PHP

PHP (Hypertext Preprocessor) is a versatile and widely used server-side scripting language for creating dynamic and interactive web applications. Whether you're a seasoned developer or a beginner eager to delve into the world of web development, this PHP tutorial is your gateway to mastering the intricacies of PHP programming.
What is a PHP File?

PHP files can contain text, HTML, CSS, JavaScript, and PHP code
PHP code is executed on the server, and the result is returned to the browser as plain HTML
PHP files have extension ".php"

Example:

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
?>

</body>
</html>
```

Out put:

My first PHP script!

PHP Print

PHP print statement can be used to print the string, multi-line strings, escaping characters, variable, array, etc. Some important points that you must know about the echo statement are:

- print is a statement, used as an alternative to echo at many times to display the output.
- The print statement can be used with or without parentheses: print or print().

Ex:

```
<!DOCTYPE html>
<html>
<body>

<?php
print "this is php sample :<br>";
```

```
print "Hello world!<br>";

?>

</body>
</html>
```

Output:

```
this is php sample :
Hello world!
```

Variable Declaration:

Variables in PHP start with the dollar sign (**$**) followed by the variable name. Variable names must begin with a letter or an underscore, followed by any combination of letters, numbers, or underscores. They are case-sensitive.

A variable can have a short name (like $x and $y) or a more descriptive name ($age, $carname, $total_volume).

```
<!DOCTYPE html>
<html>
<body>

<?php
$txt = "IIIT Basar.com";
echo "I love $txt!";
?>

</body>
</html>
```

Out put

I love IIIT Basar.com!

PHP Data Types

Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

PHP Boolean

Booleans are the simplest data type works like switch. It holds only two values: **TRUE (1)** or **FALSE (0)**. It is often used with conditional statements. If the condition is correct, it returns TRUE otherwise FALSE.

Example:

```
<?php
    if (TRUE)
        echo "This condition is TRUE.";
    if (FALSE)
        echo "This condition is FALSE.";
?>
```

Output:

```
This condition is TRUE.
```

PHP Integer

Integer means numeric data with a negative or positive sign. It holds only whole numbers, i.e., numbers without fractional part or decimal points.

```
?php
    $dec1 = 34;
    $oct1 = 0243;
    $hexa1 = 0x45;
```

```
5.      echo "Decimal number: " .$dec1. "</br>";
6.      echo "Octal number: " .$oct1. "</br>";
7.      echo "HexaDecimal number: " .$hexa1. "</br>";
8.  ?>
```

Output:

```
Decimal number: 34
Octal number: 163
HexaDecimal number: 69
```

PHP Float

A floating-point number is a number with a decimal point. Unlike integer, it can hold numbers with a fractional or decimal point, including a negative or positive sign.

Example:

```
1.  <?php
2.      $n1 = 19.34;
3.      $n2 = 54.472;
4.      $sum = $n1 + $n2;
5.      echo "Addition of floating numbers: " .$sum;
6.  ?>
```

Output:

```
Addition of floating numbers: 73.812
```

PHP String

A string is a non-numeric data type. It holds letters or any alphabets, numbers, and even special characters.

String values must be enclosed either within **single quotes** or in **double quotes**. But both are treated differently. To clarify this, see the example below:

Example:

```
<?php
    $company = "tcs";
    //both single and double quote statements will treat different
```

```
    echo "Hello $company";
    echo "</br>";
    echo 'Hello $company';
?>
```

Output:

```
Hello tcs
Hello $company
```

PHP Array

An array is a compound data type. It can store multiple values of same data type in a single variable.

Example:

```
?php
    $bikes = array ("Royal Enfield", "Yamaha", "KTM");
    var_dump($bikes);   //the var_dump() function returns the datatype and valees
    echo "</br>";
    echo "Array Element1: $bikes[0] </br>";
    echo "Array Element2: $bikes[1] </br>";
    echo "Array Element3: $bikes[2] </br>";
?>
```

Output:

```
array(3) { [0]=> string(13) "Royal Enfield" [1]=> string(6) "Yamaha" [2]=> string(3) "KTM" }
Array Element1: Royal Enfield
Array Element2: Yamaha
Array Element3: KTM
```

PHP object

bjects are the instances of user-defined classes that can store both values and functions. They must be explicitly declared.

Example:

```
<?php
        class bike {
```

```
        function model() {
            $model_name = "Royal Enfield";
            echo "Bike Model: " .$model_name;
           }
    }
    $obj = new bike();
    $obj -> model();
?>
```

Output:

```
Bike Model: Royal Enfield
```

PHP Operators:

Arithmetic operators
Assignment operators
Comparison operators
Increment/Decrement operators
Logical operators
String operators
Array operators
Conditional assignment operators

1. Arithmetic Operators:
Addition (+): Adds two operands.
Subtraction (-): Subtracts the right operand from the left operand.
Multiplication (*): Multiplies two operands.
Division (/): Divides the left operand by the right operand.
Modulus (%): Returns the remainder of the division.
Increment (++) and Decrement (--): Increase or decrease the value of a variable by one.
Ex:

```
<?php
$x = 10;
$y = 6;

echo $x + $y;
?>
```

Output:
16

Ex:2

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 10;
$y = 6;


echo $x + $y."<br>";
echo $x - $y."<br>";
echo $x * $y."<br>";
echo $x / $y;

?>

</body>
</html>
```

Output:
16
4
60
1.6666666666667

2. Assignment Operators:

Assignment (=): Assigns a value to a variable.
Addition Assignment (+=): Adds the right operand to the left operand and assigns the result to the left operand.
Subtraction Assignment (-=): Subtracts the right operand from the left operand and assigns the result to the left operand.
Multiplication Assignment (*=): Multiplies the left operand by the right operand and assigns the result to the left operand.
Division Assignment (/=): Divides the left operand by the right operand and assigns the result to the left operand.
Modulus Assignment (%=): Calculates the modulus of the two operands and assigns the result to the left operand.

