Module 1 – Core PHP

PHP Syntax

THEORY EXERCISE:

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

Ans. A PHP script is structured around HTML content, with embedded PHP code enclosed in tags. The script is executed on the server, generates HTML output, and sends it to the client's browser. PHP code within the tags can handle database interactions, form submissions, and other server-side logic.

Here's a more detailed breakdown:

1. HTML and PHP Code:

* A PHP script typically contains a mix of standard HTML tags for page layout and presentation, and PHP code for dynamic content generation.
* PHP code is embedded within the HTML using PHP tags: <?php ?>.

2. PHP Tags:

* **Standard tags:** <?php ?> is the most common and recommended way to delimit PHP code.
* **Short tags:** <? ?> can be used, but they are not supported on all servers and may cause compatibility issues.
* **ASP-style tags:** <% %> can be used if enabled in the PHP configuration file, but are less common.

3. PHP Statements:

* PHP scripts are made up of a sequence of statements, including assignments, function calls, loops, conditional statements, and empty statements.
* Most statements are terminated with a semicolon (;).
* Groups of statements can be enclosed in curly braces ({}) to form statement groups.

4. Execution:

* The PHP script is executed on the server, and the generated HTML output is sent back to the browser.
* The client browser displays the HTML content, not the underlying PHP code.

5. Server-Side Logic:

* PHP code is used for tasks like:
  + Handling form submissions.
  + Interacting with databases.
  + Processing data.
  + Generating dynamic content.

<html>

<head>

<title>PHP Example</title>

</head>

<body>

<?php

$name = "World";

echo "Hello, $name!";

?>

</body>

</html>

**2. What are the rules for naming variables in PHP?**

Ans. **PHP Variables**

* A variable starts with the $ sign, followed by the name of the variable.
* A variable name must start with a letter or the underscore character.
* A variable name cannot start with a number.
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )

LAB EXERCISE:

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

Ans.

<?php

echo “Hello, World!”;

?>

**3. PHP Variables**

**THEORY EXERCISE:**

**Q - Explain the concept of variables in PHP and their scope.**

Ans.

**Variable Scopes:**The scope of a variable is defined as its extent in the program within which it can be accessed, i.e. the scope of a variable is the portion of the program within which it is visible or can be accessed. Depending on the scopes, PHP has three variable scopes.

**Local variables:** The variables declared within a function are called local variables to that function and have their scope only in that particular function. In simple words, it cannot be accessed outside that function. Any declaration of a variable outside the function with the same name as that of the one within the function is a completely different variable. For now, consider a function as a block of statements.

**Global variables:** The variables declared outside a function are called global variables. These variables can be accessed directly outside a function. To get access within a function, we need to use the “global” keyword before the variable to refer to the global variable.

**Static variable**: It is the characteristic of PHP to delete the variable, once it completes its execution and the memory is free. But sometimes we need to store the variables even after the completion of function execution. To do this, we use static keywords and the variables are called static variables. PHP associates a data type depending on the value for the variable.

**LAB EXERCISE:**

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

Ans.

|  |  |  |  |
| --- | --- | --- | --- |
| **Integer** | **Float** | **String** | **Boolean** |
| <?php  $a = 5;  echo “$a”;  ?> | <?php  $b=6.3;  echo “$b”;  ?> | <?php  $c=”Hello!”;  echo “$c”;  ?> | <?php  $d=true;  echo ”$d”;  ?> |

**4. Super Global Variables**

**THEORY EXERCISE:**

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

Ans. Super global variables in PHP are predefined global variables that are always accessible from any part of a script, without needing to be declared global or passed as arguments to functions. They provide convenient access to information about the server, HTTP requests, cookies, sessions, and files.

Here are five super global arrays and their common uses:

**1. $\_SERVER:**

This array contains information about the server, such as the server's name, the request method (GET, POST), the current script's URL, and more. For example, $\_SERVER['SERVER\_NAME'] would return the hostname of the server.

**2. $\_GET:**

This array holds data sent to the script via the HTTP GET method, typically through query parameters in the URL. For example, $\_GET['param'] would access a parameter named "param" passed in the URL.

**3. $\_POST:**

This array contains data sent to the script via the HTTP POST method, usually from HTML forms. For example, $\_POST['username'] would access the value of the "username" field submitted in a POST request.

**4. $\_REQUEST:**

This array combines both $\_GET and $\_POST data, providing a single access point for data received via either HTTP method.

