HiLCoE School of Computer Science & Technology

Chapter One: Revision on PHP

Course Title : Web Technologies II

Instructor name: Yitayew Solomon
E-mail address: yitayewsolomon3@gmail.com

Iterative Statements in PHP

PHP Loops

Loops in PHP allow you to execute a block of code multiple times, either a specified number of times or while a certain condition is true. PHP supports the following types of loops:

1. while loop
2. do...while loop
3. for loop
4. foreach loop

Cont. ...

1. while Loop

The `while` loop will continue executing the block of code as long as the specified condition is true.

Syntax:

php Copy code

```php
while (condition) {
    // code to be executed
}
```

Cont. ...

Copy code

```php
<?php
$i = 1;

while ($i <= 5) {
    echo "The number is: $i\n";  // Outputs 1 to 5
    $i++;  // Increment $i by 1
}
?>
```

Explanation:

- The loop starts by checking the condition (`$i <= 5`).
- As long as `$i` is less than or equal to 5, the code inside the loop executes.
- After each iteration, `$i` is incremented by 1.
- The loop terminates when `$i` becomes 6 (and the condition is false).

Cont. ...

2. do...while Loop

The `do...while` loop is similar to the `while` loop, but the block of code is guaranteed to execute at least once, even if the condition is false, because the condition is checked after the code is executed.

Syntax:

php Copy code

```php
do {
    // code to be executed
} while (condition);
```

Cont. ...

php

Copy code

```php
<?php
$i = 6;

do {
    echo "The number is: $i\n";  // Outputs 6
    $i++;
} while ($i <= 5);
?>
```

Explanation:

- In this example, the `do...while` loop runs at least once, even though the condition (`$i <= 5`) is false at the start.
- The output will be "The number is: 6" because the code executes once before the condition is checked.

Cont. ...

3. for Loop

The `for` loop is used when you know in advance how many times you want to execute a block of code. It has three expressions: initialization, condition, and increment/decrement.

Syntax:

php Copy code

```php
for (initialization; condition; increment) {
    // code to be executed
}
```

Cont. ...

Example:

php Copy code

```php
<?php
for ($i = 1; $i <= 5; $i++) {
    echo "The number is: $i\n";  // Outputs 1 to 5
}
?>
```

Explanation:

- The loop initializes `$i` to 1.
- It checks the condition `$i <= 5` before each iteration. If true, it runs the code inside the loop.
- After each iteration, `$i++` increments `$i` by 1.
- The loop stops when `$i` becomes greater than 5.

Cont. ...

4. foreach Loop

The `foreach` loop is specifically designed to loop through arrays. It iterates over each value in an array and assigns it to a variable.

Syntax:

php Copy code

```php
foreach ($array as $value) {
    // code to be executed
}
```

Cont. ...

Example:

php Copy code

```php
<?php
$colors = array("Red", "Green", "Blue");

foreach ($colors as $color) {
    echo "The color is: $color\n";  // Outputs Red, Green, and Blue
}
?>
```

Explanation:

- The `foreach` loop iterates over each element in the `$colors` array.
- In each iteration, the value of the current element is assigned to the `$color` variable, and the code inside the loop executes.

Loop Control Statements

Break and Continue Statements

- `break`: Terminates the loop entirely.
- `continue`: Skips the current iteration and moves to the next one.

Example Using `break`:

php Copy code

```php
<?php
for ($i = 1; $i <= 10; $i++) {
    if ($i == 5) {
        break;  // Stop the loop when $i is 5
    }
    echo "The number is: $i\n";  // Outputs 1 to 4
}
?>
```

Cont. ...

Example Using `continue`:

php

Copy code

```php
<?php
for ($i = 1; $i <= 5; $i++) {
    if ($i == 3) {
        continue;  // Skip the iteration when $i is 3
    }
    echo "The number is: $i\n";  // Outputs 1, 2, 4, and 5
}
?>
```

Exercise

Exercise

1. Write a PHP script that:
   - Uses a `while` loop to display numbers from 1 to 10.
   - Uses a `do-while` loop to display even numbers from 2 to 10.
   - Uses a `for` loop to display numbers from 10 down to 1.
   - Uses a `foreach` loop to display all elements of an array of colors (e.g., "Red", "Green", "Blue", "Yellow").

