Introduction to PHP

What is PHP?

PHP (Hypertext Preprocessor) is a widely-used, open-source scripting language primarily used for web development. It is embedded within HTML and executed on the server, making it an excellent choice for dynamic web applications.

Key Features of PHP

1. **Open Source** – Free to use and has a vast community support.
2. **Server-Side Execution** – Code runs on the server before sending the output to the client.
3. **Cross-Platform** – Compatible with major operating systems like Windows, macOS, and Linux.
4. **Database Integration** – Easily connects with MySQL, PostgreSQL, and other databases.
5. **Easy to Learn** – Has a simple syntax that resembles C, Java, and Perl.
6. **Security Features** – Provides mechanisms like encryption and authentication.

Basic PHP Syntax

PHP code is written inside <?php ... ?> tags:

```
<?php
  echo "Hello, World!";
?>
```

- **echo** is used to output text or HTML.
- PHP statements end with a semicolon (;).
- It can be embedded within HTML.

Difference Between PHP and Other Languages

PHP is a server-side scripting language mainly used for web development. Here's how it compares with other popular programming languages:

Feature	PHP	Python	Java	JavaScript	C++
Primary Use	Web Development (Server-Side)	General-Purpose & AI/ML	Enterprise Applications	Web Development (Client & Server)	System & Game Development
Execution	Server-Side	Server-Side	Compiled & JVM-Based	Client-Side & Server-Side (Node.js)	Compiled
Syntax	C-like, easy for beginners	Simple, readable	Strict & verbose	Lightweight, event-driven	Complex but powerful

Feature	PHP	Python	Java	JavaScript	C++
Performance	Moderate	Moderate	High	Fast (Client-Side)	Very High
Database Support	Strong (MySQL, PostgreSQL)	Strong (SQLite, PostgreSQL)	Strong (JDBC)	Uses APIs (MongoDB, Firebase)	Limited
Security	Moderate (Manual Security Handling)	High	High	Moderate	High
Use Cases	WordPress, Laravel, Web Apps	AI, Data Science, Web	Banking, Android Apps	Web Apps, Frontend, Node.js	OS, Games, Performance Apps

Key Takeaways:

- **PHP vs. Python** → PHP is better for web development, while Python excels in AI, ML, and automation.
- **PHP vs. Java** → Java is more robust for large-scale applications, whereas PHP is easier for web projects.
- **PHP vs. JavaScript** → PHP is server-side, while JavaScript is mainly client-side (though Node.js enables server-side JS).
- **PHP vs. C++** → PHP is for web apps, whereas C++ is used for performance-critical software.

PHP Variables and Constants

1. PHP Variables

A **variable** in PHP is used to store data, such as strings, numbers, or arrays. Variables in PHP:

- Start with a $ sign.
- Must begin with a letter or an underscore (_).
- Can contain letters, numbers, and underscores (_).
- Are case-sensitive ($name and $Name are different).
- Do not require explicit data type declaration (PHP is loosely typed).

Declaring a Variable in PHP

```
<?php
  $name = "John";  // String
  $age = 25;       // Integer
  $price = 99.99;  // Float
  $is_admin = true; // Boolean
```

```
 echo "Name: " . $name . "<br>";
 echo "Age: " . $age;
?>
```

Variable Scope in PHP

PHP has three types of variable scope:

1. **Local Scope** – Defined inside a function and accessible only there.
2. **Global Scope** – Defined outside functions and accessible using global keyword.
3. **Static Variables** – Retain their value even after the function execution.

Example of Scope

```
<?php
 $globalVar = "I am global"; // Global variable

 function myFunction() {
    global $globalVar; // Accessing global variable
    echo $globalVar;
 }

 myFunction();
?>
```

2. PHP Constants

A **constant** is a variable whose value cannot be changed after declaration. Constants in PHP:

- Are defined using define() or const.
- Do **not** start with a $ sign.
- Are **automatically global** in scope.

