Module 1 – Core PHP

PHP Syntax

THEORY EXERCISE:

- **Discuss the structure of a PHP script and how to embed PHP in HTML.**

ANS:-

Structure of a PHP Script

```
<?php
// PHP code goes here
// Example: printing text
echo "Hello, World!";
?>
```

Embedding PHP in HTML

PHP is most powerful when embedded inside an HTML page to generate **dynamic content**.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>PHP in HTML</title>
</head>
<body>
<h1>Welcome to My Website</h1>
<p>
<?php
    // Embed PHP inside HTML
    $name = "Riddhi";
    echo "Hello” . $name;
```

```
    ?>
</p>
</body>
</html>
```

- **What are the rules for naming variables in PHP?**

Rules for Naming Variables in PHP

1. Start with a $ sign: Every variable in PHP must begin with $.

Ex: $name = "Riddhi";

$age = 21;

2. Must start with a letter or underscore

Ex: $name, $_value

$1name (cannot start with a number)

3. Can contain letters, numbers, and underscores

Ex: $user1, $user_name, $userName

$user-name, $user name (no hyphens or spaces)

4. Case-sensitive

$name and $Name are different variables.

5. No special characters or keywords

$@test, $class, $echo (invalid or reserved words)

LAB EXERCISE:

- **Write a PHP script to print "Hello, World!" on a web page.**

```
<?php
echo "Hello, World!";
?>
```

3. PHP Variables

THEORY EXERCISE:

Explain the concept of variables in PHP and their scope.

ANS:-

Variables are "containers" for storing information.

A variable in PHP is a container used to store data, such as numbers, strings, arrays, or objects.
A variable always starts with the dollar sign ($).Variable names are case-sensitive.They must start with a letter or underscore (not a number).

Example:

```
<?php
$name = "Riddhi";   // String
$age = 21;          // Integer
$price = 99.50;     // Float
?>
```

Variable Scope in PHP

The scope of a variable determines where in the program it can be accessed. PHP has 4 types of variable scope:

1. Local Scope: A variable declared inside a function is local to that function.It cannot be accessed outside.

2. Global Scope

- A variable declared outside a function has global scope.
- It **cannot** be accessed directly inside functions (unless declared as global).

```
<?php
$y = 20; // global variable

function testGlobal() {
   // echo $y; // Error
```

```
    global $y; // importing global variable inside function
    echo "Inside function: $y <br>";
}
testGlobal();
echo "Outside function: $y"; // Works
?>
```

3. Static Scope

- When a variable is declared as **static inside a function**, it **retains its value** across multiple function calls.
- Unlike normal local variables, it is **not destroyed** after the function ends.

```
<?php
function testStatic() {
    static $count = 0; // initialized only once
    $count++;
    echo "Count: $count <br>";
}
testStatic(); // Count: 1
testStatic(); // Count: 2
testStatic(); // Count: 3
?>
```

4. Superglobals (Global scope everywhere)

- PHP provides some **built-in variables** that are **accessible everywhere** (inside or outside functions).
- Examples:
    - $_GET → for form data via URL
    - $_POST → for form data via POST method
    - $_SESSION, $_COOKIE, $_FILES, $_SERVER, etc.

```
<?php
echo $_SERVER['PHP_SELF']; // prints current file name
```

?>

LAB EXERCISE:

- **Create a PHP script to declare and initialize different types of variables (integer, float,**
  string, boolean). Display them using echo.

```
<?php
$age = 25;
$price = 99.99;
$name = "Riddhi";
$student = true;
echo "Integer: " . $age . "<br>";
echo "Float: " . $price . "<br>";
echo "String: " . $name . "<br>";
echo "Boolean: " . ($student ? "true" : "false") . "<br>";
?>
```

4. Super Global Variables

THEORY EXERCISE:

- **What are super global variables in PHP? List at least five super global arrays and their**
  use.

Super Global Variables in PHP

Superglobals are built-in variables in PHP that are always accessible, regardless of scope (inside functions, classes, or files).
You don't need to use global to access them — they are available everywhere automatically.

1. $_GET

- **Use:** Collects data sent through the **URL query string** (using HTTP GET method).
- **Example:**

