PHP Class: Introduction, Syntax, and Variables

1. Introduction to PHP

PHP (Hypertext Preprocessor) is a powerful server-side scripting language widely used for web development. It allows developers to create dynamic, data-driven web applications that can interact with databases and handle user input effectively. PHP is embedded within HTML, meaning it can be mixed directly with HTML code to build interactive pages easily.

PHP is open-source, easy to learn, and runs on almost every operating system including Windows, Linux, and macOS. It is especially popular because of its ability to connect with databases like MySQL, making it ideal for backend web development.

Today, PHP powers millions of websites, including content management systems (CMS) such as WordPress, Joomla, and Drupal.

- Example of a simple PHP page:

```
<html>
<body>
<?php
  echo 'Welcome to PHP!';
?>
</body>
</html>
```

In the example above, the PHP code inside <?php ... ?> runs on the server and sends the output (“Welcome to PHP!”) as plain HTML to the browser.

2. PHP Syntax

Every PHP script must be written inside special tags. The most common opening and closing tags are:

```
<?php ... ?>
```

When the web server encounters PHP tags, it processes the code inside them before sending the final output to the browser.

Basic PHP Script Structure

```
<?php
  // This is a PHP comment
  echo 'Hello, World!'; // This line prints a message
?>
```

• PHP statements end with a semicolon (;)
• PHP is case-sensitive for variables but not for function names or keywords.
• Comments can be added using // for single-line and /* ... */ for multi-line comments.

Embedding PHP in HTML

PHP can be easily mixed with HTML. For example:

```
<html>
<body>
  <h1>My First PHP Page</h1>
  <?php echo 'This text is generated by PHP!'; ?>
</body>
</html>
```

When this code runs, the HTML part is shown directly in the browser, while PHP dynamically generates its output.

3. PHP Variables

Variables in PHP are used to store data that can be reused throughout the script. A variable starts with a dollar sign ($), followed by the variable name.

Variable Declaration Example

```
<?php
$name = 'Emon';
```

```php
$age = 26;
$language = 'PHP';
echo 'My name is ' . $name . ', I am ' . $age . ' years old and I am learning ' .
$language . '.';
?>
```

In the example above:
• $name, $age, and $language are variables.
• Variables can hold values such as strings, numbers, or even arrays.
• The dot (.) operator is used to concatenate (join) strings together.

PHP Variable Naming Rules

- A variable starts with the $ sign.
- A variable name must start with a letter or underscore (_).
- A variable name cannot start with a number.
- A variable name can only contain letters, numbers, and underscores.
- Variable names are case-sensitive ($Name and $name are different).

Variable Example with Arithmetic

```php
<?php
$x = 10;
$y = 5;
$sum = $x + $y;
echo 'Sum: ' . $sum;
?>
```

In this example, $x and $y store numbers, and $sum stores their addition result. PHP automatically treats them as integers and performs arithmetic operations.

4. Data Types in PHP (Overview)

PHP supports several types of data that can be stored in variables. The main data types are:

- String – Sequence of characters (e.g., 'Hello')
- Integer – Whole numbers (e.g., 100, -45)

- Float (Double) – Decimal numbers (e.g., 3.14)
- Boolean – TRUE or FALSE
- Array – A collection of multiple values
- Object – Instance of a class (used in OOP)
- NULL – Represents an empty variable

String Example

```
<?php
$greeting = 'Hello, PHP Learners!';
echo $greeting;
?>
```

Strings can be enclosed in single (' ') or double (" ") quotes. Double quotes allow variable interpolation, meaning variable values can be inserted directly.

String Interpolation Example

```
<?php
$name = 'Emon';
echo "Welcome, $name!";
?>
```

Output: Welcome, Emon!
Using double quotes automatically replaces $name with its value.

5. Summary

In this one-hour class, we explored PHP's introduction, basic syntax, and variables. You learned how to write PHP code, declare variables, use data types, and embed PHP within HTML. These are the foundations of PHP programming. In the next class, we'll move on to control statements (if-else, loops) and working with user input via forms.