UNIT-1 Expression and control statement in PHP

PHP is an acronym for "PHP: Hypertext Preprocessor.
PHP is a “widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.”
PHP is an open-source scripting language.

Advantages of PHP

Performance: PHP script is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.

Open Source: PHP source code and software are freely available on the web. We can develop all the versions of PHP according to our requirement without paying any cost. All its components are free to download and use.

Familiarity with syntax: PHP has easily understandable syntax. Programmers are comfortable coding with it. Embedded: PHP code can be easily embedded within HTML tags and script.

Platform Independent: PHP is available for WINDOWS, MAC, and LINUX& UNIX operating system. A PHP application developed in one OS can be easily executed in other OS also.

Control: Different programming languages require long script or code, whereas PHP can do the same work in a few lines of code. It has maximum control over the websites like we can make changes easily whenever we want.

Database Support: PHP supports all the leading databases such as MySQL, SQLite, ODBC, etc.

Loosely Typed Language: PHP allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.

Error Reporting: PHP has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.

Security: PHP is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.

Web servers Support: PHP is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.

Helpful PHP Community: It has a large community of developers who regularly updates documentation, tutorials, online help, and FAQs. Learning PHP from the communities is one of the significant benefits.

Basic PHP syntax:

A PHP script can be placed anywhere in the document.
A PHP script starts with <?php and ends with ?>:
Syntax:

```
<? php
// PHP code goes here
?>
```

The default file extension for PHP files is ".php". A PHP file normally contains HTML tags, and some PHP scripting code.
Example: demo.php

```
<html>
<head>
<title>Hello World</title>
</head>
<body>
<?php echo "Hello, World!"; ?>
</body>
</html>
```

Commenting PHP code:

A comment is the portion of a program that exists only for the human reader and stripped out before displaying the programs result. There are two commenting formats in PHP

Single-line comments: They are generally used for short explanations or notes relevant to the local code. Here are the examples of single line comments.

```
<! DOCTYPE html>
<html>
<body>
<?php
// This is a single-line comment
# This is also a single-line comment
?>
</body>
</html>
```

Multi-lines comments: They are generally used to provide pseudo code algorithms and more detailed explanations when necessary. The multiline style of commenting is the same as in C. Here is an example of multi-lines comments:

```
<! DOCTYPE html>
<html>
<body>
<?php
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
?>
```

```
</body>
</html>
```

Variables in PHP

- Variable is an identifier which holds data or another one variable and whose value can be changed at the execution time of script.
- **Syntax**: $variable_name=value;
- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name can't start with a number.
- A variable name can only contain alpha-numeric characters and
- Underscores (A-z, 0-9 and _)
- Variable names are case-sensitive ($str and $STR both are two different)

Example:

```
<?php
$str="Hello world!";
$a=5;
$b=10.5;
echo "String is: $str </br>";
echo "Integer is: $x <br/>";
echo "Float is: $y <br/>";
?>
```

Data types in PHP

PHP data types are used to hold different types of data or values. PHP categorises data types into three types as

1. Scalar type:
   It is a single valued data like String, Integer, Float, Boolean etc.
2. Compound type:
   It is collection of values which includes data of type Array and Object.
3. Special type:
   It includes data type as NULL and Resource.

PHP String:

A string is a sequence of characters, like "Hello world!". A string can be any text inside quotes. You can use single or double quotes. Example

```
<?php
$str = "Hello world!";
$strs = 'Web Development using PHP';
echo $str;
echo "<br>";
echo $strs;
?>
```

PHP Integer:

An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

The following are the rules for integers:

- An integer must have at least one digit.
- An integer must not have a decimal point.

- An integer can be either positive or negative.
- Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation.

PHP Float:

A float (floating point number) is a number with a decimal point or a number in exponential form. In the following example $a is a float. The PHP var_dump() function returns the
data type and value:

```
<?php
$a = 15.18;
var_dump($a);
?>
```

PHP Boolean:

A Boolean represents two possible states: TRUE or FALSE.

```
$x = true;
$y = false;
```

Booleans are often used in conditional testing.