  ```
  echo $_GET['name']; // Output: Riddhi
  ```

echo $_GET['age']; // Output: 20

---

2. $_POST

- **Use:** Collects data sent through an **HTML form using POST method** (not visible in URL).
- **Example:**

  // form action="page.php" method="post"

  echo $_POST['username'];

  echo $_POST['password'];

3. $_REQUEST

- **Use:** Collects data from both GET and POST methods, and also cookies.
- **Example:**
- echo $_REQUEST['email'];

4. $_SESSION

- **Use:** Stores and retrieves session variables across multiple pages.
- **Example:**
- session_start();
- $_SESSION['user'] = "Riddhi";
- echo $_SESSION['user']; // Output: Riddhi

5. $_COOKIE

- **Use:** Stores and retrieves values from cookies set on the client's browser.
- **Example:**

  setcookie("user", "Riddhi", time()+3600);

echo $_COOKIE['user']; // Output: Riddhi

LAB EXERCISE:

Create a form that takes a user's name and email. Use the $_POST super global to display the entered data.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Contact Form</title>
</head>
<body>
<h2>Contact Us</h2>
<form action="" method="post">
<label for="name">Name:</label><br>
<input type="text" name="name" id="name" required><br><br>

<label for="email">Email:</label><br>
<input type="email" name="email" id="email" required><br><br>

<input type="submit" value="Submit">
</form>
<?php
if(isset($_POST['submit']))
{
        echo $name=$_POST['name'];
        echo $age=$_POST['email'];
}
```

</body>

</html>

5. Practical Example: Multiple Tables and SQL Queries

LAB EXERCISE:

Create multiple tables and perform queries using:

SELECT, UPDATE, DELETE, INSERT

WHERE, LIKE, GROUP BY, HAVING

LIMIT, OFFSET, Subqueries, AND, OR, NOT, IN

```
-- Create Customers table
CREATE TABLE Customers (
customer_id INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(50),
   email VARCHAR(100),
   city VARCHAR(50)
);
-- Create Orders table
CREATE TABLE Orders (
order_id INT PRIMARY KEY AUTO_INCREMENT,
customer_id INT,
   product VARCHAR(50),
   amount DECIMAL(10,2),
order_date DATE,
   FOREIGN KEY (customer_id) REFERENCES Customers(customer_id)
);
-- Insert customers
INSERT INTO Customers (name, email, city) VALUES
('Riddhi', 'riddhi@gmail.com', 'Ahmedabad'),
(‘Tanisha’,’tanisha@gmail.com’, 'Mumbai'),
```

```
('Priya', 'priya@example.com', 'Delhi'),
('John', 'john@example.com', 'Pune');
-- Insert orders
INSERT INTO Orders (customer_id, product, amount, order_date) VALUES
(1, 'Laptop', 55000, '2025-09-01'),
(2, 'Mobile', 15000, '2025-09-05'),
(3, 'Headphones', 2000, '2025-09-10'),
(1, 'Tablet', 12000, '2025-09-12'),
(2, 'Laptop', 60000, '2025-09-13');
-- SELECT all customers
SELECT * FROM Customers;

-- SELECT with WHERE
SELECT name, city FROM Customers WHERE city = 'Delhi';

-- SELECT with LIKE
SELECT * FROM Customers WHERE name LIKE 'R%';

-- UPDATE a customer’s email
UPDATE Customers SET email = 'riddhi_new@example.com' WHERE name = 'Riddhi';

-- DELETE an order
DELETE FROM Orders WHERE order_id = 3;

-- INSERT a new order
INSERT INTO Orders (customer_id, product, amount, order_date)
VALUES (4, 'Smartwatch', 8000, '2025-09-15');
```

```sql
-- Total amount spent by each customer
SELECT customer_id, SUM(amount) AS total_spent
FROM Orders
GROUP BY customer_id;

-- Customers who spent more than 20000
SELECT customer_id, SUM(amount) AS total_spent
FROM Orders
GROUP BY customer_id
HAVING SUM(amount) > 20000;

--Limit
SELECT * FROM Customers LIMIT 2;

--Subqueries
-- Customers who placed orders above 50000
SELECT name
FROM Customers
WHERE customer_id IN (
   SELECT customer_id
   FROM Orders
   WHERE amount > 50000
);

Logical Operators (AND, OR, NOT, IN)
-- Customers from Delhi OR Mumbai
SELECT * FROM Customers WHERE city = 'Delhi' OR city = 'Mumbai';
```

-- Customers not from Delhi

SELECT * FROM Customers WHERE NOT city = 'Delhi';

-- Customers from multiple cities using IN

SELECT * FROM Customers WHERE city IN ('Delhi', 'Pune');

-- Orders with amount between 10000 AND 60000

SELECT * FROM Orders WHERE amount >= 10000 AND amount <= 60000;

6. Conditions, Events, and Flows

THEORY EXERCISE:

Explain how conditional statements work in PHP.

