**Module1–Core PHP**

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

Structure of a PHP Script

A PHP script follows a specific structure that allows it to be executed on the server before the HTML is sent to the browser. Here's the basic structure:

Basic PHP Syntax

A PHP script can be placed anywhere in the document.

A PHP script starts with <?php and ends with ?>:

**<?php**

**// Write your code here**

**?>**

The default file extension for PHP files is ".php".

A PHP file normally contains HTML tags, and some PHP scripting code.

Embedding PHP in HTML

<!DOCTYPE html>

<html>

<head>

<title>My PHP Page</title>

</head>

<body>

<h1>Welcome to my website</h1>

<?php

$username = "Nisha";

echo "<p>Hello, $username!</p>";

?>

</body>

</html>

PHP is embedded within <?php ... ?> inside the HTML.

When the server runs this file, it processes the PHP and sends the result (pure HTML) to the browser.

2) What are the rules for naming variables in PHP?

Rules for 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 \_ )
* Variable names are **case-sensitive** ($age and $AGE are two different variables)

LAB EXERCISE

Write a PHP script to print"Hello,World!" on a webpage

<?php

Echo “Hello,World!”;

?>

PHP Variables

3) Explain the concept of variables in PHP and their scope.

A **variable** in PHP is a container used to store data such as numbers, text, or arrays. You can reuse the value throughout your script.

$variable\_name = value;

* Must begin with $
* Assign values using the = operator

Scope of Variables in PHP

The **scope** of a variable refers to where it can be accessed in your code. PHP has **3 main types** of variable scope:

* local
* global
* static

***Local Scope***

 A variable declared **inside a function** is local to that function.

 You **cannot access** it outside the function.

**function greet() {**

**$name = "Nisha"; // Local variable**

**echo "Hello $name";**

**}**

**greet(); //**

**echo $name; //**

***Global Scope***

* Variables declared **outside any function** are global.
* Cannot be used **directly inside** a function unless you **declare them global** inside that function.

**$city = "Mumbai"; // Global**

**function showCity() {**

**global $city; // Bring global variable into local scope**

**echo $city;**

**}**

**showCity(); // Output: Mumbai**

***Static Scope***

A **static variable** inside a function **retains its value** between function calls.

**function counter() {**

**static $count = 0;**

**$count++;**

**echo $count . "<br>";**

**}**

**counter(); // 1**

**counter(); // 2**

**counter(); // 3**

LAB EXERCISE

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

 <!-- Write a PHP script to print"Hello,World!" on a webpage -->

<?php

**echo** "<h1> Hello,World! </h1>";

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

 $id=1;

 $empname="Nisha Sharma";

 $salary=20000.08;

 $active=true;

**echo** "<h2>Employee details:</h2></br>";

**echo** "Employee id :".$id. "</br> Employee name :".$empname. "</br> Salary :".$salary. "</br> Status : ".$active;

?>

Super Global Variables

4)what are super global variables in PHP? List at least five super global arrays and their use.

In PHP, super global variables are predefined, built-in variables that are always accessible from any scope within a script, including functions and classes

1. $\_GET:>

Uses => Retrieves data sent through the **URL** (query string).

Common in => Search forms, filters, pagination.

1. $\_POST

Uses => Collects data from a **form submitted with POST** method.

Common in => Login forms, registration forms, secure form submissions.

1. $\_SESSION

Uses => Stores data across different pages for the **same user session**.

Common in => User login system, shopping carts.

1. $\_COOKIE

Uses => Retrieves values stored in **user's browser cookies**.

Common in => Remember-me feature, language preference, tracking.

1. $\_FILES

Uses => Handles file uploads from a form.

Common in => Uploading profile pictures, documents, etc.

Conditions, Events and Flows

5)Explain how conditional statements work in php.

In PHP, conditional statements control the flow of a program by executing different blocks of code based on whether a given condition is true or false. The most common conditional statements are if, else, elseif, and switch.

1. *If* statement =>if statement executes the block of code only if a specified condition is true.

Syntax: if (condition) {

// Code to be executed if the condition is true

}

1. *If…else* statement =>Executes one block of code **if the condition is true**, and another block **if it's false**.

Syntax: if(condition){

// Code to be executed if the condition is true

}

Else{

// Code to be executed if the condition is false

}

1. *If…elseif…else* statement => Used to test **multiple conditions**.

*Syntax:*

if(condition1){

//code to be executed if condition1 is true

}

Elseif(condition2){

//code to be executed if condition1 is true

}

Else{

*// Code to be executed if all previous conditions are false*

}

1. *Switch* statement =>An alternative to many if...elseif blocks. It checks **one expression** against **many values**.

Syntax: switch (expression){

case value1:

// Code to be executed if expression == value1

break;

case value2:

// Code to be executed if expression == value2

break;

default:

// Code to be executed if expression doesn't match any case

}

Loops: Do-While, ForEach, ForLoop

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

In PHP, for loop, foreach loop, and do-while loop offer different ways to iterate through code blocks.

1. For loop => for loop is used when the number of iterations is known in advance.

for (initialization; condition; increment) {

// Code to be executed }

1. Foreach loop =>The foreach loop is designed specifically for iterating over arrays and objects.

Syntax foreach($array as $value){

Echo “$value”;

}

Associative array

Exg $colors=[“red”,”green”,”blue”];

foreach ($colors as $index=>$color){

echo “Index $index: $color<br>”;

}

1. Do-while loop => The do-while loop is similar to the while loop, but with one crucial difference: the code block is executed at least once before the condition is checked.

Syntax do {

// Code to be executed

} while (condition);

PHP Array and Array function

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

An array in PHP is a data structure that store multiple values in a single variable. Arrays are useful for grouping related data together and accessing them using indexes or keys.

