Session 4

Variables and Operators in PHP

Session Overview

In this session, learners would be able to:

- Describe variables in PHP
- Describe various data types used in PHP
- Define the scope of PHP variables
- Explain handling of regular expressions
- Identify and elaborate on different types of operators used in PHP

Variables

Variables in a program are usually used for storing data or values that can be used anytime during the execution of a program.

A pictorial representation of the concepts of variables is shown in figure. Here, `empNumber` is a variable name.

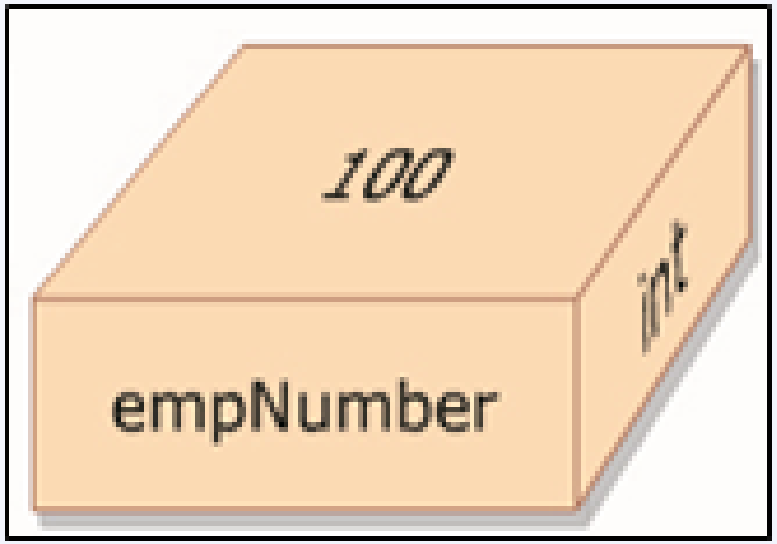

Figure: Variable

Variable Naming [1-2]

The names of all variable must start with the `$` sign. For example, `$my_var`. The syntax used to declare a variable is `$variablename=value;`.

Names of variables are case sensitive. Therefore, `$my_var` is different from `$MY_VAR`. `$ABC`, `$ABc`, `$Abc`, and `$abC` are four different variables.

Names of variables must start with an alphabet, which can be in either lowercase or uppercase, or an underscore. This is then followed by other characters. For example, `$my_var1` is a valid variable name, whereas `$1my_var` is not.

Variable Naming [2-2]

Names of variables cannot contain any spaces. Therefore, `'$first name'` is not a valid variable name. However, an underscore can be used instead of the space, such as `$first_name`.

The PHP interpreter automatically analyzes the values and appropriately assigns the correct datatype.

Once a variable has been declared, it can be reused several times throughout the code.

You can use the Assignment Operator (=) to assign any value to a variable.

PHP Local Variables

```
<?php
function Test() {
  $x = 5; // local scope
  echo "<p>Variable x inside function is: $x</p>";
}
Test();
// using x outside the function will generate an error
echo "<p>Variable x outside function is: $x</p>";
```

Code Snippet: Declaring a variable with local scope in PHP

Code Snippet shows how to declare a variable with local scope and demonstrates how it is used in PHP.

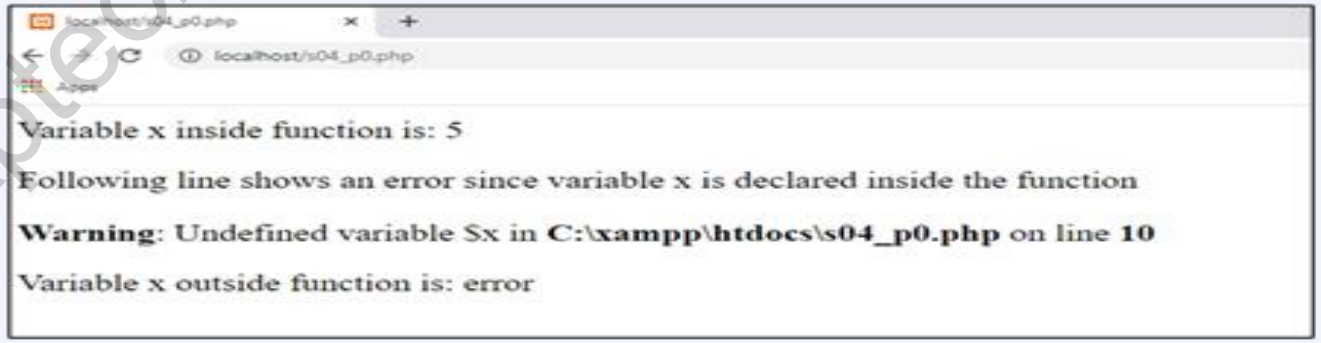

Figure: Output for Code Snippet

PHP Global Variables

```
<?php
$x = 5; // global scope
function Test() {
 // using x inside this function will generate an error
 echo "Following line shows an error since variable x is
 declared outside the function.";
 echo "<p>Variable x inside function is: $x error</p>";
}
Test();
echo "<p>Variable x outside function is: $x</p>";
?>
```

Code Snippet: Program to demonstrate a global variable

Code Snippet shows an example of a program to demonstrate a global variable.