Conditional statements in PHP are used to make decisions in the code. They allow you to execute different blocks of code depending on whether a condition is true or false.

conditional statement have5 types

1. if statementstatement:Executes code only if the condition is true.

2. if else statement: Executes one block if the condition is true, otherwise another block.

3. if elseif else statement:Used when you have **multiple conditions** to check, one after the other.

4. nested if statement: An if statement placed inside another if.Useful when one condition depends on another.

5. switch statement: Used when you want to test **one variable against multiple values**.Makes the code cleaner than writing many if...elseif.

7. If Condition and If-Else If

LAB EXERCISE:

Write a PHP program to determine if a number is even or odd using if conditions.

<?php

$number = 15;

```
if ($number % 2 == 0) {
   echo "$number is Even";
} else {
   echo "$number is Odd";
}
?>
```

8. Practical Example: Calculator and Day Finder

LAB EXERCISE:

1. Simple Calculator: Create a calculator using if-else conditions that takes two inputs and an operator (+, -, *, /).

```
<!DOCTYPE html>
<html>
<head>
<title>Simple PHP Calculator</title>
</head>
<body>
<h2>PHP Calculator</h2>
<form method="post" action="">
<label>Enter First Number:</label>
<input type="number" name="num1" required><br><br>
<label>Enter Second Number:</label>
<input type="number" name="num2" required><br><br>
<label>Choose Operator (+, -, *, /):</label>
<input type="text" name="operator" required><br><br>
<input type="submit" value="Calculate">
</form>
```

```php
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $num1 = $_POST['num1'];
    $num2 = $_POST['num2'];
    $op = $_POST['operator'];
    if ($op == "+") {
        $result = $num1 + $num2;
        echo "<h3>Result: $num1 + $num2 = $result</h3>";
    } elseif ($op == "-") {
        $result = $num1 - $num2;
        echo "<h3>Result: $num1 - $num2 = $result</h3>";
    } elseif ($op == "*") {
        $result = $num1 * $num2;
        echo "<h3>Result: $num1 * $num2 = $result</h3>";
    } elseif ($op == "/") {
        if ($num2 != 0) {
            $result = $num1 / $num2;
            echo "<h3>Result: $num1 / $num2 = $result</h3>";
        } else {
            echo "<h3>Error: Division by zero is not allowed.</h3>";
        }
    } else {
        echo "<h3>Invalid Operator! Please use +, -, * or /.</h3>";
    }
}
?>
</body>
</html>
```

2. Day Finder: Write a script that finds the current day. If it is Sunday, print "Happy Sunday."

```
<?php
$day = date("l");
if ($day == "Sunday") {
   echo "Happy Sunday.";
} else {
   echo "Today is $day.";
}
?>
```

9. Switch Case and Ternary Operator

LAB EXERCISE:

1. Restaurant Food Category Program: Use a switch case to display the category (Starter/Main Course/Dessert) and dish based on user selection.

```
<!DOCTYPE html>
<html>
<head>
<title>Restaurant Food Category</title>
</head>
<body>
<h2>Restaurant Menu</h2>
<form method="post" action="">
<label for="choice">Enter your choice (1-3):</label><br>
<b>1. Starter</b><br>
<b>2. Main Course</b><br>
```

```
<b>3. Dessert</b><br><br>
<input type="number" name="choice" min="1" max="3" required>
<input type="submit" value="Show Dish">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
   $choice = $_POST['choice'];
   switch ($choice) {
      case 1:
         echo "<h3>Category: Starter</h3>";
         echo "Dish: Spring Rolls";
         break;
      case 2:
         echo "<h3>Category: Main Course</h3>";
         echo "Dish: Paneer Butter Masala with Naan";
         break;
      case 3:
         echo "<h3>Category: Dessert</h3>";
         echo "Dish: Chocolate Ice Cream";
         break;
      default:
         echo "<h3>Invalid choice. Please select between 1 and 3.</h3>";
   }
}
?>
</body>
</html>
```

2. Ternary Operator Example: Write a script using the ternary operator to display a message if the age is greater than 18.

```
<?php
$age = 20;
$message = ($age > 18) ? "You are above 18." : "You are 18 or below.";
echo $message;
?>
```