Example: $material=array(“bag”,”pencil”,”pen”,”books”);

Type of array

There are mainly three types of array:

1)Indexed array :- An array with a numeric index. The index starts from 0 by default.

$material=array(“bag”,”pencil”,”pen”,”books”);

//accessing values through indexes

Echo $array[2]; // output is pen

2)Associative array :- An array where keys are named(strings) instead of an number. Used to store key-value pairs.

$student=array(

“rollno” => 1,

“name” =>”nisha”,

“course” =>”PHP”

) ;

//accessing values echo $array[name];

3)Multidimensional array :- An array that contains one or more array inside it. Used to represent tables or metrices.

$marks = array(

"John" => array("Math" => 85, "Science" => 90),

"Sara" => array("Math" => 78, "Science" => 88)

);

echo $marks["John"]["Science"]; // Output: 90 //accessing values

PHP Date-Time Function

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

<?php

**date\_default\_timezone\_set**("Asia/Kolkata"); *// Set timezone as needed*

**echo** "<h2>Current date and time in different formats</h2>";

*//format 1  full date and time*

**echo** "1.Full date and time =>". **date**("d-m-Y H:i:s")."<br>";

*//format 2 year and month*

**echo** "2. Date/Month/year =>".**date**("d-m-y")."<br>";

*//format 3 day of week,month and year*

**echo** "3.Day of week,Month,year(l,F Y) =>".**date**("l,F Y")."<br>";

*//format 4  12-hour format with AM/PM*

**echo** "4.12-hour format with AM/PM =>".**date**("h:i A")."<br>";

*//for future date prediction*

*// time() print unix time-stamp from 1, jan 1970*

**echo** **time**()."<br>";

$onehour=**time**()+(1\*60\*60);

**echo** "Future date from current time =>".**date**('h:i a',$onehour)."<br>";

$oneday=**time**()+(1\*24\*60\*60);

**echo** "Future day from current time =>".**date**("d/m/Y",$oneday)."<br>";

*// to get the future date and time add*

$time=**mktime**(**date**("h")+1,**date**("i")+5,**date**("s")+1);

**echo** "Date and time after 2 hour =>".$d=**date**("h:i:s",$time)."<br>";

*//it easier way from last 2 function using string*

**echo** **date**("d-m-Y",**strtotime**("+ 5 days"))."<br>";

**echo** **date**('h:i:s A',**strtotime**("+ 5 date"))."<br>";

?>

Header Function

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

The header() function in php sends a raw HTTP header to the browser. It is mainly used to:

=>Redirect pages

=>set content pages

=>Control caching

=> Force file to download

-> Handle authentication

1. **Redirect to another page**

header("Location: homepage.php");

exit(); // Always use exit after redirection

**2. Set content type (e.g., for JSON or PDF)**

header("Content-Type: application/json");

echo json\_encode(["name" => "Nisha", "status" => "active"]);

3. **Force file download**

header("Content-Disposition: attachment; filename=\"example.txt\"");

header("Content-Type: text/plain");

readfile("example.txt");

4. **Prevent browser caching**

header("Cache-Control: no-cache, must-revalidate");

header("Expires: Sat, 1 Jan 2000 00:00:00 GMT");

5. **Set custom response code**

header("HTTP/1.1 404 Not Found");

Include and required

Explain the difference between include and require in PHP.

**Include**

=> **Behavior on Error**: If the file cannot be found or included, PHP **throws a warning** (E\_WARNING) but the script **continues to run**.

=> **Use case**: Use include when the file is **not essential** for the application to run (e.g., optional sidebar or advertisement).

Exg

include("optional\_file.php"); // If file not found, script continues

echo "Script continues even if file is missing.";

**require**

=> **Behavior on Error**: If the file cannot be found or included, PHP **throws a fatal error** (E\_ERROR) and **stops the script execution**.

=> **Use case**: Use require when the file is **critical** for the application (e.g., configuration, database connection).

Exg

require("config.php"); // If file not found, script stops

echo "This will not be executed if config.php is missing.";

PHP Expressions, Operations, and String Functions

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

In PHP, an expression is any unit of code that evaluates to a value. This includes constants, variables, function calls, and combinations of these elements using operators. Essentially, if something in PHP can produce a result, it is an expression.

**🔹 Types of PHP Expressions:**

1. **Arithmetic Expressions** – Perform mathematical calculations.
2. **Logical Expressions** – Return true or false based on conditions.
3. **String Expressions** – Combine strings.
4. **Assignment Expressions** – Assign values using =.
5. **Comparison Expressions** – Compare values.

Arithmetic expression

<?php

$a = 10;

$b = 4;

// Addition

$sum = $a + $b; // 10 + 4 = 14

echo "Sum: $sum<br>";

// Subtraction

$diff = $a - $b; // 10 - 4 = 6

echo "Difference: $diff<br>";

// Multiplication

$prod = $a \* $b; // 10 \* 4 = 40

echo "Product: $prod<br>";

// Division

$div = $a / $b; // 10 / 4 = 2.5

echo "Division: $div<br>";

// Modulus (remainder)

$mod = $a % $b; // 10 % 4 = 2

echo "Modulus: $mod<br>";

?>

✅ Logical Expression Examples:

<?php

$x = 5;

$y = 10;

// Logical AND

if ($x < 10 && $y > 5) {

echo "Logical AND: true<br>"; // both conditions are true

} else {

echo "Logical AND: false<br>";

}

// Logical OR

if ($x < 3 || $y > 5) {

echo "Logical OR: true<br>"; // second condition is true

} else {

echo "Logical OR: false<br>";

}

// Logical NOT

$z = true;

if (!$z) {

echo "Logical NOT: true<br>";

} else {

echo "Logical NOT: false<br>"; // because $z is true, !z is false

}

?>