Module 5 – Core PHP

PHP Syntax:

1) Discuss the structure of a PHP script and how to embed PHP in HTML.
   - PHP Opening and Closing Tags:
     - A PHP script begins with <?php and ends with ?>. Everything between these tags is processed as PHP code.
   - Variables and Data Types:
     - PHP supports different data types like integers, strings, arrays, and objects.
   - Control Structures:
     - Conditional statements if, if-else, switch, nested if-else and loops for, while allow logical operations.
2) What are the rules for naming variables in PHP?
   - Every variable in PHP begins with $, followed by the variable name.
   - The first character after $ must be a letter (A-Z, a-z) or an underscore _, but not a number.
     - $_abc=10; → True
     - $abc123=10; → True
     - $123abc=10; → False
     - $abc a123=10; → False
   - PHP variable names are case-sensitive. $name and $Name are two different variables.
   - Variable names cannot contain spaces or special symbols (@, #, !, etc.), except underscores.

PHP Variables:

1) Explain the concept of variables in PHP and their scope.
   - A variable in PHP is a container used to store data, such as numbers, strings, or objects.
   - Every variable starts with $ followed by the variable name.
   - Can store various data types like: Strings, integers, floats, arrays, objects etc..
   - PHP has three different variable scopes:
     - local: in function that not user out of function
     - global: out of function we can use any where
     - static: static keywords

Super Global Variables:

1) What are super global variables in PHP? List at least five super global arrays and their use.
    - Superglobal variables in PHP are predefined variables that are always accessible, regardless of scope - meaning you can access them from within functions, classes, or anywhere in your script without needing to declare them as global.
    - They are arrays and contain information about the server, environment, and user input.
    - $_GET: This array holds values passed to the script through the URL's query string (e.g., example.com?name=John&age=30).
        - It's used to retrieve data submitted via the GET method in an HTML form or directly in the URL. Use: Retrieving data from URL parameters.
        - Example: If the URL is example.com?product_id=123, you can access the product_id like this: echo $_GET["product_id"]; which will output 123.
    - $_POST: This array contains values submitted to the script through the HTTP POST method.
        - It's commonly used to collect data from HTML forms when the form's method attribute is set to "POST".
        - Use: Handling form submissions, especially for sensitive data.
        - Example: If a form has an input field named "username", you can access its value like this: echo $_POST["username"];
    - $_REQUEST: This array is a combination of $_GET, $_POST, and $_COOKIE.
        - It contains the contents of $_GET, $_POST, and $_COOKIE arrays. The order of variables in the array depends on the variables_order directive in the php.ini file.
        - Use: Accessing data from both GET and POST requests, as well as cookies.
        - Example: You can check if a variable exists in any of the request methods: if (isset($_REQUEST["my_variable"])) { ... }
    - $_COOKIE: This array holds values passed to the script via HTTP cookies.
        - Cookies are small pieces of data stored on the user's computer by the web server.
        - Use: Storing and retrieving user-specific information, such as session IDs or preferences.
        - Example: If a cookie named "user_id" is set, you can access its value like this: echo $_COOKIE["user_id"];

- $_SESSION: This array is used to store session variables.
    - Sessions are a way to store information about a user across multiple pages.
    - Session data is stored on the server and associated with a unique session ID, usually stored in a cookie.
    - Use: Maintaining user-specific data across multiple page requests.
    - Example: To store a variable named "username" in the session, you would do this: $_SESSION["username"] = "John";. To retrieve it later: echo $_SESSION["username"]

Conditions, Events, and Flows:

1) Explain how conditional statements work in PHP.
    - if Statement:
        - The if statement is the most basic conditional statement. It executes a block of code only if a specified condition is true.
    - if...else Statement:
        - The if...else statement provides an alternative block of code to execute if the if condition is false.
    - if...elseif...else Statement:
        - The if...elseif...else statement allows you to check multiple conditions in sequence. It's useful when you have more than two possible outcomes.
    - Nested if Statements:
        - You can nest if statements within each other to create more complex decision-making logic.
        - This means you can have an if statement inside another if, else, or elseif block.

Loops: Do-While, For Each, For Loop:

1) Discuss the difference between for loop, foreach loop, and do-while loop in PHP.
    - For loop:
        - The for loop is a fundamental control structure that allows you to execute a block of code a specific number of times.
        - It's highly versatile and gives you precise control over the loop's execution.
        -
```
$i=1;
for($i=1;$i<=50;$i++) //it print 1to 50   → for loop
{
        echo $i."<br>";
}
```

- For each loop:
    -
- Do-While Loop:
    - The do-while loop is a post-test loop, meaning the condition is checked after the code block is executed.
    - This guarantees that the code block runs at least once.

    -
      ```
      $i=1;
      do    → Do-while loop
      {
              echo $i . "<br>";
              $i++;
      ```

    -
      ```
      }while($i<=10);
      ```

Include and Require:

1) Explain the difference between include and require in PHP.
    - Include:
        - If the specified file is not found, include will generate a warning, but the script will continue to execute.
        - Incude define E_warning so script not terminate
        - This is useful when the included file is not essential for the script to function correctly. For example, you might use include for a file that contains optional features or specific configurations.
    - Require:
        - If the specified file is not found, require will generate a fatal error, and the script will stop executing.
        - his is used when the included file is crucial for the script's operation.
        - Require gives Fetel Error so script terminate
        - For instance, you would use require for files containing core functions, database connections, or class definitions that the script relies on.

PHP Array and Array Functions:

1) Define arrays in PHP. What are the different types of arrays?
    - Array is an ordered map. A map is a type that associates *values* to *keys*. This means that arrays can store multiple values in a single variable.
    - Numeric Array:
        - These arrays use numerical keys, starting from 0.
        - You can access elements using their index.
    - Associative Array:

        - These arrays use named keys (strings) instead of numerical indices.
        - They are useful for storing data where you want to associate values with meaningful labels.
    - Multidimensional Array:
        - These are arrays that contain one or more arrays as their elements.
        - They are useful for representing more complex data structures like tables or matrices.

Header Function:

1) What is the header function in PHP and how is it used?
    - The header() function in PHP is a powerful tool used to send raw HTTP headers to the client.
    - It's essential for controlling how the browser handles the response from the server.
    - Sending HTTP Headers: The primary purpose of the header() function is to send HTTP headers. This can include content type, caching policies, redirects, and more.
    - header('location:headerwelcome.php');

PHP Expressions, Operations, and String Functions:

1) Explain what PHP expressions are and give examples of arithmetic and logical operations.
    - In PHP, an expression is anything that has a value. It's the most fundamental building block of PHP code.
    - Expressions can be as simple as a single variable or a constant, or they can be more complex combinations of operators, function calls, and other expressions.
    - Arithmetical Operation
        - Addition (+): Adds two values
            - $sum = $a + $b;
        - Subtraction (-): Subtracts the second value from the first.
            - $sum = $a - $b;
        - Multiplication (*): Multiplies two values.
            - $sum = $a * $b;
        - Division (/): Divides the first value by the second.
            - $sum = $a / $b;
        - Modulo (%): Returns the remainder of a division.
            - $remainder = $a % $b;

- Logical Operation
    - And (&& or and): Returns true if both operands are true.
        - $result = $a && $b;
    - Or (|| or or): Returns true if either operand is true.
        - $result = $a || $b;
    - Not (!): Returns true if the operand is false, and false if the operand is true.
        - $result = !$a;