Solution

Solution

php

Copy code

```php
<?php

// 1. While loop to display numbers from 1 to 10
echo "While Loop (1 to 10):\n";
$i = 1;
while ($i <= 10) {
    echo "$i ";
    $i++;
}
echo "\n\n";

// 2. Do-While loop to display even numbers from 2 to 10
echo "Do-While Loop (Even numbers 2 to 10):\n";
$j = 2;
do {
    echo "$j ";
    $j += 2;
} while ($j <= 10);
```

Cont. ...

Copy code

```php
echo "\n\n";

// 3. For loop to display numbers from 10 down to 1
echo "For Loop (10 to 1):\n";
for ($k = 10; $k >= 1; $k--) {
    echo "$k ";
}
echo "\n\n";

// 4. Foreach loop to display all elements in an array of colors
echo "Foreach Loop (Colors):\n";
$colors = ["Red", "Green", "Blue", "Yellow"];
foreach ($colors as $color) {
    echo "$color ";
}
echo "\n";

?>
```

PHP Function

PHP Functions

A **function** in PHP is a block of code that can be repeatedly called from various parts of a program. Functions are used to organize code into logical chunks, increase code reusability, and make programs easier to maintain. PHP has both **built-in** functions and allows you to create **user-defined** functions.

Types of Functions in PHP

1. **Built-in Functions**: Predefined functions provided by PHP, such as `strlen()`, `echo()`, etc.
2. **User-Defined Functions**: Functions you create in PHP to perform specific tasks.

Declaring Function

Declaring and Calling Functions

A user-defined function is declared using the `function` keyword, followed by a function name, parentheses (which can optionally contain parameters), and a code block that defines the functionality of the function.

Syntax:

php Copy code

```php
function functionName() {
    // code to be executed
}
```

Cont. ...

Example:

php Copy code

```php
<?php
function greet() {
    echo "Hello, welcome to PHP functions!\n";
}

greet();  // Calling the function
?>
```

Explanation:

- The function `greet()` is defined with no parameters.
- When called, it executes the `echo` statement and prints the message `"Hello, welcome to PHP functions!"`.

Function Parameter

PHP Function with Parameters

You can pass information to a function using parameters. Parameters are specified in the parentheses after the function name.

Syntax:

php Copy code

```php
function functionName($param1, $param2) {
    // code to be executed
}
```

Cont. ...

Example:

php Copy code

```php
<?php
function greet($name) {
    echo "Hello, $name!\n";
}

greet("John");  // Outputs: Hello, John!
greet("Alice");  // Outputs: Hello, Alice!
?>
```

Explanation:

- The `greet($name)` function accepts a single parameter `$name`.
- The function prints a personalized greeting based on the value of `$name` passed during the function call.

Cont. ...

PHP Function with Return Value

A function can return a value to the calling code using the `return` statement.

Syntax:

php Copy code

```php
function functionName() {
    return value;
}
```

Cont. ...

Example:

php Copy code

```php
<?php
function add($a, $b) {
    return $a + $b;  // Returns the sum of $a and $b
}

$result = add(5, 10);
echo "The sum is: $result\n";  // Outputs: The sum is: 15
?>
```

Explanation:

- The function `add($a, $b)` takes two parameters and returns their sum.
- The returned value is stored in the variable `$result` and then printed.

Cont. ...

Default Parameters in Functions

You can set default values for parameters in case they are not provided during the function call.

Syntax:

php Copy code

```php
function functionName($param = default_value) {
    // code to be executed
}
```

Cont. ...

Example:

php Copy code

```php
<?php
function greet($name = "Guest") {
    echo "Hello, $name!\n";
}

greet();  // Outputs: Hello, Guest!
greet("Sam");  // Outputs: Hello, Sam!
?>
```

Explanation:

- The function `greet($name = "Guest")` has a default value of `"Guest"` for the parameter `$name`.
- If no argument is passed, `"Guest"` is used as the default name.

Cont. ...

PHP Functions with Multiple Parameters

A function can accept multiple parameters, separated by commas.

Example:

php Copy code

```php
<?php
function multiply($a, $b, $c) {
    return $a * $b * $c;
}

$result = multiply(2, 3, 4);
echo "The product is: $result\n";  // Outputs: The product is: 24
?>
```

Explanation:

- The `multiply($a, $b, $c)` function tak ↓ ıree arguments and returns the product of those three values.

Cont. ...

