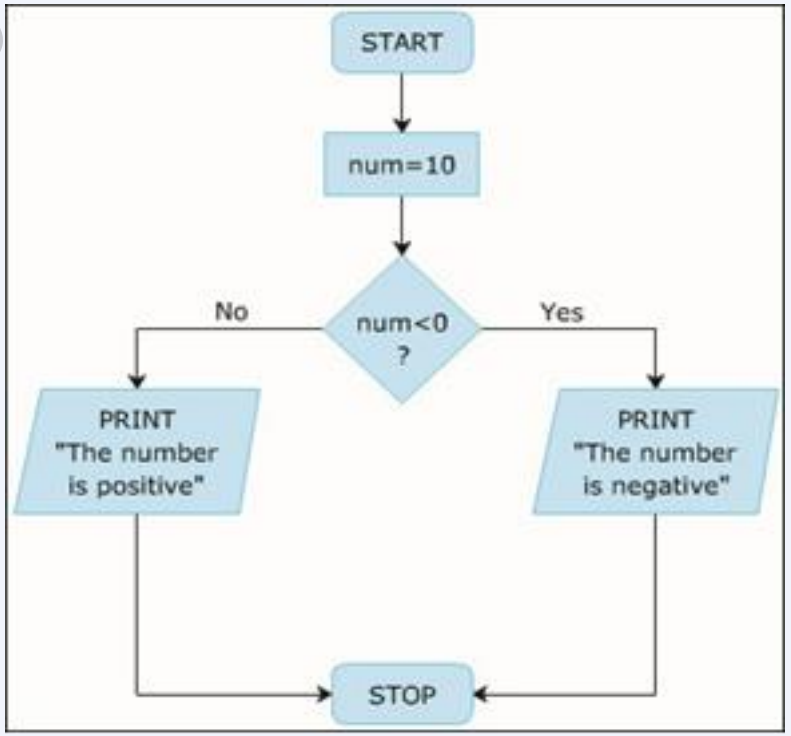

Figure 2: Flowchart for if...else Statement

`if...else` Statement [2-2]

Code Snippet:

```
<?php
$hourOfDay = date("H");
if ($hourOfDay< "18") {
  echo "Have a Nice Day
ahead!";
}
else {
  echo "Good Night!";
}
?>
```

Output

```
Good Night!
```

In Code Snippet, the `else` statement will be executed and the output displayed is **Good Night!** as the current time hour is greater than 18.

However, if the time hour is less than 18, then `if` statement executes to display **Have a Nice Day ahead!**

`if...elseif...else` Statement [1-2]

Statement `if...elseif...else` is used when the user wants to handle multiple conditions.

In other words, `if...elseif...else` statement executes different codes when there are more than two conditions.

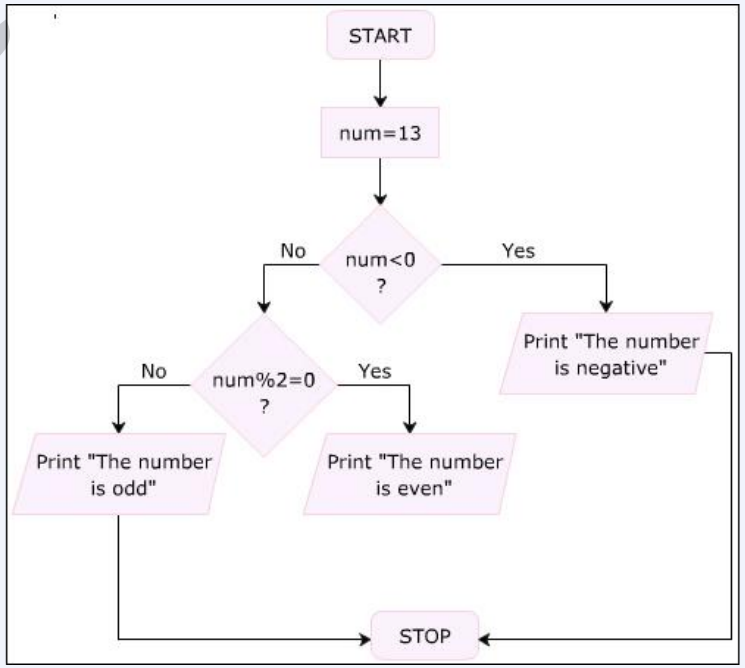

Figure: Flowchart for if…elseif…else Statement

if...elseif...else Statement [2-2]