**5. $\_SESSION:**

This array stores session-specific data, allowing you to maintain information across multiple pages within a user's session. For example, $\_SESSION['loggedin'] could store a flag indicating whether a user is currently logged in.

**LAB EXERCISE:**

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

**1. User Details Forms**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=, initial-scale=1.0">

    <title>User Details</title>

</head>

<body>

    <h2> User Details</h2>

    <form action="submit.php" method="post">

    <lable for="username"> Username: </lable>

    <input type="text" name="username" id="" required></br></br>

    <lable for="email"> Email: </lable>

    <input type="email" name="username" id="" required></br></br>

    <input type="button" value="Submit"/>

    </form>

</body>

</html>

2. Submit page

<?php

if($\_SERVER["REQUEST\_METHOD"] == "POST"){

$username = $\_POST("username");

$email = $\_POST("email";)

if(!empty($username)&&!empty($email)){

    echo "User details submitted succesfully!!!";

    echo "User Name: ".htmlspecialchars($username)."<br/">;

    echo "Email ID: ".htmlspecialchars($email)."<br/">;

}

else{

    echo "Please fill all required fields";

}

}

else{

 echo "Invalid Request Method!!";

}

?>

**5. Practical Example:**

**Multiple Tables and SQL Queries**

**LAB EXERCISE:**

**• Create multiple tables and perform queries using:**

SELECT

select \* from students;

UPDATE

update student set student\_name = ‘Ayansh’ where stud\_id=2;

DELETE

delete from student where stud\_id=2;

INSERT

insert into student(stud\_id, student\_name, age, grade) values (5,’Raj’,15,’A’);

WHERE

select \* from student where stud\_id = 2;

LIKE

select \* from student where student\_name like ‘A%’;

GROUP BY

Select grade, COUNT(\*) AS total\_students from student

GROUP BY grade;

HAVING

select grade, COUNT(\*) AS total\_students

from student

GROUP BY grade

HAVING COUNT(\*) > 2;

LIMIT

SELECT \* FROM student LIMIT 5;

OFFSET

SELECT \* FROM student LIMIT 5 OFFSET 5;

Subqueries

SELECT \* FROM student WHERE age > (SELECT AVG(age) FROM student);

AND

SELECT \* FROM student WHERE age > 18 AND grade = 'A';

OR

SELECT \* FROM student WHERE age < 20 OR grade = 'B';

NOT

SELECT \* FROM student WHERE NOT age > 18;

IN

SELECT \* FROM student WHERE grade IN ('A', 'B', 'C');

6. Conditions, Events, and Flows

THEORY EXERCISE:

**Q - Explain how conditional statements work in PHP.**

Ans

Conditional Statements are used to perform actions based on different conditions. Sometimes when we write a program, we want to perform some different actions for different actions. We can solve this by using conditional statements.

**In PHP we have these conditional statements:**

1. if Statement.
2. if-else Statement
3. If-elseif-else Statement
4. Switch statement

**1. if Statement**

This statement executes the block of code inside the if statement if the expression is evaluated as True.

**Example:**

<?php

$x = "22";

if ($x < "20") {

echo "Hello World!";

}

?>

**2. if-else Statement**

This statement executes the block of code inside the if statement if the expression is evaluated as True and executes the block of code inside the else statement if the expression is evaluated as False.

**Example:**

<?php

$x = "22";

if ($x < "20") {

echo "Less than 20";

} else {

echo "Greater than 20";

}

?>

**3. If-else-if-else**

This statement executes different expressions for more than two conditions.

**Example:**

**<?php**

$x = "22";

if ($x == "22") {

echo "correct guess";

} else if ($x < "22") {

echo "Less than 22";

} else {

echo "Greater than 22";

}

**?>**

**4. Switch Statement**

This statement allows us to execute different blocks of code based on different conditions. Rather than using if-elseif-if, we can use the switch statement to make our program.

**Example:**

<?php

$i = "2";

switch ($i) {

case 0:

echo "i equals 0";

break;

case 1:

echo "i equals 1";

break;

case 2:

echo "i equals 2";

break;

default:

echo "i is not equal to 0, 1 or 2";

}

?>

**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.**

Ans.

<?php

$number = readline(“Enter the number”);

if ($number % 2 == 0) {

echo "$number is an even number.";

} else {

echo "$number is an odd number.";

}

?>

**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 (+, -, \*, /).**

Ans.