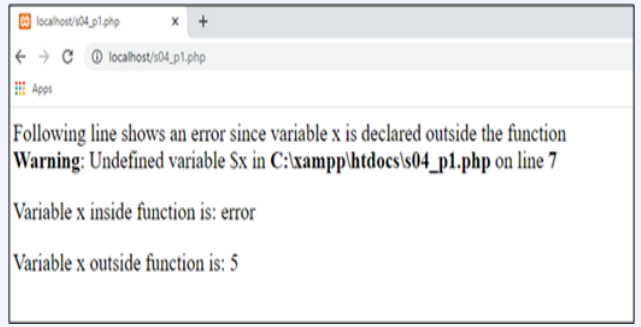

Figure: Output for Code Snippet

PHP Static Variables

```
<?php
    function static_variable()
    {
        static $X = 10;     //static variable
        $Y = 20;           // non-static variable

        $X++; //increment in static variable

        $Y++; //increment in non-static variable
        echo "Static: " .$X ."</br>";
        echo "Non-static: " .$Y ."</br>";
    }

//first function call
    static_variable();

//second function call
    static_variable();
?>
```

Code Snippet: Program to demonstrate a static variable

Code Snippet shows how to declare a static variable.

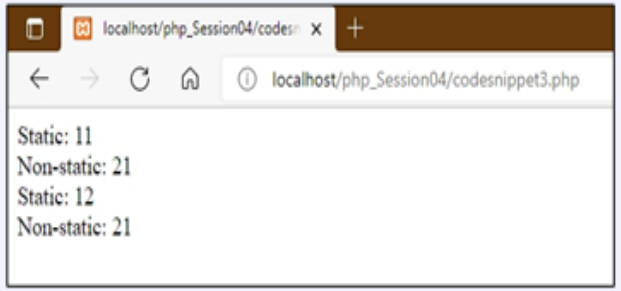

Figure: Output for Code Snippet

PHP Function Parameters

```
<?php
// multiply a value by 20 and return it to the caller
function multiply($value)
{
   $value=$value*20;
return $value;
}
$retval=multiply(10);
print "Return value is $retval\n";
?>
```

Code Snippet: Program with function parameters

Code Snippet shows a program with function parameters.

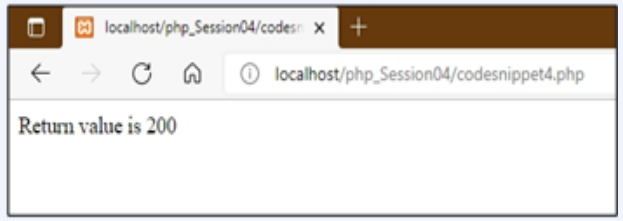

Figure: Output for Code Snippet

Variable Types [1-2]

Based on the attributes, data can be classified into different categories. Those categories are called as data types. There are eight data types that are used to construct variables.

Integers

- Are whole numbers without a decimal point. For example: 4195.

Doubles

- Are floating point numbers. For example: 49.1 or 3.14159.

Booleans

- Have only two possible values. They are either True or False.

Variable Types [2-2]

NULL

- Is a special type that has only one value, that is, NULL.

Strings

- Are sequences of characters. For example, 'PHP supports string operations'.

Arrays

- Are named and indexed collections of other values.

Objects

- Are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.

Resources

- Are special variables that hold references to resources that are external to PHP. For example, database connections.

Integers [1-2]

There are three functions supported in PHP, to check if the given variable is an integer:

- is_int()
- is_long() - alias of is_int()
- is_integer() - alias of is_int()

Following are some of the rules that apply to integers:

There must be at least one digit in an integer. For example, $int_var = 69.

There must not be any decimal point in an integer. For example, $int_var1 = 87654.