Code Snippet:

```
<?php
$t = date("H");
echo "<p>The hour (of the
server) is ". $t;
echo " and message is:</p>";
if ($t < "12") {
  echo "Happy morning!";
} elseif ($t < "20") {
  echo " Nice day!";
} else {
  echo " Good night!";
}
?>
```

In the code, `if…elseif…else` statement is used to display '**Happy morning**' if current hour is less than 12 and '**Nice day'** if 20 is less than the current time. Else, output is '**Good night**'. Since the hour of the server is 15, output displayed is Good Day!

Output

The hour (of the server) is 15 and message is:

Happy Morning!

`switch` Statement

Code Snippet:

```
<?php
$language = "PHP 8";
switch ($language) {
  case "C":
    echo "Your favorite language is C ";
    break;
  case "PHP 8":
    echo "Your favorite language is PHP 8";
    break;
  case "C++":
    echo "Your favorite language is C++";
    break;
  default:
    echo "Your favorite language is neither C, PHP 8, nor C++";
}
?>
```

Code Snippet shows how to use the switch statement.

Output

```
Your favorite language is PHP 8
```

PHP **while** Loop

Code Snippet:

```
<?php
$n = 1;
 while($n <= 10) {
  echo "The number is: $n
<br>";
  $n++;
}
?>
```

Code Snippet shows a sample program for displaying numbers from 1 to 10 using a while loop.

Output

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
The number is: 10
```

PHP do...while Loop

Code Snippet:

```
<?php
$n = 15;
do {
  echo "The number is:
$n <br>";
  $n++;
} while ($n <= 20);
?>
```

Output

```
The number is: 15
The number is: 16
The number is: 17
The number is: 18
The number is: 19
The number is: 20
```

Code Snippet shows a sample program to display numbers from 1 to 5 using the `do...while` loop.

PHP `for` Loop

Code Snippet:

```
<?php
for ($n = 11; $n <= 20;
$n++) {
  echo "The number is:
$n <br>";
}
?>
```

Output

```
The number is: 11
The number is: 12
The number is: 13
The number is: 14
The number is: 15
The number is: 16
The number is: 17
The number is: 18
The number is: 19
The number is: 20
```

Code Snippet shows a sample program to display numbers from 11 to 20 using the `for` loop.

Following parameters are used in for loop:

- `init counter` is used to initialize the counter value of loop.
- `test counter` is used to evaluate iteration for each loop. The loop continues if it evaluates to TRUE. The loop ends if it evaluates to FALSE.
- `increment counter` is used to increase the counter value of loop.

foreach Loop Statement

Code Snippet:

```
<?php
$flower = array("Rose"=>"10",
"Lily"=>"20", "Lotus"=>"30");
foreach($flower as $f => $value) {
  echo "$f= $value<br>";
}?>
```

Output

```
Rose=10
Lily=20
Lotus=30
```

Code Snippet shows a sample program to display both keys and values of the array using `foreach` loop.

`break` Statement

Code Snippet:

```
<?php
for ($n = 5; $n < 15; $n++) {
  if ($n == 9) {
    break;
  }
  echo "Number is: $n <br>";
}
?>
```

Output

```
Number is: 5
Number is: 6
Number is: 7
Number is: 8
```

Code Snippet shows a sample program to display the use of `break` statement and how to jump out of a loop.

`continue` Statement

Code Snippet:

```
<?php
for ($n = 0; $n < 10; $n++) {
  if ($n == 4) {
    continue;
  }
  echo "The number is: $n
<br>";
}
?>
```

Output

```
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 5
The number is: 6
The number is: 7
The number is: 8
The number is: 9
```

Code Snippet shows a sample program to display the use of `continue` statement and display values from 0 to 9 and skip number 4.

Concepts of Arrays in PHP

array() function is used to create an array in PHP.

An array can store multiple values with the same name and the values can be accessed by referencing an index number.

For example, if you want to hold 300 numbers, then instead of defining 300 variables, it is easy to define an array of length 300.

Arrays are classified into three types:

- Numerical array
- Associative array
- Multi-dimensional array

Numerical Array

Code Snippet:

```
<?php
/*  Method for creating an array. */
$num = array(21, 22, 23, 24, 25, 26);
foreach( $num as $value )
{
echo "Value is $value <br />";
}
$num[0]="one";
$num[1]="two";
$num[2]=23;
foreach( $num as $value )
{
echo "Value is $value <br />";
}
?>
```

Output

```
Value is 21
Value is 22
Value is 23
Value is 24
Value is 25
Value is 26
Value is one
Value is two
Value is 23
```

Code Snippet shows a sample program that shows how to create and access numeric arrays.

Associative Array

Code Snippet:

```
<?php
/* One approach to create an associative array. */
$Age = array(
"John" => 20,
"Roger" => 10,
"Susan" => 60
);
echo "Age of John is ". $Age['John'] . "<br />";
echo "Age of Roger is ". $Age['Roger']. "<br />";
echo "Age of Susan is ". $Age['Susan']. "<br />";
/* Another approach to create an associative array. */
$Age['John'] = "Adult";
$Age['Roger'] = "Child";
$Age['Susan'] = "Senior Citizen";
echo " John is ". $Age['John'] . "<br />";
echo " Roger is ". $Age['Roger']. "<br />";
echo " Susan is ". $Age['Susan']. "<br />";
?>
```

Output

```
Age of John is 20
Age of Roger is 10
Age of Susan is 60
John is Adult
Roger is Child
Susan is Senior Citizen
```

Code Snippet shows a sample program to create an associative array.

Multi-Dimensional Arrays

Code Snippet:

```
<?php
$contacts = array(
 array
(
"Name" => "David",
"Email" => "David12@gmail.com",
"Number" => 39365421
),
array
(
"Name" => "Peter",
"Email" => "Peter@gmail.com",
"Number" => 299853412
),
=> array
(
"Name" => "John",
"Email" => "John13@yahoo.com",
"Number" => 39982145
)
);
/* Accessing multi-dimensional array values
*/
echo "Email ID of Peter : " ;
echo $contacts[1]['Email'] . "<br />";
echo "Contact number of David  : ";
echo $contacts[0][‘Number’] . "<br />";
echo "Contact number of John : ";
echo $contacts[2][‘Number’]. "<br/>";
?>
```

Output

```
Email ID of Peter: Peter@gmail.com
The contact number of David: 39365421
The contact number of John: 39982145
```

Code Snippet shows a sample program to create a two-dimensional array to store contact information of three people that is name, email id, and number.

Indexed Arrays

Code Snippet:

```
<?php
$student=array("Peter", "John",
"David", "Sean");
echo "Names of the students
are: ".
$Student[0]. ",".$Student[1]. "
,".$Student[2].
", and ".$Student[3]. ".";
?>
```

Output