<?php

$num1 = readline("Enter the First Number\n");

$num2 = readline("Enter the Second Number \n");

$operator = "+"; // Change this to -, \*, or /

if ($operator == "+") {

    $result = $num1 + $num2;

    echo "Result: $num1 + $num2 = $result";

} elseif ($operator == "-") {

    $result = $num1 - $num2;

    echo "Result: $num1 - $num2 = $result";

} elseif ($operator == "\*") {

    $result = $num1 \* $num2;

    echo "Result: $num1 \* $num2 = $result";

} elseif ($operator == "/") {

    if ($num2 != 0) {

        $result = $num1 / $num2;

        echo "Result: $num1 / $num2 = $result";

    } else {

        echo "Error: Division by zero is not allowed.";

    }

} else {

    echo "Invalid operator.";

}

?>

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

<?php

$day = date("l");

echo "Today is: $day";

echo "\n";

if ($day == "Sunday") {

    echo "Happy Sunday.";

}

?>

**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.**

Ans.

<?php

$dish = "Pizza"; // Try "Soup", "Pizza", "Ice Cream", etc.

echo "Selected Dish: $dish";

switch ($dish) {

case "Soup":

case "Salad":

echo "Category: Starter";

echo "Dish: $dish";

break;

case "Pizza":

case "Burger":

case "Pasta":

echo "Category: Main Course";

echo "Dish: $dish";

break;

case "Ice Cream":

case "Cake":

case "Pudding":

echo "Category: Dessert";

echo "Dish: $dish";

break;

default:

echo "Sorry, dish not found in our menu.";

}

?>

**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 eligible." : "You are not eligible.";

echo $message;

?>

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

<?php

$color = readline("Enter a Color:");

$color = strtolower($color);

switch($color){

    case "red":

        echo "You Selected: Red";

        break;

    case "green":

        echo "You Selected: Green";

        break;

    case " blue ":

        echo "You Selected: Black";

        break;

        default:

        echo "Color not recognized. Please choose red, green, or blue.";

}

?>

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

**THEORY EXERCISE:**

**Q - Discuss the difference between for loop, for each loop, and do-while loop in PHP.**

Ans.

**For Loop,** **While Loop**, and **Do-While** **Loop**are different [loops in programming](https://www.geeksforgeeks.org/loops-programming/). A For loop is used when the number of iterations is known. A While loop runs as long as a condition is true. A Do-While loop runs at least once and then continues if a condition is true.

| **Feature** | **for Loop** | **while Loop** | **do-while Loop** |
| --- | --- | --- | --- |
| **Syntax** | for (initialization; condition; increment/decrement) {} | while (condition) { } | do { } while (condition); |
| **Initialization** | Declared within the loop structure and executed once at the beginning. | Declared outside the loop; should be done explicitly before the loop. | Declared outside the loop structure |
| **Condition** | Checked before each iteration. | Checked before each iteration. | Checked after each iteration. |
| **Update** | Executed after each iteration. | Executed inside the loop; needs to be handled explicitly. | Executed inside the loop; needs to be handled explicitly. |
| **Use Cases** | Suitable for a known number of iterations or when looping over ranges. | Useful when the number of iterations is not known in advance or based on a condition. | Useful when the loop block must be executed at least once, regardless of the initial condition. |
| **Initialization and Update Scope** | Limited to the loop body. | Scope extends beyond the loop; needs to be handled explicitly. | Scope extends beyond the loop; needs to be handled explicitly. |

**LAB EXERCISE:**

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

For($i=1;$i<=10;$i++){

echo $i “ ”;

}

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

$sum = 0;

for ($i = 0; $i <= 30; $i++) {

    $sum += $i;

}

echo "The total sum from 0 to 30 is: $sum";

**3. Chess board Pattern: Use a nested loop to create a chess board pattern(8x8 grid)**

<!DOCTYPE html>

<html>

<head>

    <style>

        table {

            border-collapse: collapse;

        }

        td {

            width: 50px;

            height: 50px;

        }

        .white {

            background-color: #fff;

        }

        .black {

            background-color: #000;

        }

    </style>

</head>

<body>

<?php

echo "<table border='1'>";

for ($row = 1; $row <= 8; $row++) {

    echo "<tr>";

    for ($col = 1; $col <= 8; $col++) {

        // Sum of row and column determines color

        $colorClass = ($row + $col) % 2 == 0 ? "white" : "black";

        echo "<td class='$colorClass'></td>";