PHP Array:

Arrays are named and indexed collections of other values. An array stores multiple values in one single variable. In the following example $courses is an array. The PHP var_dump()
function returns the data type and value:

```
<?php
$courses=array ("DataStructures","Data Communications","PHP","Python");
var_dump($courses);
?>
```

PHP Object:

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

PHP NULL Value:

Null is a special data type which can have only one value: NULL. A variable of data type NULL is a variable that has no value assigned to it.

```
<?php
$a="How are you";
$b=null;
var_dump($a);
var_dump($b);
?>
```

PHP Resource:

The special resource type is not an actual data type. It is the storing of a reference to function and resources external to PHP. A common example of using the resource data type is a
database call.

Constant in PHP:

PHP constants are name or identifier that can't be changed during the execution of the script except for magic constants, which are not really constants. Conventionally, PHP constants should be defined in uppercase letters. The PHP constants can be defined by two ways:

1. Using define () function:

Use the define() function to create a constant. It defines constant at runtime.

The syntax of define () function is:
define (name, value, case-insensitive);
where:
Name: It specifies the constant name.
Value: It specifies the constant value.
Case-insensitive: Specifies whether a constant is case-insensitive. Default value is false. It
means it is case sensitive by default.
Example:

```
<?php
define (“MESSAGE”, “Welcome to the course”);
echo MESSAGE;
?>
```

2. Using const keyword:
PHP introduced a keyword const to create a constant. The const keyword defines constants at compile time. It is a language construct, not a function. The constant defined using const keyword is case-sensitive.

```
<? php
const MESSAGE=" Welcome to the course";
echo MESSAGE;
?>
```

Using define () function
PHP constants can also be used using define() function. There is another way to print the value of constants using constant () function instead of using the echo statement.
The syntax for the following constant function: constant (varName)

```
<?php
define("MSG", "Welcome to the course");
Echo MSG, "</br>";
echo constant("MSG");
//both are similar
?>
```

PHP Operators:
Operators are used to perform operations on variables and values. PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators

PHP Arithmetic Operators: PHP arithmetic operators are used to perform common arithmetic operations such as addition, subtraction, etc. with numeric values.

Operator	Name	Example	Result
+	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 and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (Introduced in PHP 5.6)

Example:

```
<?php
$x = 10;
$y = 4;
echo($x + $y); // 0utputs: 14
echo($x - $y); // 0utputs: 6
echo($x * $y); // 0utputs: 40
echo($x / $y); // 0utputs: 2.5
echo($x % $y); // 0utputs: 2
?>
```

PHP Assignment Operators:
The basic assignment operator in PHP is "=".

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

Example:

```
<?php
$x = 10;
echo $x; // Outputs: 10
$x = 20;
$x += 30;
echo $x; // Outputs: 50
$x = 50;
$x -= 20;
echo $x; // Outputs: 30
$x = 5;
$x *= 25;
echo $x; // Outputs: 125
$x = 50;
$x /= 10;
echo $x; // Outputs: 5
$x = 100;
$x %= 15;
echo $x; // Outputs: 10
?>
```

PHP Comparison Operators:

The comparison operators allow comparing two values, such as number or string.

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is **equal to** $y, **and they are of the same type**
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is **not equal** to $y, **or they are not of the same type**
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

Example:

```
<?php
$x = 25;
$y = 35;
$z = "25";
var_dump($x == $z); // Outputs: boolean true
var_dump($x === $z); // Outputs: boolean false
var_dump($x != $y); // Outputs: boolean true
var_dump($x !== $z); // Outputs: boolean true
var_dump($x < $y); // Outputs: boolean true
var_dump($x > $y); // Outputs: boolean false
var_dump($x <= $y); // Outputs: boolean true
var_dump($x >= $y); // Outputs: boolean false
```

?>

Incrementing/Decrementing Operators: The increment and decrement operators are used to increase and decrease the value of a variable.