Declaring Constants

```
<?php
 define("SITE_NAME", "MyWebsite");  // Using define()
 const PI = 3.14159;  // Using const

 echo "Welcome to " . SITE_NAME;
 echo "Value of Pi: " . PI;
?>
```

Difference Between Variables and Constants

Feature	Variables	Constants
Declaration	Using $ sign	Using define() or const
Value Change	Can be changed	Cannot be changed
Scope	Local, Global, Static	Global by default
Syntax	$name = "John";	define("NAME", "John");

Types of Data in PHP

PHP supports multiple **data types**, categorized into:

1. **Scalar (Basic) Types**
2. **Compound (Complex) Types**
3. **Special Types**

1. Scalar Data Types (Stores a single value)

Data Type	Description	Example
String	A sequence of characters enclosed in single (') or double (") quotes.	"Hello, World!"
Integer	A whole number (positive or negative, without decimals).	100, -25, 0
Float (Double)	A number with a decimal point or in exponential form.	10.5, -3.14, 2.5e3
Boolean	Represents **true** or **false** values.	true, false

Examples of Scalar Types:

```
<?php
  $name = "John Doe";  // String
  $age = 30;           // Integer
  $price = 99.99;      // Float
  $is_available = true; // Boolean

  echo "Name: " . $name . "<br>";
  echo "Age: " . $age . "<br>";
  echo "Price: $" . $price . "<br>";
  echo "Available: " . ($is_available ? "Yes" : "No");
?>
```

2. Compound Data Types (Stores multiple values)

Data Type	Description	Example
Array	Stores multiple values in a single variable.	[1, 2, 3], ["apple", "banana"]
Object	Instances of user-defined classes.	new Car();

Examples of Compound Types:

```
<?php
  // Array Example
  $fruits = array("Apple", "Banana", "Cherry");
  echo "First fruit: " . $fruits[0];

  // Object Example
  class Car {
     public $brand = "Toyota";
  }

  $myCar = new Car();
  echo "<br>Car Brand: " . $myCar->brand;
?>
```

3. Special Data Types

Data Type	Description	Example
NULL	Represents a variable with no value.	$var = NULL;
Resource	Holds a reference to an external resource (e.g., database connection, file handle).	mysqli_connect()

Examples of Special Types:

```
<?php
  // NULL Example
  $data = NULL;
  var_dump($data);  // Outputs NULL

  // Resource Example (Database Connection)
  $conn = mysqli_connect("localhost", "root", "", "test_db");
  var_dump($conn);  // Outputs resource type if connected
?>
```

Type	Category	Example
String	Scalar	"Hello"
Integer	Scalar	100
Float (Double)	Scalar	3.14
Boolean	Scalar	true
Array	Compound	["Red", "Green", "Blue"]
Object	Compound	new ClassName();
NULL	Special	NULL
Resource	Special	mysqli_connect()

Variable Scopes in PHP

In PHP, **variable scope** determines where a variable can be accessed in a script. There are **four types of variable scopes**:

1. **Local Scope**
2. **Global Scope**
3. **Static Scope**
4. **Superglobal Variables**

1. Local Scope

- A variable declared inside a function is **local** to that function.
- It **cannot** be accessed outside the function.

Example of Local Scope

```
<?php
function myFunction() {
   $localVar = "I am local";
   echo $localVar;
}
myFunction();
// echo $localVar; // ✕ ERROR: Undefined variable outside function
?>
```

Output: I am local
📌 *$localVar is only available inside myFunction().*

2. Global Scope

- A variable declared **outside** a function is **global**.
- It **cannot** be accessed inside functions unless we use the global keyword or $GLOBALS array.

Example of Global Scope

```
<?php
$globalVar = "I am global"; // Global variable

function testFunction() {
   global $globalVar; // Accessing global variable
   echo $globalVar;
}

testFunction();
?>
```

Output: I am global
Without the global keyword, $globalVar wouldn't be accessible inside the function.

Using $GLOBALS Array

Another way to access a global variable inside a function:

```
<?php
$globalVar = "I am global";

function testFunction() {
   echo $GLOBALS['globalVar'];
}

testFunction();
?>
```

Output: I am global

3. Static Scope

- A **static variable** retains its value even after the function exits.
- Useful for counting function calls or maintaining state.

Example of Static Scope

```
<?php
function counter() {
   static $count = 0; // Static variable
   $count++;
   echo "Count: " . $count . "<br>";
}

counter();
counter();
counter();
?>
```

Output:

Count: 1
Count: 2
Count: 3

Without static, $count would reset to 0 each time.

4. Superglobal Variables

PHP provides **predefined global arrays** that can be accessed anywhere, even inside functions.

