Module 5 – Core PHP

1. Introduction to C++

[THEORY EXERCISE]

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

- A PHP script typically starts with the `<?php` opening tag and ends with the `?>` closing tag.
- PHP code can be embedded within HTML pages to create dynamic web content.
- Example:

```
<?php
// PHP code here
?>
```

- You can insert PHP code anywhere in the HTML file, making it versatile for dynamic content generation.
- Comments in PHP can be added using:
- Single-line comment: `//` or `#`
- Multi-line comment: `/* ... */`

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

- Variables in PHP start with a dollar sign `$` followed by the name.
- Variable names are case-sensitive (`$var` and `$Var` are different).
- Names must begin with a letter (a-z or A-Z) or an underscore `_`.

php naming rules

1) starts with latter or _

$$_abc; ------------- true

$abc123=10; ---------- true

$123abc=10; ---------- false

$abc a123=10 --------- false //NO space

//both work same as print & declaration

[LAB EXERCISE]

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

```
<?php

echo "Hello, World"

?>
```

3. PHP Variables

[THEORY EXERCISE]

1. Explain the concept of variables in PHP and their scope.

- Variables are containers used to store data values.
- In PHP, variables are defined by a dollar sign `$` followed by the variable name.
- PHP supports different data types for variables, such as strings, integers, floats, booleans, arrays, objects, NULL, and resources.
- Example:
- $txt = "Hello World!";
- $x = 5;
- $y = 10.5;

Variable Scope in PHP

Variables in PHP can have different scopes, which determine where the variables can be accessed

1. **Local Scope**: Variables declared inside a function are local to that function and cannot be accessed outside it.
2. **Global Scope**: Variables declared outside functions are global and can be accessed outside any function. To access these inside functions, you need to use the `$GLOBALS` array or the `global` keyword.
3. **Static Scope**: Variables declared with the `static` keyword inside functions retain their value between multiple calls to the function. They are initialized only once and exist throughout the script execution.

Example Demonstrating Scope:

- $x = 10; // global scope
- function test() {
- $y = 5; // local scope
- static $z = 0; // static variable
- $z++;
- echo "Local y: $y<br>";
- echo "Static z: $z<br>";

}

- test(); // outputs y=5, z=1

test(); // outputs y=5, z=2

4. Super Global Variables THEORY EXERCISE: What are super global variables in PHP? List at least five super global arrays and their use

4. Super Global Variables

[THEORY EXERCISE]

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

Super globals are accessible everywhere in PHP, providing essential arrays to interact with server information, user input, and application state.

1. **`$_GLOBALS`**
- Use: Stores all global variables available in the script. It allows access to variables declared outside of functions or classes from anywhere within the script.
2. **`$_SERVER`**
- Use: Contains information such as headers, paths, script locations, server details, and execution environment. For example, `$_SERVER['HTTP_USER_AGENT']` provides the browser details, and `$_SERVER['REQUEST_METHOD']` indicates GET or POST requests.
3. **`$_REQUEST`**
- Use: Fetches request data sent via GET, POST, and COOKIE methods. Example: to retrieve a value sent by a form submission.
4. **`$_POST`**
- Use: Contains data submitted through an HTTP POST request. Commonly used for form handling and data submission.
5. **`$_GET`**
- Use: Holds data sent via URL parameters in a GET request. Used to retrieve variables passed in the URL query string.

6. Conditions, Events, and Flows

[THEORY EXERCISE]

1. Explain how conditional statements work in PHP.

Super globals are accessible everywhere in PHP, providing essential arrays to interact with server information, user input, and application state.

How the conditional statements work:

- `if` **Statement:** The most basic conditional statement. It executes a block of code only if the specified condition evaluates to true.
- `if-else` **Statement:** This statement extends the `if` statement by providing an alternative block of code (the `else` block) that is executed if the `if` condition is false.
- `if-elseif-else` **Statement:** This is used when there are multiple conditions to be checked sequentially. PHP evaluates each `elseif` condition only if the preceding `if` and `elseif` conditions were false. The final `else` block executes if none of the preceding conditions are met.
- `switch` **Statement:** The `switch` statement is an alternative to a long `if-elseif-else` structure. It compares a variable against a series of values (cases) and executes the block of code corresponding to the matching case.
- `Nested if` **Statement:** It is possible to embed one `if` statement inside another. This allows for more complex conditional logic where a subsequent condition is checked only if the outer condition is true.

10. Loops:

Do-While, For Each, For Loop

[THEORY EXERCISE]

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

1. For Loop

- When the number of iterations is known ahead of time (e.g., counting from 1 to 10).

- for (initialization; condition; increment)

  ```
  for ($i = 1; $i <= 5; $i++) {

      echo "Count: $i<br>";

  }
  ```

2. Foreach Loop

- Specifically designed to **iterate over arrays**.
- foreach ($array as $value)

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

  foreach ($colors as $color) {

      echo "Color: $color<br>";

  }
  ```

3. Do-While Loop

- When you want the loop to execute at least once, even if the condition is false.
- do {
  }
  while (condition);

  ```
  $i = 1;
  do {
  echo "Value is: $i<br>";
  $i++;
  }
  while ($i <= 3);
  ```

Feature	`for` **loop**	`foreach` **loop**	`do-while` **loop**
Purpoe	Fixed number of loops	Loop through array	Execute at least once
Best for	Numeric iteration	Arrays	Pre-run action before check

Condition Check	Before loop starts	Automatically handled	After loop runs once
Usage Limitation	General-purpose	Arrays/objects only	Rarely used, niche cases

11. PHP Array and Array Functions

[THEORY EXERCISE]

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

1. Definition of Arrays in PHP

- An array in PHP is a special variable that can store multiple values in a single variable.