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

Example:

```
<?php
$x = 10;
echo ++$x; // Outputs: 11
echo $x; // Outputs: 11
$x = 10;
echo $x++; // Outputs: 10
echo $x; // Outputs: 11
$x = 10;
echo --$x; // Outputs: 9
echo $x; // Outputs: 9
$x = 10;
echo $x--; // Outputs: 10
echo $x; // Outputs: 9
?>
```

Logical Operators: PHP logical operators are used to combine conditional statements

Operator	Name	Example	Explanation
and	And	$a and $b	Return TRUE if both $a and $b are true
Or	Or	$a or $b	Return TRUE if either $a or $b is true
xor	Xor	$a xor $b	Return TRUE if either $ or $b is true but not both
!	Not	! $a	Return TRUE if $a is not true
&&	And	$a && $b	Return TRUE if either $a and $b are true
||	Or	$a || $b	Return TRUE if either $a or $b is true

Example:

```
<?php
$num = 2014;
// number num is even or odd
if($numr % 2 == 0){
 echo "$num is even number.";
}
else
{
 echo "$num is odd number”;
}
?>
```

String Operators:

These operators are used to perform the operation on strings. There are two string operators in PHP, which are given below:

Operator	Name	Example	Explanation
.	Concatenation	$a . $b	Concatenate both $a and $b
.=	Concatenation and Assignment	$a .= $b	First concatenate $a and $b, then assign the concatenated string to $a, e.g. $a = $a . $b

Example:

```
<?php
$x = "Hello";
$y = " World!";
echo $x . $y; // Outputs: Hello World!
$x .= $y;
```

```
echo $x; // Outputs: Hello World!
?>
```

Array Operators:
The array operators are used in case of array. Basically, these operators are used to compare the values of arrays.

Operator	Name	Example	Explanation
+	Union	$a + $y	Union of $a and $b
==	Equality	$a == $b	Return TRUE if $a and $b have same key/value pair
!=	Inequality	$a != $b	Return TRUE if $a is not equal to $b
===	Identity	$a === $b	Return TRUE if $a and $b have same key/value pair of same type in same order
!==	Non Identity	$a !== $b	Return TRUE if $a is not identical to $b
<>	Inequality	$a <> $b	Return TRUE if $a is not equal to $b

Conditional Operator:
The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

Operator	Name	Example	Result
?:	Ternary	$x = *expr1* ? *expr2* : *expr3*	Returns the value of $x. The value of $x is *expr2* if *expr1* = TRUE. The value of $x is *expr3* if *expr1* = FALSE

Example:

```
<?php
 $a = 10;
 $b = 20;

 /* If condition is true then assign a to result otheriwse b */
 $result = ($a > $b ) ? $a :$b;

 echo " Largest Value is $result </br>";
?>
```

Spaceship operator in PHP

The "<=>" operator is referred to as the "spaceship operator" or "three-way comparison operator." It was first used in PHP 7 to compare two expressions and return a value based on their relationship.

These operators are used to compare values and it returns integer values. It returns -1, 0 or 1.

- If both the operands are equal, it returns 0.
- If the right operand is greater, it returns -1.
- If the left operand is greater, it returns 1.

Example:

```
<?php
  echo "Integer comparison<br>";
  print( 1 <=> 1);print("<br/>");
  print( 1 <=> 2);print("<br/>");
  print( 2 <=> 1);print("<br/>");
  print("<br/>");
  echo "float comparison<br>";
  print( 1.5 <=> 1.5);print("<br/>");
  print( 1.5 <=> 2.5);print("<br/>");
  print( 2.5 <=> 1.5);print("<br/>");
  print("<br/>");
  echo "String comparison<br>";
  print( "a" <=> "a");print("<br/>");
  print( "a" <=> "b");print("<br/>");
  print( "b" <=> "a");print("<br/>");
?>
```

Decision making control statements in PHP:

Decision making control or Conditional statements are used to execute different code based on different conditions.

PHP provides us with four conditional statements:

1. if statement
2. if…else statement
3. if…elseif…else statement
4. switch statement

The If statement

The if statement executes a piece of code if a condition is true.

The syntax is:

```
if (condition) {
// code to be executed in case the condition is true
}
```

A practical example would be:

```
<?php
$age = 18;
if ($age < 20) {
```

```
echo "You are a teenager";
}
?>
```
Because the condition is true, the result would be:
You are a teenager

The If. . . Else statement
The If. . . Else statement executed a piece of code if a condition is true and another piece of code if the condition is false.
The syntax is:
```
if (condition) {
// code to be executed in case the condition is true
}
else {
// code to be executed in case the condition is false
}
```

An example of an If. . . Else statement would be:
```
<?php
$age = 25;
if ($age < 20) {
echo "You are a teenager";
}
else {
echo "You are an adult";
}
?>
```
Because the condition is false, the result in this case would be:
You are an adult

The If. . . Elseif. . . Else statement
This kind of statement is used to define what should be executed in the case when two or more conditions are present.