    }

    echo "</tr>";

}

echo "</table>";

?>

</body>

</html>

**4. Various Patterns: Generate different patterns using loops.**

1. Pattern

for ($i = 1; $i <= 10; $i++){

    for($j = 1; $j <=$i; $j++){

        echo "\* ";

    }

    echo"\n";

}

2. Pattern

<?php

for ($i = 5; $i >= 1; $i--) {

    for ($j = 1; $j <= $i; $j++) {

        echo "\* ";

    }

    echo "\n";

}

?>

**11.PHP Array and Array Functions**

**THEORY EXERCISE**

**Q - Define arrays in PHP. What are the different types of arrays.**

Ans. Arrays are one of the most important data structures in PHP. They allow you to store multiple values in a single variable. PHP arrays can hold values of different types, such as strings, numbers, or even other arrays. Understanding how to use arrays in PHP is important for working with data efficiently.

* PHP offers a wide range of built-in array functions for sorting, merging, searching, and more.
* PHP Arrays can store values of different types (e.g., strings, integers, objects, or even other arrays) in the same array.
* They are dynamically sized.
* They allow you to store multiple values in a single variable, making it easier to manage related data.

**Types of Arrays in PHP**

There are three main types of arrays in [PHP](https://www.geeksforgeeks.org/php-introduction/):

**1. Indexed Arrays**

Indexed arrays use numeric indexes starting from 0. These arrays are ideal when you need to store a list of items where the order matters.

**Now, let us understand with the help of the example:**

<?php

$fruits = **array**("apple", "banana", "cherry");

**echo** $fruits[0]; *// Outputs: apple*

?>

**Output**

apple

You can also explicitly define numeric keys in an indexed array:

<?php

$fruits = **array**(0 => "apple", 1 => "banana", 2 => "cherry");

?>

**2. Associative Arrays**

Associative arrays use named keys, which are useful when you want to store data with meaningful identifiers instead of numeric indexes.

**Now, let us understand with the help of the example:**

<?php

$person = **array**("name" => "GFG", "age" => 30, "city" => "New York");

**echo** $person["name"];

?>

**Output**

GFG

**3. Multidimensional Arrays**

Multidimensional arrays are arrays that contain other arrays as elements. These are used to represent more complex data structures, such as matrices or tables.

**Now, let us understand with the help of the example:**

<?php

$students = **array**(

"Anjali" => **array**("age" => 25, "grade" => "A"),

"GFG" => **array**("age" => 22, "grade" => "B")

);

**echo** $students["GFG"]["age"];

?>

**LAB EXERCISE:**

**1. Display the value of an array**.

<?php

$array = ["T-shirt","Shirt","Pant"];

echo implode(" ",$array);

?>

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

Ans.

<?php

$numbers = ["2","4","9","100"];

$odddigit = 0;

$evendigit = 0;

foreach ($numbers as $num) {

if($num % 2 ==0)

{

  $odddigit++;

}

else{

  $evendigit++;

} }

echo "Even numbers: " . $evendigit . "\n";

echo "Odd numbers: " . $odddigit . "\n";

?>

Output:

Even numbers: 1

Odd numbers: 3

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

Ans.

$user = array("name"=>"Gargi","email"=>"gargigodia@gmail.com","age"=>25);

echo implode(" ",$user);

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

Ans.

$array = [0,15, 0, 31, 0, 12, 92, 0, 2];

$nonZeros = [];

$zeroCount = 0;

foreach ($array as $value) {

    if ($value === 0) {

        $zeroCount++;

    } else {

        $nonZeros[] = $value;

    }

}

// Add zeros to the end

for ($i = 0; $i < $zeroCount; $i++) {

    $nonZeros[] = 0;

}

// Output the result

echo "Array after shifting zeros to the bottom:\n";

print\_r($nonZeros);

**Extra LAB EXERCISES for Core PHP**

**1. PHP Syntax**

**Extra LAB EXERCISES:**

**PHP Comments: Write a PHP script that demonstrates the use of single-line(//), multi-line(/\* \*/),and** **inline(#)comments.**

//Output Result

/\*This is a error

Report\*/

$number = 20; # Set a number variable

• Embedding HTML and PHP: Create a webpage that uses PHP to dynamically generate HTML content (e.g., a table with user information using PHP).

