# Module 1 – Core PHP

* PHP Syntax

THEORY EXERCISE:

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

* First open the file in any editor and then write HTML code according to requirement. If we have to[add PHP code](https://www.geeksforgeeks.org/different-ways-to-write-a-php-code/) then we can add by simply adding **<?php ..... ?>** tags in between and add your PHP code accordingly.

<!DOCTYPE html>

<html>

<head>

<title>PHP</title>

</head>

<body>

<h1>

<?php

echo " embed PHP in HTML "

?>

</h1>

</body>

</html>

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

* A variable can have a short name (like $x and $y) or a more descriptive name ($age, $carname, $total\_volume).
* 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**)
* PHP Variables THEORY EXERCISE:

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

* In PHP, a variable starts with the $ sign
* In PHP, variables are used to store data values that can be referenced and manipulated within a script. They act as containers that hold information, such as numbers, strings, or arrays, and their values can be changed during the execution of the script.
* PHP supports the following data types:
* String
* Integer
* Float (floating point numbers - also called double)
* Boolean
* Array
* Object
* NULL
* Resource
* PHP Variables Scope:
* In PHP, variables can be declared anywhere in the script.
* The scope of a variable is the part of the script where the variable can be referenced/used.
* PHP has three different variable scopes:
* local
* global
* static

1. **Local Scope**

* Variable declared inside a function or block.
* Accessible only within that function or block.

1. **Global Scope**

* Variable declared outside all functions.
* Accessible throughout the program.
* Super Global Variables THEORY EXERCISE:

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

* A global variable is super global & predefined variable.
* A global variable is a programming language construct, a variable type that is declared outside any function and is accessible to all functions throughout the program.
* There are some types of global variable,

1. $GLOBALS
2. $\_SERVER
3. $\_REQUEST
4. $\_POST
5. $\_GET
6. $\_FILES
7. $\_ENV
8. $\_COOKIE
9. $\_SESSION
10. **$GLOBALS** :

**Use:** we can make variable as local to global & access anywhere in program

1. **$\_GET :**

**Use:** Collects data sent via URL parameters (query string).

**Example:** example.php?name=John → $\_GET['name'] returns "John".

1. **$\_POST:**

**Use:** Collects data sent through HTTP POST method (usually from forms).

**Example:** After submitting a form with method="post", access input with $\_POST['username'].

1. **$\_REQUEST**:

**Use Case:** It provides a unified way to access form data, regardless of whether it's sent via GET or POST.

1. **$\_COOKIE :**

**Use**: Use the setcookie() function before any output is sent to the browser.

* Conditions, Events, and Flows

**1. Explain how conditional statements work in PHP.**

* Conditional statements let your PHP code make **decisions**.
* They check if something is **true** or **false**, and then run code **based on that result**.

1. **if Statement**: The if statement is used to execute a block of code only if a specified condition is true.

$age = 25;

if ($age > 18) {

echo "You are an adult.";

}

1. **if-else Statement**: The if-else statement is used to execute one block of code if a condition is true and another block if it's false.

if ($age > 18) {

echo "Adult";

} else {

echo "Not an adult";

}

1. **if-elseif-else** Statement: This checks **multiple conditions**, one by one.

$score = 85;

if ($score >= 90) {

echo "A grade";}

elseif ($score >= 75) {

echo "B grade";}

else {

echo "Try again";

}

1. **switch statement (Good for many options**): This checks the value of $day and runs the matching block.

$day = "Monday";

switch ($day) {

case "Monday":

echo "Start of the week";

break;

case "Friday":

echo "Almost weekend";

break;

default:

echo "Just another day";

}

2. Loops: Do-While , For , For Each :

* 1. **Discuss the difference between for loop, foreach loop, and do-while loop in PHP.**
* **Looping Statements**: Looping statements are used to repeat a block of code multiple times.

1. for loop: Used when you know how many times you want to repeat something.

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

echo $i;

}

This will print: 12345

2. foreach loop: Used for looping through arrays.