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 50;
$x += 30;

echo "Assignment Operators($x)";
?>

</body>
</html>
```

Output:
Assignment Operators (80)

3 Comparison Operators:

Equal (==): Checks if two operands are equal.
Identical (===): Checks if two operands are equal and of the same data type.
Not Equal (!= or <>): Checks if two operands are not equal.
Not Identical (!==): Checks if two operands are not equal and of different data types.
Greater Than (>): Checks if the left operand is greater than the right operand.
Less Than (<): Checks if the left operand is less than the right operand.
Greater Than or Equal To (>=): Checks if the left operand is greater than or equal to the right operand.
Less Than or Equal To (<=): Checks if the left operand is less than or equal to the right operand.

Ex:1

```
<!DOCTYPE html>
<html>
```

```
<body>

<?php
$x = 100;
$y = 50;

var_dump($x > $y); // returns true because $x is greater than $y
?>

</body>
</html>
```

Output:
bool(true)

4. Logical Operators:

Logical AND (&& or and): Returns true if both operands are true.

Logical OR (|| or or): Returns true if either of the operands is true.

Logical NOT (!): Returns true if the operand is false and vice versa.

```
<!DOCTYPE html>
<html>
<body>

<p>if both conditions are true.</p>

<?php
$x = 100;
$y = 50;

if ($x == 100 and $y == 50) {
  echo "Hello world!";
}
?>

</body>
</html>
```

Output:

if both conditions are true.

Hello world!

5. String Operators:

- **Concatenation (.):** Concatenates two strings together.

6. Conditional Ternary Operator:

Ternary Operator (expr1 ? expr2 : expr3): If expr1 evaluates to true, it returns expr2, otherwise, it returns expr3.

These are some of the most commonly used operators in PHP. Understanding and mastering these operators are essential for effective PHP programming.

Control Structures:

control structures are used to control the flow of execution in a program. They enable you to make decisions, iterate over data, and execute code based on certain conditions. Here are some common control structures in PHP:

If Statement:

The **if** statement executes a block of code if a specified condition is true.

```
$x = 10;
if ($x > 5) {
   echo "x is greater than 5";
}
```

If-else Statement:

The **if-else** statement executes one block of code if the condition is true and another block of code if the condition is false.

```
$x = 10; if ($x > 5) { echo "x is greater than 5"; } else { echo "x is not greater than 5"; }
```

If-elseif-else Statement:

The **if-elseif-else** statement allows you to test multiple conditions.

```
$x = 10; if ($x > 10) { echo "x is greater than 10"; } elseif ($x < 10) { echo "x is less than 10"; } else { echo "x is equal to 10"; }
```

Switch Statement:

The **switch** statement is used to perform different actions based on different conditions.

```
$day = "Monday"; switch ($day) { case "Monday": echo "Today is Monday"; break; case "Tuesday": echo "Today is Tuesday"; break; default: echo "Today is neither Monday nor Tuesday"; }
```

While Loop:

The **while** loop executes a block of code as long as the specified condition is true.

```
$x = 1; while ($x <= 5) { echo "The number is: $x <br>"; $x++; }
```

Do-while Loop:

The **do-while** loop is similar to the **while** loop, but the condition is evaluated after executing the block of code.

```
$x = 1; do { echo "The number is: $x <br>"; $x++; } while ($x <= 5);
```

For Loop:

The **for** loop is used to execute a block of code a specified number of times.

```
for ($x = 0; $x <= 5; $x++) { echo "The number is: $x <br>"; }
```

Foreach Loop:

The **foreach** loop is used to iterate over arrays.

```
$colors = ["red", "green", "blue"]; foreach ($colors as $color) { echo "$color <br>"; }
```

Functions:
PHP using the **function** keyword followed by the function name and a pair of parentheses containing optional parameters. The function body is enclosed within curly braces **{}**. Here's an example:

```
<!DOCTYPE html>
<html>
<body>

<?php
function myMessage() {
  echo "Hello world!";
}

myMessage();
?>

</body>
</html>
```

Output:
Hello world!

Data from Web forms:

Creating the HTML Form: First, you need to create an HTML form in your webpage. This form will contain input fields where users can enter data. The form should have an action attribute pointing to the PHP file where you'll process the form data, and a method attribute specifying how the form data should be sent (usually either GET or POST).

```
<!DOCTYPE html>
<html>
<head>
  <title>Sample Form</title>
</head>
<body>
  <form action="process_form.php" method="post">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name"><br>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email"><br>

    <label for="message">Message:</label><br>
    <textarea id="message" name="message"></textarea><br>

    <input type="submit" value="Submit">
  </form>
</body>
</html>
```

Processing the Form Data in PHP

```
<?php
// Check if the form is submitted
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  // Retrieve form data
  $name = $_POST["name"];
  $email = $_POST["email"];
  $message = $_POST["message"];

  // Process the data (e.g., store in database, send email, etc.)
  // For simplicity, just echoing here
  echo "Name: $name <br>";
  echo "Email: $email <br>";
  echo "Message: $message <br>";
}
?>
```

Additional Considerations

1. **Validation and Sanitization**: Always validate and sanitize user input to prevent security vulnerabilities and ensure data integrity.
2. **Security**: Protect against SQL injection, XSS attacks, etc., especially if you're storing user data in a database or displaying it on your website.
3. **Feedback**: Provide meaningful feedback to users, especially if there are errors in their form submission.
4. **Form Design**: Make sure your forms are user-friendly and accessible across different devices and browsers.