<!DOCTYPE html>

<html>

<head>

<title>User Information Table</title>

<style>

table {

border-collapse: collapse;

width: 60%;

margin: 20px auto;

}

th, td {

border: 1px solid #444;

padding: 10px;

text-align: left;

}

th {

background-color: #ddd;

}

</style>

</head>

<body>

<h2 style="text-align:center;">User Information</h2>

<?php

$users = [

['name' => ‘Gargi’, 'email' => ‘gargigodia@gmail.com', 'age' => 28],

['name' => 'priya', 'email' => 'priya@yahoo.com', 'age' => 34],

['name' => 'ram', 'email' => 'ram@live.com', 'age' => 22],

];

echo "<table>";

echo "<tr><th>Name</th><th>Email</th><th>Age</th></tr>";

foreach ($users as $user) {

echo "<tr>";

echo "<td>" . htmlspecialchars($user['name']) . "</td>";

echo "<td>" . htmlspecialchars($user['email']) . "</td>";

echo "<td>" . htmlspecialchars($user['age']) . "</td>";

echo "</tr>";

}

echo "</table>";

?>

</body>

</html>

Output Statements : Experiment with echo, print, and var\_dump. Write a script that outputs different types of data using these functions.

<?php

// String

$name = "Gargi";

// Integer

$age = 25;

// Float

$height = 5.7;

// Boolean

$isStudent = true;

// Array

$skills = ["PHP", "HTML", "CSS"];

// Object

class Person {

public $firstName = "Ram";

public $lastName = "Ayansh";

}

$person = new Person();

// Using echo

echo "Using echo:<br>";

echo "Name: $name<br>";

echo "Age: $age<br><br>";

// Using print

print "Using print:<br>";

print "Height: $height<br>";

print "Is student: " . ($isStudent ? 'Yes' : 'No') . "<br><br>";

// Using var\_dump

echo "Using var\_dump:<br>";

var\_dump($name);

echo "<br>";

var\_dump($age);

echo "<br>";

var\_dump($height);

echo "<br>";

var\_dump($isStudent);

echo "<br>";

var\_dump($skills);

echo "<br>";

var\_dump($person);

?>

**Output:**

* echo: Fast and simple, can output multiple items (but not return a value).
* print: Similar to echo but returns 1, so it can be used in expressions.
* var\_dump: Best for debugging, gives detailed info about type and value.

**2.PHP Variables**

**Extra LAB EXERCISES:**

**• Type Casting: Write a script that declares variables of different types and converts them into other types(e.g., integer to float, string to integer).Display the type and value before and after the conversion.**

Ans.

<?php

// Original variables

$intVar = 42;

$strVar = "123.45";

$floatVar = 9.99;

$boolVar = true;

// Display before conversion

echo "<h3>Before Conversion:</h3>";

echo "intVar: ";

var\_dump($intVar);

echo "strVar: ";

var\_dump($strVar);

echo "floatVar: ";

var\_dump($floatVar);

echo "boolVar: ";

var\_dump($boolVar);

// Conversions

$intToFloat = (float)$intVar; // Integer to Float

$strToInt = (int)$strVar; // String to Integer

$floatToString = (string)$floatVar;// Float to String

$boolToInt = (int)$boolVar; // Boolean to Integer

// Display after conversion

echo "<h3>After Conversion:</h3>";

echo "intVar to float: ";

var\_dump($intToFloat);

echo "strVar to int: ";

var\_dump($strToInt);

echo "floatVar to string: ";

var\_dump($floatToString);

echo "boolVar to int: ";

var\_dump($boolToInt);

?>

**Variable Variables: Demonstrate the use of variable variables in PHP. Write a script where a variable name is stored in another variable, and then use it to print the value.**

Ans.

<?php

// Define the value of the variable

$var\_name = "my\_variable";

$my\_variable = "Hello, world!";

// Use a variable variable to access the value

${$var\_name} = $my\_variable;

// Alternatively, you can use curly braces

$another\_var = "my\_variable";

echo ${$another\_var};

// Print the value of the variable

echo "<br>";

echo ${$var\_name};

?>

variable variables allow you to use the value of one variable as the name of another variable. This can be useful in situations where you need to dynamically create or access variable names based on other variables.