```
Names of the students
are: Peter, John, David, and
Sean.
```

Code Snippet shows a sample program for creating an indexed array named `$Student`, assigning it with four elements. The output is a text that contains the array values.

Sorting Arrays

The most popular functions for sorting arrays are:

- `sort()` and `rsort()`: Indexed arrays are sorted using this array.
- `ksort()` and `krsort()`: Associative arrays are sorted by key using this array.
- `asort()` and `arsort()`: These are used to sort associative arrays by value.

Sort Functions	Description
sort()	Sort arrays in ascending order
rsort()	Sort arrays in descending order
asort()	Sort associative arrays in ascending order, according to the value
ksort()	Sort associative arrays in ascending order, according to the key
arsort()	Sort associative arrays in descending order, according to the value
krsort()	Sort associative arrays in descending order, according to the key

Table: Array Sort Functions

Sort Array in Ascending Order - `sort()`

Code Snippet:

```
<?php
$Student = array("Peter", "John", "David");
sort($Student);
$clength = count($Student);
for($x = 0; $x < $clength; $x++) {
  echo $Student[$x];
  echo "<br>";
}
?>
```

Output

```
David
John
Peter
```

In Code Snippet, the elements of the `$Student` array are sorted in ascending alphabetical order.

Sort Array (Ascending Order), According to Value - `asort()`

Code Snippet:

```
<?php
$age = array("Peter"=>"35", "John"=>"37", "David"=>"43");
asort($age);
foreach($age as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}
?>
```

Output

```
Key=Peter, Value=35
Key=John, Value=37
Key=David, Value=43
```

Using `asort()`, Code Snippet demonstrates an example for sorting an associative array in ascending order according to the value.

Sort Array (Ascending Order), According to Key - `ksort()`

Code Snippet:

```
<?php
$age = array("Peter"=>"35", "John"=>"37", "David"=>"43");
ksort($age);
foreach($age as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}
?>
```

Output

```
Key=David, Value=43
Key=John, Value=37
Key=Peter, Value=35
```

Code Snippet shows an example that sorts an associative array by key in ascending order.

Sort Array (Descending Order), According to Value - `arsort()`

Code Snippet:

```
<?php
$age = array("Peter"=>"35", "John"=>"37", "David"=>"43");
arsort($age);

foreach($age as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}
?>
```

Output

```
Key=David, Value=43
Key=John, Value=37
Key=Peter, Value=35
```

In Code Snippet, `arsort()` array is used to sort the associative array in descending order, according to the value.

Sort Array (Descending Order), According to Key - krsort()

Code Snippet:

```
<?php
$age = array("Peter"=>"35", "John"=>"52", "David"=>"43");
krsort($age);
foreach($age as $x => $x_value) {
  echo "Key=" . $x . ", Value=" . $x_value;
  echo "<br>";
}
?>
```

Output

```
Key=Peter, Value=35
Key=John, Value=52
Key=David, Value=43
```

Code Snippet shows an example for sorting an associative array in descending order, according to the key.

Sort Array in Descending Order - `rsort()`

Code Snippet:

```
<?php
$Student = array("Peter", "John", "David");
rsort($Student);
$clength = count($Student);
for($x = 0; $x < $clength; $x++) {
  echo $Student[$x];
  echo "<br>";
}
?>
```

Output

```
Peter
John
David
```

Code Snippet shows an example for sorting the elements of the `$Student` array in descending alphabetical order.

Summary

- There are three types of conditional statements in PHP, namely `if`, `if…else`, and `if…elseif…else` statements.
- If a test condition can give multiple values, then the `switch` statement is used for testing multiple conditions and explaining different types of conditional statements and using them.
- Loops are used for executing the same block of code multiple times till a condition is satisfied.
- There are three types of loop statements, namely `while`, `do…while`, and `for` loops.
- `foreach` loop statement loops through a block of code for each element in an array.
- `break` statement is used to jump out of a particular code block.
- The `continue` statement skips the iteration in the loop and proceeds to the next iteration when a specified condition is satisfied.
- There are three different types of arrays, namely numeric, associative, and multi-dimensional array.
- The elements in an array can be sorted in alphabetical or numerical order and in descending or ascending manner.
- There are six functions for sorting in PHP namely, `sort()`, `rsort()`, `asort()`, `ksort()`, `arsort()`, and `krsort()`.