3. Color Selector: Write a program to display the name of a color based on user input (red, green, blue).

```
<!DOCTYPE html>
<html>
<head>
<title>Color Display Program</title>
</head>
<body>
<h2>Enter a Color (red, green, blue):</h2>
<form method="post" action="">
<input type="text" name="color" required>
<input type="submit" value="Show Color">
</form>

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $color = strtolower($_POST['color']); // convert input to lowercase
    if ($color == "red") {
```

```
        echo "<h3>You selected: Red</h3>";
    } elseif ($color == "green") {
        echo "<h3>You selected: Green</h3>";
    } elseif ($color == "blue") {
        echo "<h3>You selected: Blue</h3>";
    } else {
        echo "<h3>Invalid color! Please enter red, green, or blue.</h3>";
    }
}
?>
</body>
</html>
```

10. Loops: Do-While, For Each, For Loop

THEORY EXERCISE:

Discuss the difference between for loop, foreach loop, and do-while loop in PHP.

Feature	**for loop**	**foreach loop**	**do-while loop**
Primary Use	Fixed number of iterations	Iterating arrays/objects	Run at least once, then check condition
Condition Check	Before each iteration	Implicit (loops through elements)	After each iteration
Initialization	Required	Not needed	Not needed
Works with	Numbers / ranges	Arrays / objects	Any condition
Guaranteed Run?	No (if condition false initially)	Yes, if array not empty	Yes, at least once

LAB EXERCISE:

1. For Loop: Write a script that displays numbers from 1 to 10 on a single line.

```php
<?php
for ($i = 1; $i<= 10; $i++) {
   echo $i . " ";
}
?>
```

2. For Loop (Addition): Add all integers from 0 to 30 and display the total.

```php
<?php
$total = 0;

for ($i = 0; $i<= 30; $i++) {
   $total += $i; // add each number to total
}
echo "The sum of integers from 0 to 30 is: " . $total;
?>
```

3. Chessboard Pattern: Use a nested loop to create a chessboard.

```php
<?php
echo "<table border='1' cellspacing='0' cellpadding='20'>";

for ($row = 1; $row <= 8; $row++) {
   echo "<tr>";
   for ($col = 1; $col <= 8; $col++) {

      if (($row + $col) % 2 == 0) {
         echo "<td bgcolor='white'></td>";
      } else {
         echo "<td bgcolor='black'></td>";
      }
```

```
    }
    echo "</tr>";
}

echo "</table>";
?>
```

11. PHP Array and Array Functions

THEORY EXERCISE:

Define arrays in PHP. What are the different types of arrays?

An **array** is a special variable in PHP that can hold **multiple values** under a single name.

Instead of creating separate variables for each value, we can store them in an array and access them using **indexes** or **keys**.

Example:

```
$fruits = ["Apple", "Banana", "Mango"];
echo $fruits[0];
```

1. Numeric (Indexed) Arrays

Arrays with **numeric indexes** (starting from 0 by default).

Useful for storing ordered data.

Example:

```
$colors = ["Red", "Green", "Blue"];
echo $colors[1]; // Output: Green
```

2. Associative Arrays

Arrays that use **named keys** instead of numbers.

Useful when you want to map **key → value** pairs.

Example:

```
$student = [
```

```
    "name" => "Riddhi",
    "age" => 21,
    "course" => "PHP"
];
echo $student["name"]; // Output: Riddhi
```

3. Multidimensional Arrays

Arrays containing one or more **arrays inside them**.

Useful for storing **tabular data** (like matrices or records).

Example:

```
$marks = [
    ["Riddhi", "Maths", 90],
    ["Ankit", "Maths", 85],
    ["Priya", "Maths", 92]
];
echo $marks[0][0]; // Output: Riddhi
echo $marks[2][2]; // Output: 92
```

LAB EXERCISE:

1. Display the value of an array.

```
<?php
$fruits = ["Apple", "Banana", "Mango", "Orange"];
foreach ($fruits as $fruit) {
    echo $fruit . "<br>";
}
?>
```

2. Find and display the number of odd and even elements in an array.

```php
<?php
$numbers = [2, 5, 8, 11, 14, 17, 20, 23, 26];
$evenCount = 0;
$oddCount = 0;
foreach ($numbers as $num) {
   if ($num % 2 == 0) {
      $evenCount++;
   } else {
      $oddCount++;
   }
}
echo "Total Even Numbers: " . $evenCount . "<br>";
echo "Total Odd Numbers: " . $oddCount;
?>
```

3. Create an associative array for user details (name, email, age) and display them.

```php
<?php
$user = [
   "name"  => "Riddhi Gandharva",
   "email" => "riddhi@example.com",
   "age"   => 21
];
echo "Name: " . $user["name"] . "<br>";
echo "Email: " . $user["email"] . "<br>";
echo "Age: " . $user["age"];
?>
```