Integer can be specified in three different formats:

- Octal (8-based and prefixed with 0)
- Decimal (10-based)
- Hexadecimal (16-based and prefixed with 0x)

Integers [2-2]

```
<?php
// Check if the type of a variable is integer
$x1 = 1024;
var_dump(is_int($x1));
echo "The given number $x1 is an integer </br>";
// Check again...
$x2= 99.84;
var_dump(is_int($x2));
echo "The given number $x2 is an integer </br>";
echo"<br>";
$y=6987;
var_dump(is_int($y));
echo "The given number $y is an integer </br>";
?>
```

Code Snippet: Program to check whether the variable is an integer or not

Code Snippet shows a program to check whether the variable is an integer or not.

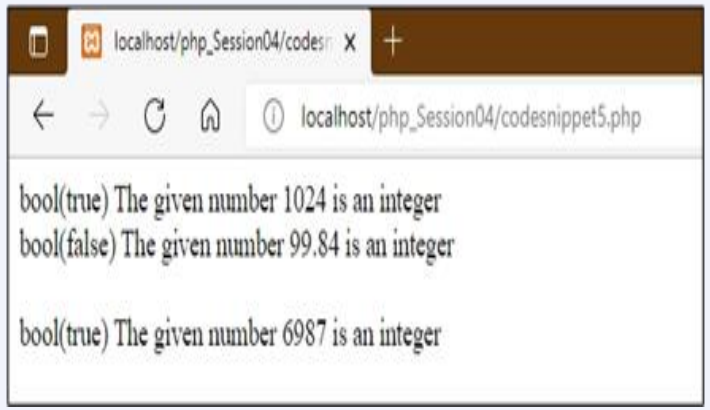

Figure: Output for Code Snippet

Doubles

```
<?php
 $many = 2.2888800;
 $many_2 = 2.2111200;
 $few = $many + $many_2;
 print("$many + $many_2 = $few <br>");
?>
```

Code Snippet: Program to print doubles

Code Snippet shows a program to print doubles.

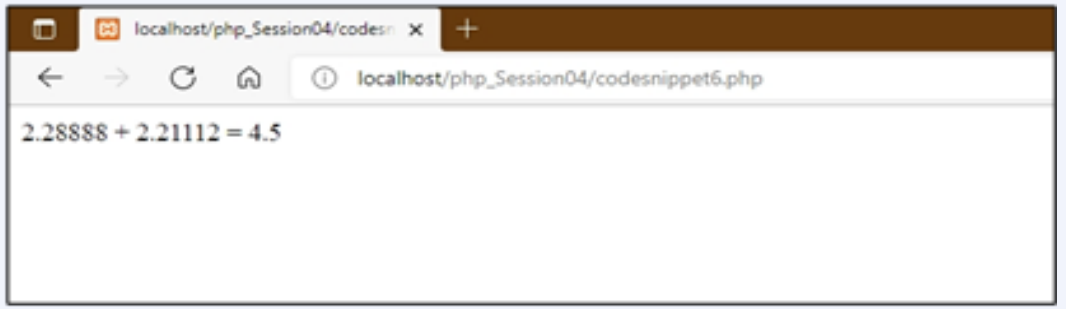

Figure: Output for Code Snippet

Boolean

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

Code Snippet: Program to understand the usage of Boolean data type

Booleans are the simplest data type and their operation is similar to an electrical toggle switch.

They can hold only one of the two values at a time, namely, TRUE (1) or FALSE (0**).**

Code Snippet shows a program to highlight the usage of Boolean data type.

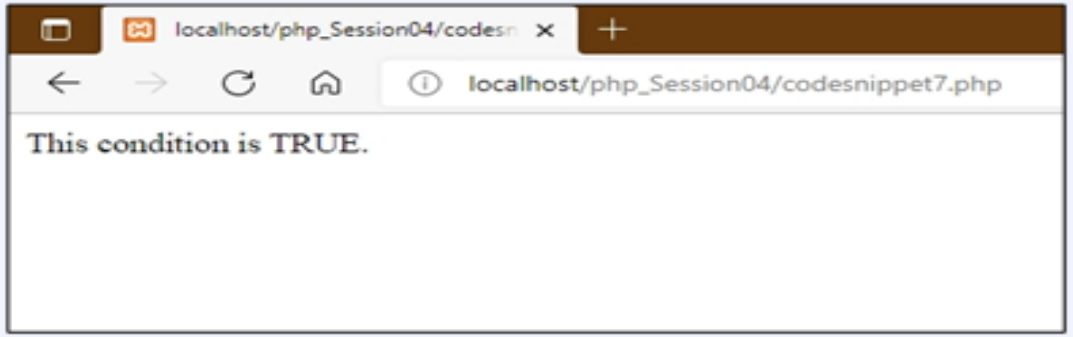

Figure: Output for Code Snippet

NULL

```
<?php
  $nl = NULL;
  echo $nl;  //it will not give any output
?>
```

Code Snippet: Program to show the usage of data type NULL

It is generally written in capital letters, as it is case sensitive.