**Global and Local Scope: Write a script that shows how global and local variables work. Use the global keyword inside a function to access a global variable.**

<?php

$value = "This is a global Function";

Function valueData()

{

global $value;

echo $value;

}

valueData();

?>

**3.Super Global Variables**

**Extra LAB EXERCISES:**

**$\_GET and $\_POST: Create two separate forms: one that uses the $\_GET method and one that uses $\_POST. Display the difference in the URL and how data is passed.**

$\_GET Method – Create Form.php

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>User Form</title>

</head>

<body>

    <h2> Users Forms </h2>

    <form action="get.php" method="get">

<lable for="name"> User Name: </lable>

<input type ="Text" name="name" value="Gargi Kahar"/>

<lable for="address"> Address: </lable>

<input type ="Text" name="address" value="Manjalpur"/>

<input type="submit" value="Submit"/>

    </form>

</body>

</html>

Get.php

<?php

if (isset($\_GET['name'])) {

    echo "<p><strong>GET:</strong> " . htmlspecialchars($\_GET['name']) . "</p>";

}

if (isset($\_POST['address'])) {

    echo "<p><strong>POST:</strong> " . htmlspecialchars($\_POST['address']) . "</p>";

}

echo "<p><strong>URL:</strong> " . $\_SERVER['REQUEST\_URI'] . "</p>";

?>

Output: <http://localhost:8000/get.php?name=Gargi+Kahar&address=Manjalpur>

**$\_POST METHOD:**

Form.php

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>User Login</title>

</head>

<body>

    <h2> Users Forms </h2>

    <form action="post.php" method="POST">

<lable for="name"> User Name: </lable>

<input type ="Text" name="name"/>

<lable for="address"> Address: </lable>

<input type ="Text" name="address" />

<input type="submit" value="Submit"/>

    </form>

</body>

</html>

Post.php

<?php

if (isset($\_POST['name'])) {

    echo "<p><strong>Name:</strong> " . htmlspecialchars($\_POST['name']) . "</p>";

}

if (isset($\_POST['address'])) {

    echo "<p><strong>Address:</strong> " . htmlspecialchars($\_POST['address']) . "</p>";

}

echo "<p><strong>URL:</strong> " . $\_SERVER['REQUEST\_URI'] . "</p>";

?>

Output:

**Name:** Gargi

**Address:** Manjalpur

**URL:** /post.php

**5. Conditions, Events, and Flows**

**Extra LAB EXERCISES:**

**Nested Conditions: Write a script that uses nested if-else conditions to categorize a number as positive, negative, or zero, and also check if it's an even or odd number.**

Ans.

<?php

$num = readline("Enter the number:");

echo "The number is: $num\n";

if ($num > 0) {

    echo "It is a positive number.\n";

    if ($num % 2 == 0) {

        echo "It is even.";

    } else {

        echo "It is odd.";

    }

} elseif ($num < 0) {

    echo "It is a negative number.\n";

    if ($num % 2 == 0) {

        echo "It is even.";

    } else {

        echo "It is odd.";

    }

} else {

    echo "The number is zero.\n";

    echo "Zero is considered even.";

}

?>

**Switch Case with Multiple Cases: Write a script that accepts a grade (A,B,C,D,F) and displays a message using a switch statement. Handle multiple cases that fall under the same logic(e.g., A and B show "Excellent")**

Ans.

<?php

$grade = readline("Enter the Grade:");

switch($grade){

    case "A":

        echo "Excellent!!";

    break;

    case "B":

        echo "Excellent!!";

    break;

    case "C":

        echo "Very Good!";

    break;

    case "D":

        echo "Good!";

    break;

    case "E":

        echo "Poor!";

    break;

    default:

    echo "You Get $grade ";

}

?>

**6. If Condition and If-Else If**

**Extra LAB EXERCISES:**

**Grading System: Write a PHP program that accepts a student’s marks and outputs their grade using if-else conditions(A, B, C, D, Fail based on score).**

Ans.

<?php

$marks = readline("Enter student Marks: ");

if($marks >= 90 && $marks <=100){

    echo "Grade: A";

}elseif($marks>=80){

    echo "Grade: B";

}

elseif($marks>=70){

    echo "Grade: C";

}

elseif($marks>=50){

    echo "Grade: D";

}

elseif($marks>=0){

    echo "Grade: Fail";

}

else{

    echo "Input valid Value";

}

?>

**Temperature Converter: Write a script that takes temperature in Celsius or Fahrenheit as input and converts it to the other format using if conditions**

Ans.