4. Write a script to shift all zero values to the bottom of an array.

```php
<?php
$numbers = [1, 0, 5, 0, 9, 0, 7, 3, 0, 4];
$nonZero = [];
$zeros = [];
foreach ($numbers as $num) {
   if ($num == 0) {
      $zeros[] = $num;  // store zero
   } else {
      $nonZero[] = $num;  // store non-zero
   }
}
$result = array_merge($nonZero, $zeros);
echo "Original Array: ";
print_r($numbers);
echo "<br>Modified Array: ";
print_r($result);
?>
```

12. PHP Date-Time Function

LAB EXERCISE:

Write a script to display the current date and time in different formats.

```php
<?php
echo "<h2>Current Date and Time in Different Formats</h2>";
echo "Format 1: " .date("d/m/Y H:i:s") . "<br>";
echo "Format 2: " .date("m-d-Y h:i A") . "<br>";
echo "Format 3: " .date("l, jS F Y") . "<br>";
echo "Format 5: Today is " . date("l") . ", and the current time is " .date("h:i A") . "<br>";
?>
```

13. Header Function

THEORY EXERCISE:

What is the header function in PHP and how is it used?

The header() function in PHP is used to send raw HTTP headers to the browser before any actual output is sent.

It's commonly used to:

- Redirect users to another page
- Set content type (e.g., JSON, PDF, etc.)
- Control caching behavior
- Force file downloads

LAB EXERCISE:

Redirect users to another page using the header() function.

```
<?php

header("Location: welcome.php");

exit();

?>
```

14. Include and Require

THEORY EXERCISE:

Explain the difference between include and require in PHP.

1. **include Statement**

Purpose: Includes and evaluates the specified file.

Behavior on Error:

If the file is not found, PHP gives a warning, but the script continues to run.

2**. require Statement**

Purpose: Works like include, but it requires the file for the script to run.

Behavior on Error:

If the file is not found, PHP gives a fatal error, and the script stops executing.

LAB EXERCISE:

Use include and require to insert common header and footer files into multiple PHP pages.

Header.php

```
<!DOCTYPE html>
<html>
<head>
<title>My PHP Website</title>
</head>
<body>
<header style="background-color:lightblue; padding:10px;">
<h2>Welcome to My Website</h2>
<nav>
<a href="home.php">Home</a> |
<a href="about.php">About</a>
</nav>
</header>
<hr>
```

Footer.php

```
<hr>
<footer style="background-color:lightgray; padding:10px;">
<p>&copy; 2025 My Website. All rights reserved.</p>
</footer>
</body>
```

```
</html>
```

Home.php

```
<?php
include("header.php");
?>
<h3>Home Page</h3>
<p>Welcome to the homepage of our PHP website!</p>
<?php
include("footer.php");
?>
```

About.php

```
<?php
?>
<h3>About Us</h3>
<p>This page contains information about our website.</p>
<?php
require("footer.php");
?>
```

15. Practical Example: Calculator, Factorial, String Reverse

LAB EXERCISE:

1. Calculator: Create a calculator using user-defined functions.

```
<?php
function add($a, $b) {
   return $a + $b;
}
```

```
function subtract($a, $b) {
    return $a - $b;
}
function multiply($a, $b) {
    return $a * $b;
}
function divide($a, $b) {
    if ($b == 0) {
        return "Error! Division by zero.";
    }
    return $a / $b;
}

if (isset($_POST['calculate'])) {
    $num1 = $_POST['num1'];
    $num2 = $_POST['num2'];
    $operator = $_POST['operator'];

    switch ($operator) {
        case 'add':
            $result = add($num1, $num2);
            break;
        case 'sub':
            $result = subtract($num1, $num2);
            break;
        case 'mul':
            $result = multiply($num1, $num2);
            break;
```

```
        case 'div':
            $result = divide($num1, $num2);
            break;
        default:
            $result = "Invalid Operation!";
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Calculator</title>
</head>
<body>
    <h2>Simple Calculator</h2>
    <form method="post">
        <input type="number" name="num1" required placeholder="Enter first number">
        <input type="number" name="num2" required placeholder="Enter second number">
        <select name="operator">
            <option value="add">Add</option>
            <option value="sub">Subtract</option>
            <option value="mul">Multiply</option>
            <option value="div">Divide</option>
        </select>
        <input type="submit" name="calculate" value="Calculate">
    </form>
```

```
    <?php if (isset($result)) echo "<h3>Result: $result</h3>"; ?>
</body>
</html>
```

