UNIT-I

EXPRESSIONS AND CONTROL STATEMENTS

Syllabus :

1.1	• History • Advantages Of PHP • Syntax of PHP
1.2	• Variables • Data Types • Expressions • Operators • Constants
1.3	Decision Making Control Statements : • if • if-else • nested if • switch • break • continue
1.4	Loop Control Structures : • while • do-while • for • foreach

UNIT-I

EXPRESSIONS AND CONTROL STATEMENTS

1.1 **Introduction to PHP** :

PHP, an acronym for Hypertext Preprocessor, is a widely-used open source general-purpose scripting language. It is a cross-platform, HTML-embedded server-side scripting language and is especially suited for web development.

Where

- Server-side means that PHP scripts execute on the Web server, not within the browser on your local machine.
- Cross-platform means that PHP scripts can run on many different operating systems and Web servers. PHP is available for the two most popular Web server configurations IIS and Apache.
- HTML embedded scripting language means that PHP statements and commands are actually embedded in your HTML documents.

(i) **History of PHP :**

1) PHP was developed by Rasmus Lerdorf in 1994. Early non-released versions were used on his home page to keep track of who was looking at his online resume.
2) The first version used by others was available sometime in early 1995 and was known as the Personal Home Page Tools.
3) In mid-1995 he extended them to work with web forms and to communicate with databases, and called this implementation "Personal Home Page/Forms Interpreter" or PHP/FI.
4) Zeev Suraski and Andi Gutmans rewrote the parser in 1997 and formed the base of PHP 3, changing the language's name to the recursive acronym *PHP: Hypertext Preprocessor*.
5) Latest version of PHP is 8.2 which was released on 8 December 2022 and it will be supported till 8 december 2025.

(ii)

Advantages of PHP :

Some of the most important advantages of PHP are as following :

1. **Open Source and Free of Cost**:

   One of the most vital advantages of PHP is that it is accessible to all. People can download it from an open-source and get it for free. One can download it anywhere and readily use it for web application development.

2. **Platform Independence:**

   Another important factor is that since PHP-based applications can run on any OS such as UNIX, Windows, Linux, etc., people can use it without worrying about a platform where one can use it.

3. **Easy loading**:

   One can load the PHP-based applications easily and connect them to a database. People mainly use it since it has a fast rate of loading even if it is over a slow internet connection and speed than other programming languages.

4. **User-friendly:**

   It has a less learning curve, and one can learn it quickly. The language is straightforward to use, and if one knows about C programming, they can catch on to PHP language quickly for application development.

5. **Stable:**

   Unlike other scripting languages, PHP is very stable over the years and can provide one with assistance and continuous support. They can get help over various versions as well.

6. **No lengthy code required:**

   As we mentioned earlier, it is a simple language that people can utilize for various purposes. It has such a quality that one can use it without having to write lengthy codes and sophisticated structures for any web application event.

7. **Flexible:**

   It is highly flexible, and people can readily use it to combine its function with various other programming languages. They can use the software packages as the foremost effective technology for every feature.

8. **Increased Job opportunity:**

   Since PHP is very popular, many developers and developing communities have evolved who have knowledge of this language. People who know the simple language can become potential candidates for jobs.

9. **Database connection:**

   It has a built-in database connection that helps to connect databases and reduce the trouble and the time to develop web applications or content-based sites altogether.

10. **Library support:**

    PHP has strong library support using which one can utilize the various function modules for data representation.

(iii) **Syntax of PHP** :

There are four different pairs of opening and closing tags which can be used in php. Here is the list of tags.

a) Default syntax

b) Short open Tags

c) Omit the PHP closing tag at the end of the file.

a) Default Syntax

The default syntax starts with "<? php" and ends with "?>".

Example:

```
<?php
echo "Default Syntax";
?>
```

b) Short open Tags

The short tags starts with "<?" and ends with "?>". Short style tags are only available when they are enabled in php.ini configuration file on servers.

Example:

```
<?
echo "PHP example with short-tags";
?>
```

c) Omit the PHP closing tag at the end of the file :

It is recommended that a closing PHP tag shall be omitted in a file containing only PHP code so that occurrences of accidental whitespace or new lines being added after the PHP closing tag, which may start output buffering causing uncalled for effects can be avoided.