<?php

$input = readline("Enter the temperature (e.g., 36C or 97F): ");

// Get the last character (C or F)

$unit = strtoupper(substr($input, -1));

// Get the numeric part

$value = floatval(substr($input, 0, -1));

if ($unit === 'C') {

    // Convert Celsius to Fahrenheit

    $fahrenheit = ($value \* 9/5) + 32;

    echo "{$value}°C is " . round($fahrenheit, 2) . "°F\n";

} elseif ($unit === 'F') {

    // Convert Fahrenheit to Celsius

    $celsius = ($value - 32) \* 5/9;

    echo "{$value}°F is " . round($celsius, 2) . "°C\n";

} else {

    echo "Invalid input. Please end the temperature with 'C' or 'F'.\n";

}

?>

**7. Practical Example:**

**Calculator and Day Finder**

**Extra LAB EXERCISES:**

**Enhanced Calculator: Modify the calculator to handle more complex operations such as exponentiation(^), modulus(%), and square root(√).**

<?php

$num1 = (float)readline("Enter the First Number: ");

$operator = readline("Enter the Operator (+, -, \*, /, %, ^, sqrt): ");

if ($operator == "sqrt") {

    // Only one number needed for square root

    if ($num1 >= 0) {

        $result = sqrt($num1);

        echo "Result: √$num1 = $result\n";

    } else {

        echo "Error: Cannot take square root of a negative number.\n";

    }

} else {

    $num2 = (float)readline("Enter the Second Number: ");

    if ($operator == "+") {

        $result = $num1 + $num2;

        echo "Result: $num1 + $num2 = $result\n";

    } elseif ($operator == "-") {

        $result = $num1 - $num2;

        echo "Result: $num1 - $num2 = $result\n";

    } elseif ($operator == "\*") {

        $result = $num1 \* $num2;

        echo "Result: $num1 \* $num2 = $result\n";

    } elseif ($operator == "/") {

        if ($num2 != 0) {

            $result = $num1 / $num2;

            echo "Result: $num1 / $num2 = $result\n";

        } else {

            echo "Error: Division by zero is not allowed.\n";

        }

    } elseif ($operator == "%") {

        if ($num2 != 0) {

            $result = $num1 % $num2;

            echo "Result: $num1 % $num2 = $result\n";

        } else {

            echo "Error: Modulo by zero is not allowed.\n";

        }

    } elseif ($operator == "^") {

        $result = pow($num1, $num2);

        echo "Result: $num1 ^ $num2 = $result\n";

    } else {

        echo "Invalid operator.\n";

    }

}

?>

**Date Finder with Time Zone: Write a script that finds the current day and prints "Happy Sunday" if it's Sunday, but also adjusts for different time zones.**

Ans.

<?php

// Ask user for a time zone (you could also hardcode it)

$timezoneInput = readline("Enter a time zone (e.g., 'America/New\_York', 'Asia/Kolkata', 'Europe/London'): ");

// Create DateTime object with the user-specified time zone

try {

$timezone = new DateTimeZone($timezoneInput);

$date = new DateTime("now", $timezone);

$day = $date->format("l"); // Full name of the day

$time = $date->format("h:i A"); // Current time in that zone

echo "In $timezoneInput, today is: $day and the time is: $time\n";

if ($day == "Sunday") {

echo “ Happy Sunday!\n";

}

} catch (Exception $e) {

echo "Invalid time zone. Please enter a valid one like 'Asia/Tokyo' or 'Europe/London'.\n";

}

?>

**Month Display: Create a program using switch case that takes a number (1-12) and displays the corresponding month.**

Ans.

<?php

$monthNumber = (int)readline("Enter a month number (1-12): ");

switch ($monthNumber) {

case 1:

echo "January";

break;

case 2:

echo "February";

break;

case 3:

echo "March";

break;

case 4:

echo "April";

break;

case 5:

echo "May";

break;

case 6:

echo "June";

break;

case 7:

echo "July";

break;

case 8:

echo "August";

break;

case 9:

echo "September";

break;

case 10:

echo "October";

break;

case 11:

echo "November";

break;

case 12:

echo "December";

break;

default:

echo "Invalid month number. Please enter a number between 1 and 12.";

}

?>

**Discount Calculation (Ternary Operator): Write a script that calculates and displays the discount on a product based on a user-defined price. If the price Is above 500, give a 10% discount; otherwise, no discount(use the ternary operator).**

<?php

// Get the product price from the user

$price = (float)readline("Enter the product price: ");