The syntax is: `$my_var = null;`

Code Snippet shows a program that shows the usage of data type NULL.

In Code Snippet, since the variable has been assigned with a null value, there is no output.

Strings

A string is a series of characters, where the size of each character is one byte.

In PHP, only the 256-character set is supported and not the native Unicode.

However, in 32-bit systems, a string can have a size of 2 GB (maximum of 2147483647 bytes).

PHP Variables Scope

Variables can be declared anywhere within the script.

However, the scope of a variable will depend on where the variable has been declared.

There are three different variable scopes.

Local

- If you declare a variable within the function, then it has a local scope and visibility.
- Access to that is within that function only.

Global

- A variable that is declared outside a function has global visibility and scope.
- Access to that is anywhere within the script, even inside other functions.

Static

- Use this keyword in the variable declaration statement to retain local variable even after function.
- The variable is still local to the function.

PHP – Global Keyword

```
<?php
$x1 = 25;
$y1 = 10;
function myTest() {
  global $x1, $y1;
  $y1 = $x1 + $y1;
}
myTest();  // run the function
echo $y1; // display updated value for variable $y
?>
```

Code Snippet: Program using `global` keyword

In PHP, all global variables are stored in an array that is called `$GLOBALS[index]`. This holds the variable name.

Code Snippet shows a program using the `global` keyword.

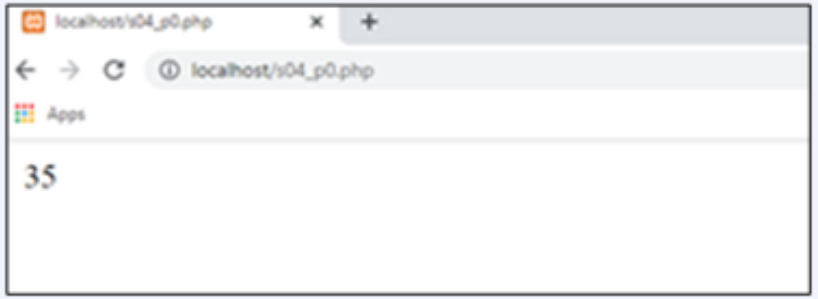

Figure: Output for Code Snippet

PHP – Static Keyword

```
<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}
myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
```

Code Snippet: Program using `static` keyword

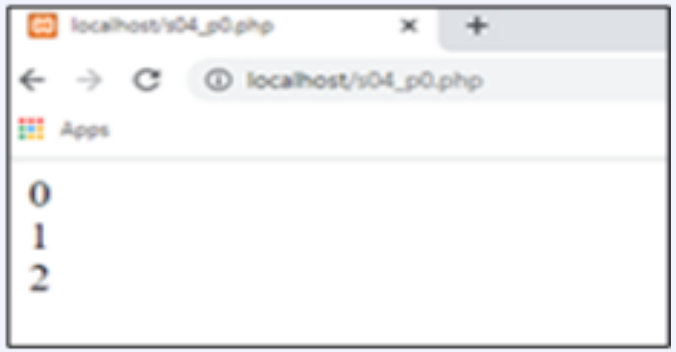

Figure: Output for Code Snippet

Code Snippet shows a program using the `static` keyword.

Regular Expression

Generally called regex, regular expressions are a sequence of characters that describe a specific search pattern in the form of text strings.

Regular expressions are strings composed of delimiters, a pattern, and optional modifiers.

The common delimiter that is most commonly used is the forward slash (/).

There are two types of Regular Expressions: POSIX Regular Expressions and PERL Style Regular Expressions.

POSIX Regular Expressions

Portable Operating System Interface for uniX (POSIX) is a collection of standards that are used to define some of the functionality that should be supported by UNIX operating system.

One such standard defines two types of regular expressions. POSIX regular expressions are to be used only with textual data.

If data contains a NUL-byte (`\x00`), the regular expression will interpret that as the end of the string and matching will not happen beyond that point.

Brackets (`[]`) can be used to find a range of characters. For example, if a match is to be found for any digit between 0 and 9, then, the expression is `[0-9]`.

To get a match for a lowercase character between a and z, use the expression `[a-z]`.

To get a match for a character between uppercase A and uppercase Z, use the expression `[A-Z]`.

To find a match for a character between lowercase a and uppercase Z, use the expression `[a-Z]`.