2. Factorial: Write a function that finds the factorial of a number using recursion.

```
<?php
function factorial($n) {
    if ($n == 0 || $n == 1)
        return 1;
    else
        return $n * factorial($n - 1);
}

if (isset($_POST['find'])) {
    $num = $_POST['num'];
    $result = factorial($num);
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Factorial Using Recursion</title>
</head>
<body>
    <h2>Find Factorial</h2>
    <form method="post">
        <input type="number" name="num" required placeholder="Enter a number">
        <input type="submit" name="find" value="Find Factorial">
```

```
    </form>

    <?php if (isset($result)) echo "<h3>Factorial of $num is: $result</h3>"; ?>
</body>
</html>
```

3. String Reverse: Reverse a string without using built-in functions.

```
<?php
function reverseString($str) {
    $rev = '';
    for ($i = strlen($str) - 1; $i >= 0; $i--) {
        $rev .= $str[$i];
    }
    return $rev;
}

if (isset($_POST['reverse'])) {
    $input = $_POST['input'];
    $result = reverseString($input);
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Reverse String</title>
</head>
<body>
    <h2>Reverse a String</h2>
```

```
  <form method="post">
    <input type="text" name="input" required placeholder="Enter a string">
    <input type="submit" name="reverse" value="Reverse">
  </form>

  <?php if (isset($result)) echo "<h3>Reversed String: $result</h3>"; ?>
</body>
</html>
```

4. Download File: Create a button that allows users to download a file.

```
<!DOCTYPE html>
<html>
<head>
  <title>Download File</title>
</head>
<body style="text-align:center; margin-top:50px;">
  <h2>Click the button to download a file</h2>
  <a href="example.txt" download>
    <button style="padding:10px 20px; font-size:16px;">Download File</button>
  </a>
</body>
</html>
```

16. PHP Expressions, Operations, and String Functions

THEORY EXERCISE:

Explain what PHP expressions are and give examples of arithmetic and logical operations.

What are PHP Expressions?

An **expression** in PHP is anything that has a **value**.

It can be a **variable**, **constant**, or a **combination of values, operators, and functions** that results in a value.

Example:

```
$x = 5 + 10;  // Expression: 5 + 10

$y = $x * 2;  // Expression: $x * 2
```

Arithmetic Operations

Arithmetic operators are used to perform mathematical calculations.

Operator	Description	Example	Result
+	Addition	$a + $b	Sum of $a and $b
-	Subtraction	$a - $b	Difference
*	Multiplication	$a * $b	Product
/	Division	$a / $b	Quotient
%	Modulus	$a % $b	Remainder

Example:

```
$a = 10;

$b = 3;

echo $a + $b; // 13

echo $a - $b; // 7

echo $a * $b; // 30

echo $a / $b; // 3.333...

echo $a % $b; // 1
```

Logical Operations

Logical operators are used to combine conditional statements.

Operator	Description	Example
&&	AND	($a > 0 && $b > 0)
`		`
!	NOT	!($a > 0)

Example:

```
$x = 10;
$y = 5;

if ($x > 0 && $y > 0) {
   echo "Both are positive numbers";
}
```

- **LAB EXERCISE:**

  Write a script to perform various string operations like concatenation, substring extraction, and string length determination.

```
<?php
// String Operations Example

// 1. String Concatenation
$str1 = "Hello";
$str2 = "World";
$concat = $str1 . " " . $str2;
echo "<b>Concatenation:</b> " . $concat . "<br>";
```

// 2. Substring Extraction

```
$substring = substr($concat, 0, 5); // Extracts "Hello"
echo "<b>Substring:</b> " . $substring . "<br>";
```

// 3. String Length

```
$length = strlen($concat);
echo "<b>String Length:</b> " . $length . "<br>";
```

// 4. String Uppercase and Lowercase

```
echo "<b>Uppercase:</b> " . strtoupper($concat) . "<br>";
echo "<b>Lowercase:</b> " . strtolower($concat) . "<br>";
```

// 5. Replacing part of a string

```
$replace = str_replace("World", "PHP", $concat);
echo "<b>After Replacement:</b> " . $replace . "<br>";
?>
```

OUTPUT:

Concatenation: Hello World

Substring: Hello

String Length: 11

Uppercase: HELLO WORLD

Lowercase: hello world

After Replacement: Hello PHP