Example:

```
<?php
echo "PHP example with short-tags";
```

1.2	**(i)**	**Variables in PHP :** The syntax for PHP variables is similar to C and most other programming languages. There are three primary differences: 1. Variable names must be preceded by a dollar sign ($). 2. Variables do not need to be declared before being used. 3. Variables are dynamically typed, so you do not need to specify the type (e.g., char, int, float, etc.). **Rules for Naming Variables :** A variable can have a short name, like x, or a more descriptive name, like carName. The rules for PHP variable names: • Variables in PHP starts with a $ sign, followed by the name of the variable • The variable name must begin with a letter or the underscore character • A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _) • A variable name should not contain spaces • Variable names are case sensitive (y and Y are two different variables). **Example** :- <?php $x = 5; $y = 4; echo $x + $y; ?> output : 9
		Variable Variables You can reference the value of a variable whose name is stored in another variable. For example: $foo = 'bar'; $$foo = 'baz'; After the second statement executes, the variable $bar has the value "baz".

Variable References

In PHP, references are how you create variable alternatives. To make $black an another name for the variable $white, use:
$black =& $white;
The old value of $black is lost.

Scope of Variable

PHP has three different variable scopes:

1) Local
2) global
3) static

1) Local Scope :

- A variable declared within a PHP function is local and can only be accessed within that function. (the variable has local scope):

```
<?php
$a = 5; // global scope
function myTest()
{
echo $a; // local scope
}
myTest();
?>
```

- The script above will not produce any output because the echo statement refers to the local scope variable $a, which has not been assigned a value within this scope.
- You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
- Local variables are deleted as soon as the function is completed.

2) Global Scope :

- Global scope refers to any variable that is defined outside of any function. Global variables can be accessed from any part of the script that is not inside a function. To access a global variable from within a function, use the global keyword:

```
<?php
$a = 5;
$b = 10;
function myTest()
{
global $a, $b;
$b = $a + $b;
}
myTest();
echo $b;
?>
```

Output:
15

3) Static Scope :

- When a function is completed, all of its variables are normally deleted. However, sometimes you want a local variable to not be deleted.
- To do this, use the static keyword when you first declare the variable:

```
<? php
function add()
{
static $a=10;
$b=20;
$a++;
$b++;
echo $a;
echo "<br>";
echo $b;
echo "<br>";
}
add();
add();
?>
```

Output :
11
21
12
21

- Then, each time the function is called, that variable will still have the information it contained from the last time the function was called.
- The variable is still local to the function.

(ii) **Data Types in PHP :**

The main data types used to construct variables are:

- **Integers** - whole numbers like 23, 1254, 964 etc.

Example:

```
<?php
$x = 55;
var_dump($x);
?>
```

Output-

```
int(55)
```

- **Float** - floating-point numbers like 46.2, 733.21 etc

Example:

```
<?php
$x = 15.36;
var_dump($x);
?>
```

Output-

```
float(15.36)
```

- **Booleans** - only two possible values, true or false.

```
<?php
$x = true;
$y = false;
Var_dump($x);
Var_dump($y);
?>
```

Output-

```
bool(true)
bool(false)
```

- **Strings** - set of characters, like Welcome to PHP Programming.

Example:

```
<?php
$x = "Hello world!";
var_dump($x);
?>
```

Output-

```
string(12)
```

- **Arrays** - named and indexed collections of other values

Example-

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

Output

```
array(3) {
[0]=>
string(5) "Volvo"
[1]=>
string(3) "BMW"
[2]=>
string(6) "Toyota"
}
```

- **Objects** - instances of predefined classes

Example-

```
<?php
    class fruits {
        function mango() {
            $desc = "Mango is king of fruits...";
            echo $desc.”<br>”;
          }
    }
    $obj = new fruits();
    $obj -> mango();
    Var_dump($obj);
?>
```

Output –
Mango is king of fruits...
object(fruits)#1 (0) { }

```
<?php
    class fruits {
        function mango() {
         $weight=50;
            $desc = "Mango is king of fruits...";
            $this->weight=$weight;
            echo $desc."It has weight ".$weight." gms.";
            echo "<br>";
          }
    }
    $obj = new fruits($weight);
    $obj->mango();
    var_dump($obj);
?>
```

Output :
Mango is king of fruits...It has weight 50 gms.
object(fruits)#1 (1) { ["weight"]=> int(50) }

- **Null** - special data type which can have only one value i.e. NULL.