-
  ```
  $fruit1 = "Apple";
  $fruit2 = "Banana";

  $fruits = array("Apple", "Banana");
  ```

Types of Arrays in PHP : three main types of arrays:

1. **Indexed Array (Numeric Array)**

   Uses numeric indices (starting from 0).

   Best for ordered lists.

   ```
   $colors = array("Red", "Green", "Blue");
   echo $colors[1];
   ```

2. **Associative Array**

   Uses **named keys** instead of numbers.

   Best for **key-value pairs** (like a dictionary).

   ```
   $person = array("name" => "John", "age" => 30);

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

3. **Multidimensional Array**

   An array containing one or more arrays.

Used for tabular data (like a matrix or table).

```
$students = array(

    array("Amit", 85, "A"),
    array("Ravi", 78, "B"),
    array("Neha", 92, "A+")

);

echo $students[1][0];
```

Type	Index Format	Use Case
Indexed Array	Numeric (0,1,2…)	Ordered list (colors, fruits)
Associative Array	Named keys	Key-value data (name, age)
Multidimensional	Arrays inside array	Tables, nested data structures

13. Header Function

[THEORY EXERCISE]

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

1. What is the header() Function in PHP

- The header() function in PHP is used to send raw HTTP headers to the browser before any output is sent. It is commonly used to control redirection, content type, caching, file downloads, and more.

  Important:
  header() must be called before any actual output (HTML, echo, or whitespace) is sent to the browser.
- $fruit1 = "Apple";
  $fruit2 = "Banana";

Uses of header()

1. **Redirect to Another Page**

   ```
   header("Location: dashboard.php");
   exit();
   ```
2. **Set Content-Type**

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

3. **Force File Download**

   ```
   header("Content-Disposition: attachment; filename=\"file.pdf\"");
   header("Content-Type: application/pdf"); readfile("file.pdf");
   header("Content-Type: application/json");
   ```

4. **Prevent Caching**

   ```
   header("Cache-Control: no-cache, must-revalidate");
   header("Expires: Sat, 1 Jan 2000 00:00:00 GMT");
   ```

5. **Refresh or Delay Redirect**

   ```
   header("Refresh: 5; URL=home.php"); // Redirect after 5 seconds
   ```

6. **Example :**

   ```
   <?php

   session_start();

   if ($_SESSION['user']) {

   header("Location: dashboard.php");

   exit();

   } else {
   ```

```
echo "Unauthorized access!";

}

?>
```

14. Include and Require

[THEORY EXERCISE]

1. Explain the difference between include and require in PHP.

1. What are `include` and `require` in PHP?

- Both `include` and `require` are used to insert the content of one PHP file into another PHP file.
- They help in reusing code, such as headers, footers, configuration files, or database connections.

2. Common

- include 'filename.php';
- require 'filename.php';

3. Difference Between `include` and `require

Feature	include	require
On Error	Generates a warning	Generates a fatal error
Script Execution	Continues after error	Stops execution on error
Return Value	Returns `false` if file not found	Does not return — stops immediately
Use Case	Optional files (e.g., optional menu)	Critical files (e.g., DB connection)

Use `require` when the file is ”essential”.

Use `include` when the file is “optional”.

4. Example : include

-
  ```
  <?php

  include 'optional_file.php';
  echo "This will still execute even if the file is missing.";

  ?>
  ```

Example : require

-
  ```
  <?php

  require 'important_file.php';
  echo "This will NOT execute if the file is missing.";

  ?>
  ```

16. PHP Expressions, Operations, and String Functions

[THEORY EXERCISE]

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

1. Expressions

- A PHP expression is anything that evaluates to a value. Expressions are the building blocks of PHP code and are used in assignments, conditions, loops, and function arguments

  .

- **Examples :**

```
$sum = 5 + 10;
$isAdult = $age > 18;
$message = "Hello " . "World";
```

1. Arithmetic Operations

Operator	Description	Example	Result
+	Addition	5 + 3	8
-	Subtraction	10 - 4	6
*	Multiplication	6 * 2	12
/	Division	8 / 2	4
%	Modulus (remainder)	9 % 4	1
These are used to perform mathematical calculations.			

1. Logical (Boolean) Operations

Operator	Name	Example	Result
#ERROR!	Equal	5 == 5	TRUE
!=	Not equal	5 != 3	TRUE
>	Greater	7 > 2	TRUE
<	Lesser	3 < 10	TRUE
&&	AND	true && false	FALSE
`		`	OR
!	NOT	!true	FALSE
These are used for decision-making and comparisons.			

- **Examples :**

```
<?php

$a = 10;
$b = 5;
$c = $a + $b;
if ($c > 10 && $b < $a) {
    echo "Expression is true";
}
?>
```