$colors = ["red", "green", "blue"];

foreach ($colors as $color) {

echo $color;

}

This will print: redgreenblue

3. do-while loop: Always runs the code at least once, then checks the condition.

$i = 1;

do {

echo $i;

$i++;

} while ($i <= 5);

This will print: 12345

* PHP Array and Array Functions

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

* **There are 3 types of array**

**1) Nemeric array**: These use **numeric keys** (starting from 0 by default).

Example:

$fruits = ["Apple", "Banana", "Mango"];

$fruits = array(0 => "Apple", 1 => "Banana", 2 => "Mango");

**2) Associate array :** These use **named keys** (strings).

Example:

$person = ["name" => "John","age" => 30,"city" => "New York"];

echo $person["name"]; // Output: John

**3) Multidemetional** **array**:

* These are arrays that **contain other arrays** as elements.
* Useful for storing complex data like tables or matrices.

Example:

$employees = [

["John", 28, "Engineer"],

["Alice", 32, "Designer"],

["Mark", 25, "Developer"]];

echo $employees[1][0]; // Output: Alice

* Header Function:

1. 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 allows you to control aspects of the HTTP response such as redirection, content type, cache control, and more.
* **Syntax**
* header(string $header, bool $replace = true, int $response\_code = 0);
* $header: The header string to send (e.g., Location:, Content-Type:).
* $replace (optional): If true, replaces a previous similar header. Defaults to true.
* $response\_code (optional): Forces the HTTP response code (e.g., 404, 200).
* **Important Rules**
* **Must be called before any output** (HTML, echo, or print).
* Headers are sent as part of the HTTP response, so no whitespace or content should be sent before using header().
* **Common Uses of header()**

**1. Redirecting to Another Page**

header("Location: https://example.com");exit; // Always follow redirects with exit

**2. Setting Content Type**

header("Content-Type: application/json");echo json\_encode(["status" => "success"]);

**3. Force File Download**

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

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

readfile("file.txt");

**4. Custom Status Codes**

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

echo "Page not found.";

* Include and Require:

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

* **Include** :

include => include/load page

include\_once => include/load page only once

include('include\_demo1.php');

include('include\_demo.php');

echo " Mornig";

include\_once('include\_demo.php');

include\_once('include\_demo.php');

echo " Mornig";

* **Require:**

require => require/load page

require\_once => require/load page only once

require('require\_demo1.php');

require('require\_demo.php');

echo " Mornig";

require\_once('require\_demo.php');

require\_once('require\_demo.php');

echo

" Mornig";

* **difference between include and require**
* Incude & require both same but if file not exits than Incude define E\_warning so script not terminate but Require gives Fetel Error so script terminate
* So ideal Include use
* PHP Expressions, Operations, and String Functions :

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

* What are **PHP Expressions**?

A **PHP expression** is **anything that has a value**.

* It's a **combination of values, variables, and operators** that PHP can **evaluate** to get a result.

**For example:**

$x = 5 + 3;

Here, 5 + 3 is an **expression**, and its value is 8.

| Operator | Meaning | Example | Result |
| --- | --- | --- | --- |
| + | Addition | 5 + 3 | 8 |
| - | Subtraction | 10 - 4 | 6 |
| \* | Multiplication | 2 \* 3 | 6 |
| / | Division | 10 / 2 | 5 |
| % | Modulus | 7 % 3 | 1 |

* **Arithmetic operators** **:**

* For example: $a=11;

$b=20;

$sum=$b+$a;

echo $sum;

* **Logical Expressions (Used for decisions):** These expressions return **true or false**, often used in if statements.

| **Operator** | **Meaning** | **Example** | **Result** |
| --- | --- | --- | --- |
| && | AND | True && false | false |
| || | OR |  | OR |
| ! | NOT | !true | false |

* **Logical operators** **:**
* For example: $age = 20;

if ($age >= 18 && $age <= 25) {

echo "Young adult";

}