Superglobal	Description
$_GET	Collects data from URL parameters
$_POST	Collects data from form submissions
$_SESSION	Stores session data
$_COOKIE	Stores cookies
$_SERVER	Provides server details

Example of Superglobal Variable ($_SERVER)

```
<?php
echo "PHP Script Name: " . $_SERVER['PHP_SELF'];
?>
```

Outputs the current script's filename.

Summary of Variable Scopes in PHP

Scope	Where Defined	Accessible Inside Function?	Special Handling Required?
Local	Inside a function	✗ No	No
Global	Outside a function	✗ No (unless global or $GLOBALS used)	Yes
Static	Inside a function	✓ Yes	Uses static keyword
Superglobal	Built-in PHP global arrays	✓ Yes	No

PHP Operators

Operators in PHP are symbols used to perform operations on variables and values. The key types of operators are:

1. **Arithmetic Operators**
2. **Assignment Operators**
3. **Relational (Comparison) Operators**
4. **Logical Operators**
5. **Bitwise Operators**
6. **Ternary Operator**

1. Arithmetic Operators

These operators perform mathematical calculations like addition, subtraction, multiplication, etc.

Operator	Name	Example	Output
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x divided by $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	$x raised to the power of $y

Example

```
<?php
  $a = 10;
  $b = 5;
  echo "Addition: " . ($a + $b); // Output: 15
  echo "<br>Multiplication: " . ($a * $b); // Output: 50
?>
```

2. Assignment Operators

Assignment operators are used to assign values to variables.

Operator	Example	Equivalent To
=	$x = 5	$x = 5
+=	$x += 3	$x = $x + 3
-=	$x -= 2	$x = $x - 2
*=	$x *= 4	$x = $x * 4
/=	$x /= 2	$x = $x / 2
%=	$x %= 3	$x = $x % 3

Example

```
<?php
  $x = 10;
  $x += 5; // Equivalent to $x = $x + 5
  echo $x; // Output: 15
?>
```

3. Relational (Comparison) Operators

These operators compare two values and return true or false.

Operator	Name	Example	Output
==	Equal	$x == $y	true if $x is equal to $y
===	Identical	$x === $y	true if $x is equal to $y and same data type
!= or <>	Not Equal	$x != $y	true if $x is not equal to $y
!==	Not Identical	$x !== $y	true if $x is not equal to $y or not same type
>	Greater than	$x > $y	true if $x is greater than $y
<	Less than	$x < $y	true if $x is less than $y
>=	Greater than or equal to	$x >= $y	true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	true if $x is less than or equal to $y

Example

```
<?php
  $a = 10;
  $b = 5;
  echo ($a > $b) ? "a is greater" : "b is greater"; // Output: a is greater
?>
```

4. Logical Operators

These operators are used in conditional statements.

Operator	Name	Example	Description
&& or and	Logical AND	$x && $y	true if both $x and $y are true
`		oror`	Logical OR
!	Logical NOT	!$x	true if $x is false

Example

```
<?php
  $a = true;
  $b = false;
  echo ($a && $b) ? "Both true" : "One is false"; // Output: One is false
?>
```

5. Bitwise Operators

Bitwise operators perform operations at the binary level.

Operator	Name	Example	Binary Equivalent
&	AND	$x & $y	Performs bitwise AND
`	`	OR	`$x
^	XOR	$x ^ $y	Performs bitwise XOR

Operator	Name	Example	Binary Equivalent
~	NOT	~$x	Inverts bits of $x
<<	Left Shift	$x << 2	Shifts bits left by 2
>>	Right Shift	$x >> 2	Shifts bits right by 2

Example

```
<?php
  $x = 6; // 110 in binary
  $y = 3; // 011 in binary
  echo $x & $y; // Output: 2 (010 in binary)
?>
```

6. Ternary Operator (?:)

- A shorthand for if-else statements.
- Syntax: condition ? true_value : false_value

Example

```
<?php
  $age = 20;
  echo ($age >= 18) ? "Adult" : "Minor"; // Output: Adult
?>
```

Summary Table

Type	Operator	Description
Arithmetic	+, -, *, /, %, **	Mathematical calculations
Assignment	=, +=, -=, *=, /=, %=	Assign values to variables
Relational	==, !=, >, <, >=, <=	Compare values
Logical	&&, `	
Bitwise	&, `	, ^, ~, <<, >>`
Ternary	?:	Shortened if-else