Example:

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

Output –
NULL

- **Resource** – It is not an actual data type. It is the storing of a reference to functions and resources external to PHP e.g. Database call.

(iii)

Expressions and Operators :

Expressions :

Using variables within expressions to do something is what PHP is all about. The statement which gives value is called statements.

```
<?php
$name = 'Rob';
echo $name;
?>
```

Expression

Operators :

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

1) Arithmetic operators
2) Assignment operators
3) Comparison operators
4) Increment/Decrement operators
5) Logical operators
6) String operators
7) Array operators

1) Arithmetic operators :

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

2) Assignment operators :

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

3) String Operator :

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

4) Increment/ Decrement Operators:

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

5) Logical Operators:

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

6) Comparision Operators :

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
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

7) Array Operators :

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y
==	Equality	$x == $y	Returns true if $x and $y have the same key/value pairs
===	Identity	$x === $y	Returns true if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	Returns true if $x is not equal to $y
<>	Inequality	$x <> $y	Returns true if $x is not equal to $y
!==	Non-identity	$x !== $y	Returns true if $x is not identical to $y

(iv) **Constants in PHP :**

- Constants are like variables except that once they are defined they cannot be changed or undefined.
- A constant is an identifier (name) for a simple value. The value cannot be changed during the script.
- A valid constant name starts with a letter or underscore (no $ sign before the constant name).
- Note: Unlike variables, constants are automatically global across the entire script.
- To create a constant, use the define() function.

Syntax

define(name, value, case-insensitive)

Parameters:

- name: Specifies the name of the constant
- value: Specifies the value of the constant
- case-insensitive: Specifies whether the constant name should be case-insensitive. Default is false.

Example :

Program-1(case insensitive is false)

```
<?php
define("GREETING", "Welcome to PHP Programming…");
echo GREETING;
?>
```

Output :

Welcome to PHP Programming…

Program-2(case insensitive is true)

```
<?php
define("Greeting", "Welcome to PHP Programming…",true);
echo GREETING;
?>
```

Output :

Welcome to PHP Programming…

1.3

Decision making Control Statements :-

Decision making or flow control is the process of determining the order in which statements execute in a program.

- **PHP If Else**

PHP if else statement is used to test condition. There are various ways to use if statement in PHP.

1) if
2) if-else
3) if-else-if
4) nested if

(i)

PHP If Statement

- PHP if statement allows conditional execution of code. It is executed if condition is true.
- If statement is used to executes the block of code exist inside the if statement only if the specified condition is true.
- **Syntax**

```
if(condition){
//code to be executed
}
```

- **Example**

```
<?php
$num=12;
if($num<100){
echo "$num is less than 100";
}
?>
```

- **Output**:

12 s less than 100

(ii)

PHP If-else Statement

- PHP if-else statement is executed whether condition is true or false.
- If-else statement is slightly different from if statement. It executes one block of code if the specified condition is true and another block of code if the condition is false.
- **Syntax**

```
if(condition){
```

```
//code to be executed if true
}else
{
//code to be executed if false
}
```

- **Example**

```
<?php
$num=12;
if($num%2==0){
echo "$num is even number";
}else{
echo "$num is odd number";
}
?>
```

- **Output:**

12 is even number

(iii)

PHP If-else-if Statement(else-if ladder)

- The PHP if-else-if is a special statement used to combine multiple if.else statements. So, we can check multiple conditions using this statement.
- **Syntax**

```
if (condition1){
//code to be executed if condition1 is true
} elseif (condition2){
//code to be executed if condition2 is true
} elseif (condition3){
//code to be executed if condition3 is true
...
} else{
//code to be executed if all given conditions are false
}
```

- **Example**

```
<?php
$marks=69;
if ($marks<33){
echo "fail";
}
else if ($marks>=34 && $marks<50) {
echo "D grade";
}
```

```
else if ($marks>=50 && $marks<65) {
echo "C grade";
}
else if ($marks>=65 && $marks<80) {
echo "B grade";
}
else if ($marks>=80 && $marks<90) {
echo "A grade";
}
else if ($marks>=90 && $marks<100) {
echo "A+ grade";
}
else {
echo "Invalid input";
}
?>
```

