**Environment Setup**

1. Download & install XAMPP
2. Check your PHP version in terminal ***php -v***
3. Download and install composer
4. Check composer version in terminal ***composer -v***
5. Download XDebug
6. Download VS code

**XDebug Setup**

* Paste XDebug dll file inside XAMPP/php/ext
* Open php.ini and add properties
* ***xdebug.mode=debug***
* ***xdebug.start\_with\_request=yes***
* ***xdebug.client\_port=9003***
* ***xdebug.client\_host = localhost***
* ***zend\_extension="C:\xampp\php\ext\xdebug.dll"***
* Again check PHP version in terminal ***php -v***

**VS Code Extensions for PHP**

* PHP Debug
* PHP IntelliSense
* PHP Server
* MySQL by Weijan Chen

**Write your First PHP Code & Run**

* Write your first php code echo ***“Hello PHP”***
* Run from terminal command ***php hello.php***
* Run from custom server & port ***php -S 127.0.0.1:8000 hello.php***
* Run using PHP server extension

**PHP Variable**

**Variable Naming Rules:**

* Variable names in PHP start with a dollar sign ($).
* The name must begin with a letter or underscore (\_), followed by letters, numbers, or underscores.
* Variable names are case-sensitive, meaning **$myVariable** and **$myvariable** are treated as different variables.
* PHP does not require you to declare a variable before using it; it automatically creates the variable when you assign a value to it.

**Variable Assignment:**

$name = "John";

$age = 30;

**Variable Types:**

PHP is a loosely typed language, which means you don't need to declare the data type of a variable explicitly. PHP determines the data type based on the value assigned to it. Common data types in PHP include:

* Strings: Textual data enclosed in single ('') or double ("") quotes.
* Integers: Whole numbers.
* Floats (or Doubles): Numbers with decimal points.
* Booleans: True or false values.
* Arrays: Ordered collections of data.
* Objects: Instances of user-defined classes.
* Null: Represents a variable with no value.

**Variable Types:**

$name = "John Smith";

$age = 30;

$price = 19.99;

$isStudent = true;

$fruits = array("apple", "banana", "cherry");

class Person {

public $name;

public $age;

}

$person1 = new Person();

$person1->name = "Alice";

$person1->age = 25;

$noValue = null;

**Variable Scope:**

PHP variables have different scopes, which determine where the variable can be accessed. The main variable scopes in PHP are:

* **Local Scope:** Variables declared within a function are only accessible within that function.
* **Global Scope:** Variables declared outside of any function can be accessed anywhere in the script.
* **Superglobals:** Special global arrays like $\_GET, $\_POST, $\_SESSION, etc., which are accessible from anywhere in the script.

**Local Scope:** Variables declared within a function have local scope, meaning they are only accessible within that function.

function greet() {

$message = "Hello, World!";

echo $message;

}

greet(); /

**Global Scope:** Variables declared outside of any function have global scope and can be accessed from anywhere in your script.

$name = "John";

function greet() {

global $name;

echo "Hello, $name!";

}

greet(); // Outputs: Hello, John!

// You can also access $name outside of the function.

echo "Outside function: $name"; // Outputs: Outside function: John

$name is declared outside the greet() function and is made accessible inside the function using the global keyword.

**PHP Constant**

PHP, constants are like variables but with a significant difference: their values cannot be changed after they are defined. Constants are used to store values that should remain the same throughout your PHP script.

**Defining Constants:** Constants are defined using the define() function or the const keyword. The define() function is typically used for defining global constants, while the const keyword is used for class constants.

// Using define() for global constants

define("SITE\_NAME", "My Website");

define("MAX\_LOGIN\_ATTEMPTS", 3);

// Using const for class constants

class MathConstants {

const PI = 3.14159265359;

}

**Accessing Constants:**

You can access constants in the same way you access variables, but without the dollar sign ($) before the constant name.

echo SITE\_NAME; // Outputs: My Website

echo MAX\_LOGIN\_ATTEMPTS; // Outputs: 3

echo MathConstants::PI; // Outputs: 3.14159265359

**PHP Comments**

PHP, comments are used to annotate your code with explanatory text that is not executed as part of the program. Comments are ignored by the PHP interpreter and are solely for human readers to understand and document the code. PHP supports two types of comments: single-line comments and multi-line comments.

**Single-Line Comments:** Single-line comments are used for adding comments on a single line. They begin with a double forward slash (//) and continue until the end of the line.

// This is a single-line comment

**Multi-Line Comments:**

/\*

This is a multi-line comment.

It can span multiple lines.

\*/

**PHP printf**

PHP, the printf function is used for formatted output. It allows you to print formatted strings with placeholders that can be replaced by values.

$name = "Alice";

$age = 25;

printf("Name: %s, Age: %d", $name, $age);

Here %s is a placeholder for a string, and %d is a placeholder for an integer. The printf function replaces these placeholders with the values of $name and $age,

**PHP printf placeholders**

%s is used for inserting strings into the formatted output.

$name = "Alice";

printf("Name: %s", $name);

%d or %i is used for inserting signed decimal integers into the formatted output.

$age = 25;

printf("Age: %d", $age);

%f is used for inserting floating-point (decimal) numbers into the formatted output.

$price = 19.99;

printf("Price: %.2f", $price);

%x is used for inserting lowercase hexadecimal numbers into the formatted output.

$hexValue = 255;

printf("Hex Value: %x", $hexValue);

%X - Uppercase Hexadecimal Number:

$hexValue = 255;

printf("Hex Value: %X", $hexValue);

**PHP sprint**

In PHP, the sprintf function is used to format a string according to a specified format and returns the formatted string. It is similar to the printf function, but instead of printing the formatted string, sprintf allows you to store the formatted result in a variable or use it in further operations.

$name = "Alice";

$age = 25;

$formattedString = sprintf("Name: %s, Age: %d", $name, $age);

echo $formattedString;