// Calculate discount using ternary operator

$discount = ($price > 500) ? $price \* 0.10 : 0;

// Calculate final price after discount

$finalPrice = $price - $discount;

// Display the result

echo "Original Price: ₹$price\n";

echo "Discount: ₹$discount\n";

echo "Final Price: ₹$finalPrice\n";

?>

**9. Loops: Do-While, For Each, For Loop**

**Extra LAB EXERCISES:**

**Fizz Buzz Program: Write a program using a for loop that prints numbers from 1to 100.But for multiples of 3,print"Fizz"instead of the number, for multiples of 5print "Buzz", and for multiples of both 3 and 5 print "FizzBuzz".**

<?php

$i = 1;

do {

if ($i % 3 == 0 && $i % 5 == 0) {

echo "FizzBuzz\n";

} elseif ($i % 3 == 0) {

echo "Fizz\n";

} elseif ($i % 5 == 0) {

echo "Buzz\n";

} else {

echo "$i\n";

}

$i++;

} while ($i <= 100);

?>

**Multiplication Table: Write a PHP script using a nested for loop to generate a multiplication table from 1 to 10.**

<?php

echo "Multiplication Table from 1 to 10:\n\n";

for ($i = 1; $i <= 10; $i++) {

    for ($j = 1; $j <= 10; $j++) {

        $result = $i \* $j;

        echo "$i x $j = $result\n";

    }

    echo "-----------------------\n"; // Separator for each table

}

?>

**Reverse Number Sequence: Write a script using a do-while loop that displays numbers from 10 to 1**

Ans.

<?php

$num = 10;

do {

echo "$num\n";

$num--;

} while ($num >= 1);

?>

**10. PHP Array and Array Functions**

**Extra LAB EXERCISES:**

**Sorting Arrays: Write a script that demonstrates the use of sort(), rsort(), asort(), and ksort() functions to sort arrays.**

Ans.

<?php

// Indexed Array

$numbers = [5, 3, 9, 1, 7];

echo "Original Indexed Array:\n";

print\_r($numbers);

// sort()

sort($numbers);

echo "\nSorted with sort():\n";

print\_r($numbers);

// rsort()

rsort($numbers);

echo "\nSorted with rsort():\n";

print\_r($numbers);

// Associative Array

$students = [

"Gargi" => 85,

"Ayansh" => 92,

"Ram" => 78,

"Raj" => 90

];

echo "\nOriginal Associative Array:\n";

print\_r($students);

// asort()

asort($students);

echo "\nSorted with asort() (by value):\n";

print\_r($students);

// ksort()

ksort($students);

echo "\nSorted with ksort() (by key):\n";

print\_r($students);

?>

**Multi-dimensional Array: Create a multi-dimensional array to store information about products (name, price, and stock). Write a script to display the information in a tabular format.**

**Ans.**

<?php

// Multi-dimensional array for products

$products = [

    ["name" => "Laptop", "price" => 75000, "stock" => 10],

    ["name" => "Smartphone", "price" => 30000, "stock" => 25],

    ["name" => "Tablet", "price" => 20000, "stock" => 15],

    ["name" => "Monitor", "price" => 12000, "stock" => 8],

    ["name" => "Keyboard", "price" => 1500, "stock" => 50],

];

// Display header

echo "Product Information:\n";

echo str\_repeat("-", 45) . "\n";

printf("%-15s %-10s %-10s\n", "Name", "Price (₹)", "Stock");

echo str\_repeat("-", 45) . "\n";

// Display each product row

foreach ($products as $product) {

    printf("%-15s %-10d %-10d\n", $product["name"], $product["price"], $product["stock"]);

}

?>

Array Merge and Diff: Write a PHP script that merges two arrays and finds the difference between them using array\_merge() and array\_diff()

Ans.

<?php

// Two sample arrays

$array1 = ["apple", "banana", "cherry"];

$array2 = ["banana", "cherry", "date", "fig"];

// Merging the arrays

$mergedArray = array\_merge($array1, $array2);

echo "Merged Array:\n";

print\_r($mergedArray);

// Finding difference: elements in array1 that are not in array2

$diff1 = array\_diff($array1, $array2);

echo "\nItems in array1 but not in array2:\n";

print\_r($diff1);

// Finding difference: elements in array2 that are not in array1

$diff2 = array\_diff($array2, $array1);

echo "\nItems in array2 but