MODULE 1-CORE PHP

1.1) Discuss the structure of a PHPs Script and how to embed PHP in HTML ?.

-> 1.structure of php script

```
<?php
// PHP code goes here
statement1;
statement2;
...
?>
```

-> <?php ... ?> :Opening and closing tags that tell the server “this is PHP code.”

->output :Use echo or print to send text to the browser.

->2. Embedding PHP in HTML ?

->example

```
<!DOCTYPE html>
<html>
<head>
  <title>My PHP Page</title>
</head>
<body>
  <h1>Welcome to My Website</h1>

  <p>
    <?php
    // Embedding PHP to display dynamic content
    $name = "faizan";
    echo "Hello, $name! Today is " . date("l");
```

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

1.2) What are the rules for naming variables in PHP ?.

-1. Start with a $ sign

example: $age = 20;

$name = "dk";

-2. The first character after $ must be a letter or underscore

example: $_value = 10;

$name = "faizan";

-3. After the first character, you can use letters, numbers, or underscores

example:$user1 = "Alex";

$_temp_value = 50;

-4. Variable names are case-sensitive

example:$Name = "nasir";

$name = "Danish";

echo $Name; // Outputs: nasir

echo $name; // Outputs: Danish

-5. No spaces or special characters allowed

example:$first_name = "Ali";

$first-name = "Ali"; // Error

$first name = "Ali"; // Error

3.) PHP Variables

THEORY EXERCISE:

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

-- Variables in PHP

-A variable in PHP is a named container that stores data, such as numbers, text, or arrays, for use in a script

->Key points about PHP variables:

->They start with a $ sign.

-The value they hold can change (hence the name “variable”).

T-hey can store different data types like strings, numbers, arrays, etc.

example:

```
<?php
$city = "wankaner";
$population = 15000;
echo "City: $city, Population: $population";
?>
```

=>Scope of Variables in PHP

--local :Declared inside a function; accessible only within that function.

--global :Declared outside any function; accessible in the whole script except inside functions unless imported with global.

--static :Declared inside a function with static; retains its value between function calls.

4.0) Super Global Variables

-THEORY EXERCISE:

4.1) What are super global variables in PHP ? List at least five super global arrays and their ?

--what are super global variable ?

->Superglobals are built-in variables in PHP that are always accessible from anywhere in the script inside functions, classes, or directly in the global scope without using global.

-- hey are associative arrays that store different kinds of information such as:

Form input data

Server environment details

Session & cookie data

URL query parameters

--Five Common Superglobal Arrays & Their Uses

->$_GET :

--Stores data sent through the URL query string (HTTP GET method).

example:$_GET['id'] → Get value from page.php?id=10

->$_POST :

--Stores data sent via HTTP POST method (e.g., form submissions).

example:$_POST['username'] → Get form input value

->$_REQUEST :

--Stores data from both GET and POST (and COOKIE if enabled).

example:$_REQUEST['search'] → Works for GET or POST

->$_SERVER:

--Contains server and execution environment information.

example:$_SERVER['HTTP_USER_AGENT'] → Get browser info

->$_SESSION:

--Stores session variables for a user (persistent across pages).

example:$_SESSION['user'] = 'faizan';

6.0) Conditions , Events ,and Flows ?

=> THEORY EXERCISE:

6.1)Explain how conditional statements work in PHP ?

--What Are Conditional Statements?

->Conditional statements let you control the flow of a PHP program they decide which block of code should run based on whether a condition is true or false.

--Types of Conditional Statements in PHP

1. if Statement

2. if...else Statement

3. if...elseif...else Statement

4. switch Statement

1.if statement

->Executes code only if the condition is true.

--syntax

```
->if (condition) {
   // code to execute if condition is true
}
```

2.if ...else statement

->Executes one block if the condition is true, otherwise executes another block.

--example

```
->i$age = 16;
if ($age >= 18) {
   echo "You can vote.";
} else {
```

```
    echo "You are too young to vote.";
}
```

3. if...elseif...else Statement

->Checks multiple conditions in sequence.

--example

```
-->$marks = 75;
if ($marks >= 90) {
    echo "Grade: A";
} elseif ($marks >= 75) {
    echo "Grade: B";
} else {
    echo "Grade: C";
}
```

4. switch Statement

->Used to compare the same variable/value against multiple possible matches.

--example

```
->$day = "Monday";

switch ($day) {
    case "Monday":
        echo "Start of the week.";
        break;
    case "Friday":
        echo "Weekend is near.";
        break;
    default:
        echo "Just another day.";
```

}

How It Works Internally

1.Condition Evaluation → The expression inside if() or switch is evaluated.

2.Boolean Result → PHP converts it to true or false.

3.Branch Execution → Only the matching code block runs; others are skipped.

4.Optional Else/Default → Executes if no condition matches.