Exercise

Create a PHP function named `calculateCircleArea` that:

- Accepts one parameter: `$radius`, the radius of a circle.
- Calculates and returns the area of the circle using the formula:

$$\text{Area} = \pi \times (\text{radius})^2$$

- Use PHP's built-in constant `M_PI` for the value of π (pi).
- If the radius is negative or zero, the function should return `"Invalid radius"`.

Finally, test this function by calling it with different radius values (e.g., 5, -2, 0).

Cont. ...

function.php

function.php > ...

```php
<?php
// Function to calculate the area of a circle
function calculateCircleArea($radius) {
    if ($radius <= 0) {
        return "Invalid radius";
    }
    return M_PI * pow($radius, 2);
}
// Testing the function
echo "Area with radius 5: " . calculateCircleArea(5) . "\n";   // Expected output: 78.5398
echo "Area with radius -2: " . calculateCircleArea(-2) . "\n"; // Expected output: Invalid radius
echo "Area with radius 0: " . calculateCircleArea(0) . "\n";   // Expected output: Invalid radius
?>
```

PHP Arrays

PHP Arrays: Detailed Explanation with Examples

In PHP, an **array** is a data structure that stores multiple values in a single variable. Arrays are extremely useful because they allow you to organize and manipulate sets of data efficiently. An array can hold values of different data types, including numbers, strings, and even other arrays.

PHP provides various types of arrays, functions, and methods for handling them.

Types of Array in PHP

Types of Arrays in PHP

1. **Indexed Arrays:** Arrays where the keys (indices) are automatically assigned as numeric values, starting from 0.
2. **Associative Arrays:** Arrays where you manually assign keys (which can be strings) to the values.
3. **Multidimensional Arrays:** Arrays containing one or more arrays as their values.

Cont. ...

1. Indexed Arrays

In indexed arrays, the keys are numeric and are automatically assigned by PHP.

Syntax:

```php
$array = array(value1, value2, value3);
```

From PHP 5.4+, you can also use the short array syntax:

```php
$array = [value1, value2, value3];
```

Cont. ...

C:\xampp\htdocs\web\hello.php - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

hello.php

```php
<?php
// Create an indexed array
$fruits = array("Apple", "Banana", "Cherry");

// Accessing elements
echo $fruits[0];  // Outputs: Apple
echo $fruits[1];  // Outputs: Banana

// Adding elements
$fruits[] = "Orange";  // Automatically adds at the next index
echo $fruits[3];  // Outputs: Orange

// Loop through the array
foreach ($fruits as $fruit) {
    echo $fruit . "\n";
}
?>
```

Cont. ...

Explanation:

- The array `$fruits` contains three values (`"Apple"`, `"Banana"`, and `"Cherry"`), indexed at 0, 1, and 2.
- You can access elements using their numeric index.
- Elements can be added dynamically, and they will be indexed automatically.

Cont. ...

2. Associative Arrays

An associative array allows you to assign custom keys to values, which can be more descriptive than numeric keys.

Syntax:

php Copy code

```php
$array = array(key1 => value1, key2 => value2, key3 => value3);
```

Or using the short syntax:

php Copy code

```php
$array = [key1 => value1, key2 => value2, key3 => value3];
```

Cont. ...

C:\xampp\htdocs\web\hello.php - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

hello.php

```php
<?php
// Create an associative array
$person = array("name" => "John", "age" => 30, "city" => "New York");

// Accessing elements
echo $person["name"];  // Outputs: John
echo $person["age"];   // Outputs: 30

// Adding elements
$person["country"] = "USA";
echo $person["country"];  // Outputs: USA

// Loop through the associative array
foreach ($person as $key => $value) {
    echo "$key: $value\n";
}
?>
```

Cont. ...

Explanation:

- The array `$person` uses descriptive keys (`"name"`, `"age"`, and `"city"`) to store values.
- Keys are used instead of numeric indices, which makes the array easier to understand and manage.
- You can add more key-value pairs dynamically.

Cont. ...

3. Multidimensional Arrays

A multidimensional array is an array that contains one or more arrays. These arrays can be of any type (indexed or associative) and can be nested to multiple levels.

Syntax:

php Copy code

```php
$array = array(
    array(value1, value2, value3),
    array(value4, value5, value6)
);
```

Cont. ...