The syntax of this case would be:
```
if (condition1) {
// code to be executed in case condition1 is true
}
```

```
elseif (condition2) {
// code to be executed in case condition2 is true
}
else {
// code to be executed in case all conditions are false
}
```

Example to demonstrate this:
```
<?php
$age = 3;
if ($age < 10) {
```

```
echo "You are a kid";
} elseif ($age < 20) {
echo "You are a teenager";
} else {
echo "You are an adult";
}
?>
```

The result would be:
You are a kid

Switch case statement:
The switch statement is similar to the series of if-else statements.
It first evaluates an expression and then compares it with the values of each case. If a case matches then the same case is executed.

Syntax:

```
switch (expression) {
  case label1:
    //code block
    break;
  case label2:
    //code block;
    break;
  case label3:
    //code block
    break;
  default:
    //code block
}
```

- The expression which must be a constant and it is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed
- The break keyword breaks out of the switch block
- The default code block is executed if there is no match

Example:

```
<?php
 $per=95;
 $ratings= (int)($per/10);
 echo "Your grade is ";
 switch($ratings)
 {
        case 10:
        case 9:
        case 8: echo "Distinction";
              break;
        case 7:
        case 6: echo "First Class";
              break;
        case 5: echo "Second Class";
              break;
        case 4: echo "PassCclass";
              break;
        default: echo "Fail.";

}
 ?>
```

LOOPs in PHP

PHP supports four different types of loops.
1. for loop
2. while loop
3. do-while loop
4. foreach loop

The for loop
It is used when the programmer knows in advance how many times the block of code should be executed.
This is the most common type of loop encountered in almost every programming language.

Syntax:
```
for (initialization; condition; step){
// executable code
}
```
An example where we use the for loop would be:

```
for ($i=0; $i < 5; $i++) {
echo "This is loop number $i";
}
```
The result of this code snippet would be:
This is loop number 0
This is loop number 1

This is loop number 2
This is loop number 3
This is loop number 4

The while loop
The while loop is used when we want to execute a block of code as long as a test expression continues to be true.

Syntax:

```
Initialization statement;
while (condition){
// executable code
}
```

An example where we use the while loop would be:

```
$i=0; // initialization
while ($i < 5) {
echo "This is loop number $i";
$i++; // step
}
```

The result of this code snippet would be just the same as before:
This is loop number 0
This is loop number 1
This is loop number 2
This is loop number 3
This is loop number 4

The do. . . while loop

The do...while loop is used when we want to execute a block of code at least once and then as long as a test expression is true.

Syntax:

```
do {
// executable code
}
while (condition);
```

An example where we use the do. . . while loop would be:

```
$i = 0; // initialization
do {
$i++; // step
echo "This is loop number $i";
}
while ($i < 5); // condition
```

Output:

This is loop number 1
This is loop number 2
This is loop number 3
This is loop number 4
This is loop number 5

The foreach loop

The foreach loop is used to loop through arrays, using a logic where for each pass, the array element is considered a value and the array pointer is advanced by one, so that the next element can be processed.

Syntax:

```
foreach (array as value) {
// executable code
}
```

An example where we use the foreach loop would be:

```
$var = array('a','b','c','d','e'); // array declaration
foreach ($var as $key) {
echo "Element is $key";
}
```

This time the first loop number would be 1, because the first echo was executed only after variable incrementation:

Element is a
Element is b
Element is c
Element is d
Element is e

Difference between for loop and foreach loop:

	For Loop	Foreach Loop
1	The for loop is a control structure for specifying iteration that allows code to be repeatedly executed.	The foreach loop is a control structure for traversing items in an array or a collection.
2	A for loop can be used to retrieve a particular set of elements.	The foreach loop cannot be used to retrieve a particular set of elements.
3	It performs iterations till the set condition is true.	It executes till the end of array elements.

4	Syntax:
	for(expr1; expr2; expr3)
	{
	  // block of code;
	}
		Syntax:
		foreach ($array as $value)
		{
		  // block of code;
		}
		
		foreach ($array as $key=>$value)
		{
		  // block of code;
		}