10.Loops :Do-While, ForEach ,ForLoop

-- THEORY EXERCISE:

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

->Loops in PHP are used to repeat a block of code multiple times, but each loop type works differently and is suited for specific use cases.

1. for Loop

->urpose: Used when the number of iterations is known in advance.

Structure: Combines initialization, condition check, and increment/decrement in a single statement.

Condition Check: Happens before each iteration — if the condition is false, the loop will not execute.

->Example:

```
        for ($i = 1; $i <= 5; $i++) {
   echo "Number: $i <br>";
}
```

2. foreach Loop

->Purpose: Specifically designed for looping through arrays and objects.

Structure: No need to handle indexes manually — automatically assigns each element to a variable.

Condition Check: Iterates through all elements until the array ends.

-->example

```
        $colors = ["sky blue", "Green", "white"];
foreach ($colors as $color) {
   echo "$color <br>";
}
```

3. do-while Loop

->Purpose: Runs the code at least once, even if the condition is false initially.

Structure: Executes the loop body first, then checks the condition after each iteration.

Condition Check: Happens after execution.

->example

```
        $count = 1;
do {
   echo "Count: $count <br>";
   $count++;
} while ($count <= 3);
```

11. PHP Array and Array Functions

=>THEORY EXERCISE:

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

--defination of array ?

--> --An array in PHP is a special variable that can store multiple values in a single name, instead of creating separate variables for each value.

Each value in an array is stored with a key (index), which can be a number or a string.

-->type of array ?

->main three type in array

1)Indexed array

->Keys are numeric (starting from 0 by default).

Values are accessed using their index number.

example:

```
$fruits = ["Apple", "Banana", "Mango"];
echo $fruits[1]; // Outputs: Banana
```

2)Associative Arrays

->Keys are strings instead of numbers.

-Useful for storing data with named keys.

->example:

```
$student = [
   "name" => "faizan",
   "age" => 22,
   "city" => "Rajkot"
];
echo $student["name"]; // Outputs: faizan
```

3. Multidimensional Arrays

->An array containing one or more arrays inside it.

Used for representing tables or complex data structures.

->example

```
        $marks = [
```

```
    ["Math", 85],
    ["Science", 90],
    ["English", 88]
];
echo $marks[1][0]; // Outputs: Science
```

13.HeaderFunction

->THEORY EXERCISE:

13.1) ? What is the header function PHP and how is it used?

-> Definition

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

It allows you to control things like:

--Page redirection

--Content type

--Cache control

--File downloads

->syntax:

```
header(string $header, bool $replace = true, int $http_response_code = 0);
```

example: page redirect

```
<?php
// Redirect the user to another page
```

```
header("Location: https://www.google.com");


// Always call exit() after a redirect to stop further execution
exit;
?>
```

14.0)Include and Require

=>THEORY EXERCISE

14.1)Explain the difference between include and require in PHP ?

-> Difference Between include and require in PHP

->Both include and require are used to insert the content of one PHP file into another before the server executes it.They help in code reusability and modular programming.

-->1. include

--Includes and evaluates the specified file.

--If the file is missing or has an error, PHP will show a warning but continue executing the script.

example:

```
<?php
include "header.php"; // If missing → warning, but script continues
echo "Main content here.";
?>
```

->2. require

--Also includes and evaluates the specified file.

--If the file is missing or has an error, PHP will show a fatal error and stop script execution.

example:

```
<?php
require "header.php"; // If missing → fatal error, script stops
echo "This will not run if file is missing.";
?>
```

=>When to Use

include → For optional files (like extra widgets, analytics code).

require → For essential files (like configuration, database connection).

16.PHP Expressions,Operations,and String Functions

=>THEORY EXERCISE

16.1) Explain what PHP expressions are and give examples of arithmetic and logical operations ?

->PHP Expressions

--An expression in PHP is anything that produces a value.

--It can be a variable, a constant, or a combination of values and operators.

example:

```
$x = 10;          // simple assignment expression
$y = $x + 5;      // arithmetic expression
$isTrue = ($x > 5); // logical expression
```

=>Arithmetic Operations in PHP

--Addition (+)

--Subtraction (-)

--Multiplication (*)

--Division (/)

--Modulus (%)

example:

```
<?php
$a = 8;
$b = 3;
```

```
echo $a + $b; // 11
echo $a - $b; // 5
echo $a * $b; // 24
echo $a / $b; // 2.666...
echo $a % $b; // 2
echo $a ** $b; // 512
?>
```

=>Logical Operations in PHP

These are used to combine conditions and produce a true or false result.

->AND (&& or and)

--OR (|| or or)

--NOT (!)

--XOR (xor)

example:

```
<?php
$x = 7;
$y = 4;

var_dump($x > 5 && $y < 10); // true
var_dump($x > 5 || $y > 10); // true
var_dump(!($x == 7));       // false
var_dump($x > 5 xor $y < 3); // true (only one is true)
?>
```