PHP's Regexp POSIX Functions

At present, PHP has seven different functions to search for strings using POSIX-style regular expressions. They are as follows:

- `ereg()`
- `ereg_replace()`
- `eregi()`
- `eregi_replace()`
- `split()`
- `spliti()`
- `sql_regcase()`

Operators in PHP

An operator is used to perform a variety of operations on variables and values.

Operator Types: There are different types of operators which perform different operations.

Arithmetic Operators

Arithmetic operators perform different arithmetic operations.

Operator	Name	Description
+	Addition	Returns the sum of the operands
-	Subtraction	Returns the difference between the two operands
*	Multiplication	Returns the product of two operands
/	Division	Returns the quotient after dividing the first operand by the second operand
%	Modulus	Returns the remainder after dividing the first operand by the second operand

Table: Arithmetic Operators in PHP

Comparison Operators

Comparison operators compare two operands.

Operator	Name	Description
==	Equal to	Returns true if both the operands are equal
===	Identical	Returns true if both the operands are equal and are of the same data type
!=	Not equal to	Returns true if the first operand is not equal to the second operand
<>	Not equal to	Returns true if the first operand is not equal to the second operand
!==	Not Identical	Returns true if the first operand is not equal to the second operand or they are not of the same data type
<	Less than	Returns true if the first operand is less than the second operand
<=	Less than or equal to	Returns true if the first operand is less than or equal to the second operand
>	Greater than	Returns true if the first operand is greater than the second operand
>=	Greater than or equal to	Returns true if the first operand is greater than or equal to the second operand

Table: Comparison Operators in PHP

Logical Operators

Logical operators connect multiple expressions. PHP supports various logical operators.

Operator	Name	General Form	Description
AND && 	Logical AND operator	Expression1 AND Expression 2 Expression1 && Expression2	Returns true only if both the expressions are true
OR ||	Logical OR operator	Expression1 OR Expression2 Expression1 || Expression2	Returns true if any one of the expressions is true
XOR	Logical XOR operator	Expression1 XOR Expression2	Returns true if either Expression 1 or Expression 2 is true, but not both
!	Logical NOT operator	!Expression	Returns true only if the condition is not true

Table: Logical Operators in PHP

Assignment Operators

Assignment operators are used to assign values to variables.

Besides the primary assignment operator represented by the symbol =, PHP also supports combined assignment operators.

+= and -= are examples of combined assignment operators that perform addition and subtraction in combination with assignment, respectively.

Combined operators can also be used with array union and string operators.

Shorthand Expression	Description
$a+= $b $a = $a + $b	Adds $a and $b and assigns the result to $a
$a-= $b $a = $a -$b	Subtracts $b from $a and assigns the result to $a
$a*= $b $a = $a*$b	Multiplies $a and $b and assigns the result to $a
$a/= $b $a = $a/$b	Divides $a by $b and assigns the quotient to $a
$a%= $b $a = $a%$b	Divides $a by $b and assigns the remainder to $a
$a.= $b $a=$abs	Concatenates $b with $a and assigns the result to $a

Table: Shorthand Operators (examples with the combination operators)

Conditional Operator

The conditional operator is used to perform various operations that are based on different conditions.

PHP supports the conditional operator `?:`. It is used to first evaluate an expression for a TRUE or FALSE value and then, execute one of the two given statements depending upon the result of the evaluation.

It is commonly used to evaluate the `if-then-else` operation.

Operator Categories

On the basis of the number of operands used, all the operators that have been discussed so far, can be categorized into different types as shown in Table.

Operator Category	Description
Unary prefix operators	Precede a single operand
Binary operators	Take two operands and execute different arithmetic and logical operations
Conditional or ternary operators	Take three operands. Based on how the first expression is evaluated, either the second or third expression will be evaluated
Assignment operators	Assign a value to a variable

Table: Operator Categories

Precedence of PHP Operators

The precedence of operators decides the order of execution of operators in an expression. Some operators have a higher precedence than others.

To override precedence and force an operation, use parentheses.

A few operators have an equal level of precedence. In such a case, the order of operations is decided by the order of associativity, that is left or right.

Summary

- Variables can be used to store and access data during program execution.
- Three types of variables, local, global, and static are supported by PHP.
- There are different rules to be followed while naming a variable.
- Variable types are integer, Boolean, doubles, character strings, and floating-point numbers.
- A string literal can be specified in different ways such as single quoted, double quoted, heredoc, or Nowdoc.
- You can categorize operators as unary prefix, binary, ternary, or assignment operators.
- PHP language provides support to different operator types such as arithmetic, comparison, logical, assignment, and conditional operators.
- A regular expression is a character sequence that forms a search pattern.