C:\xampp\htdocs\web\hello.php - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

hello.php

```php
<?php
// Create a multidimensional array
$matrix = array(
    array(1, 2, 3),
    array(4, 5, 6),
    array(7, 8, 9)
);

// Accessing elements
echo $matrix[0][1];  // Outputs: 2 (first row, second element)
echo $matrix[2][0];  // Outputs: 7 (third row, first element)

// Loop through the multidimensional array
for ($i = 0; $i < count($matrix); $i++) {
    for ($j = 0; $j < count($matrix[$i]); $j++) {
        echo $matrix[$i][$j] . " ";
    }
    echo "\n";  // New line after each row
}
?>
```

Cont. ...

Explanation:

- The array `$matrix` is a 3x3 matrix where each row is an indexed array.
- You can access elements by providing both the row and column index (e.g., `$matrix[0][1]` accesses the second element of the first row).
- Nested loops are used to iterate over the multidimensional array.

Cont. ...

Common Array Functions in PHP

PHP provides many built-in functions to manipulate arrays. Here are a few commonly used ones:

Function	Description	Example
`count()`	Returns the number of elements in an array.	`count($array)`
`array_merge()`	Merges two or more arrays into one.	`array_merge($array1, $array2)`
`array_push()`	Adds one or more elements to the end of an array.	`array_push($array, "new element")`
`array_pop()`	Removes and returns the last element of an array.	`$value = array_pop($array)`
`in_array()`	Checks if a value exists in an array.	`in_array("value", $array)`
`array_search()`	Searches for a value and returns the key if found.	`array_search("value", $array)`
`array_keys()`	Returns all the keys of an array.	`array_keys($array)`

Cont. ...

`array_values()`	Returns all the values of an array.	`array_values($array)`
`sort()`	Sorts an indexed array in ascending order.	`sort($array)`
`rsort()`	Sorts an indexed array in descending order.	`rsort($array)`
`asort()`	Sorts an associative array in ascending order, according to values.	`asort($array)`
`ksort()`	Sorts an associative array in ascending order, according to keys.	`ksort($array)`
`array_reverse()`	Returns an array with the elements in reverse order.	`array_reverse($array)`

Cont. ...

C:\xampp\htdocs\web\hello.php - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

hello.php

```php
<?php
// Initial array
$numbers = array(4, 6, 2, 22, 11);
// Counting the elements
echo "Number of elements: " . count($numbers) . "\n";  // Outputs: 5
// Sorting the array
sort($numbers);  // Sort in ascending order
echo "Sorted array: ";
print_r($numbers);  // Outputs: Array ( [0] => 2 [1] => 4 [2] => 6 [3] => 11 [4] => 22 )
// Adding elements
array_push($numbers, 33);  // Add 33 to the end
echo "Array after push: ";
print_r($numbers);

// Searching for a value
if (in_array(6, $numbers)) {
    echo "6 is found in the array\n";
}

// Reversing the array
$reversed = array_reverse($numbers);
echo "Reversed array: ";
print_r($reversed);
?>
```

Cont. ...

Explanation:

- The `count()` function returns the number of elements in the array.
- `sort()` sorts the array in ascending order.
- `array_push()` adds a new element to the end of the array.
- `in_array()` checks if a value exists in the array.
- `array_reverse()` returns the reversed array.

Cont. ...

Multidimensional Associative Array Example

You can also create multidimensional arrays with associative arrays, where the inner arrays are key-value pairs.

Explanation:

- Each student has their own associative array (`"age"` and `"grade"` as keys).
- The nested elements can be accessed by specifying both the outer and inner keys (e.g., `$students["John"]["age"]`).
- The loop iterates over the students and prints each student's details.

Cont. ...

C:\xampp\htdocs\web\hello.php - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

hello.php

```php
<?php
$students = array(
    "John" => array("age" => 20, "grade" => "A"),
    "Alice" => array("age" => 22, "grade" => "B"),
    "Bob" => array("age" => 23, "grade" => "C")
);

// Accessing nested associative array elements
echo "John's age: " . $students["John"]["age"] . "\n"."<br>";  // Outputs: 20
echo "Alice's grade: " . $students["Alice"]["grade"] . "\n"."<br>" ;  // Outputs: B

// Loop through the associative array
foreach ($students as $name => $details) {
    echo "$name is " . $details["age"] . " years old and got a grade of " . $details["grade"] ."
        \n"."<br>";
}
?>
```

Thank you!

Appreciate your action.