- **Output:**
  B Grade

(iv)

PHP nested if Statement

- The nested if statement contains the if block inside another if block. The inner if statement executes only when specified condition in outer if statement is true.
- **Syntax**

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

- **Example**

```
<?php
$age = 23;
$nationality = "Indian";
//applying conditions on nationality and age
if ($nationality == "Indian")
{
if ($age >= 18) {
echo "Eligible to give vote";
}
```

```
else {
echo "Not eligible to give vote";
}
}
?>
```

- **Output**:
  Eligible to give vote

(v)

PHP Switch

- PHP switch statement is used to execute one statement from multiple conditions. It works like PHP if-else-if statement.
- **Syntax**:

```
switch(expression){
case value1:
//code to be executed
break;
case value2:
//code to be executed
break;
.....
default:
code to be executed if all cases are not matched;
}
```

- **Example**

```
<?php
$num=20;
switch($num){
case 10:
echo("number is equals to 10");
break;
case 20:
echo("number is equal to 20");
break;
case 30:
echo("number is equal to 30");
break;
default:
echo("number is not equal to 10, 20 or 30");
}
?>
```

- **Output**:
  number is equal to 20

(vi)

PHP Break Statement

- PHP break statement breaks the execution of the current for, while, dowhile, switch, and for-each loop. If you use break inside inner loop, it breaks the execution of inner loop only.
- The break keyword immediately ends the execution of the loop or switch structure.
- It breaks the current flow of the program at the specified condition and program control resumes at the next statements outside the loop.
- The break statement can be used in all types of loops such as while, dowhile, for, foreach loop, and also with switch case.
- **Syntax**

  statements;
  break;

- **Example**
  Let's see a simple example to break the execution of for loop if value of i is equal to 5.

```
<?php
for($i=1;$i<=10;$i++){
echo "$i <br/>";
if($i==5){
break;
}
}
?>
```

Output:
1
2
3
4
5

(vii)

PHP continue statement

- The PHP continue statement is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
- The continue statement is used within looping and switch control structure when you immediately jump to the next iteration.
- The continue statement can be used with all types of loops such as - for, while, do-while, and foreach loop. The continue statement allows the user to skip the execution of the code for the specified condition.

- **Syntax**

  The syntax for the continue statement is given below:

```
Statements;
Continue;
```

- **Example**

  In the following example, we will print only those values of i and j that are same and skip others.

```
<?php
//outer loop
for ($i =1; $i<=3; $i++) {
//inner loop
for ($j=1; $j<=3; $j++) {
if (!($i == $j) ) {
continue; //skip when i and j does not have same values
}
echo $i.$j;
echo "</br>";
}
}
?>
```

Output:

```
11
22
33
```

1.4 **Loop Control Structure**

(i) **PHP While Loop**

- PHP while loop can be used to traverse set of code like for loop. The while loop executes a block of code repeatedly until the condition is FALSE. Once the condition gets FALSE, it exits from the body of loop.It should be used if the number of iterations is not known.
- The while loop is also called an Entry control loop because the condition is checked before entering the loop body. This means that first the condition is checked. If the condition is true, the block of code will be executed.
- **Syntax**

```
while(condition){
//code to be executed
}
```

- **Alternative Syntax**

```
while(condition):
//code to be executed
endwhile;
```

PHP While Loop Flowchart

PHP While Loop Example

```
<?php
$n=1;
while($n<=10){
 echo "$n<br/>";
$n++;
}
?>
```

- **Output:**

```
1
2
3
4
5
6
7
8
9
10
```

(ii)

PHP do-while loop

- PHP do-while loop can be used to traverse set of code like php while loop. The PHP do-while loop is guaranteed to run at least once.
- The PHP do-while loop is used to execute a set of code of the program several times. If you have to execute the loop at least once and the number of iterations is not even fixed, it is recommended to use the do-while loop.
- It executes the code at least one time always because the condition is checked after executing the code.
- The do-while loop is very much similar to the while loop except the condition check.
- The main difference between both loops is that while loop checks the condition at the beginning, whereas do-while loop checks the condition at the end of the loop.
- **Syntax**

```
do{
//code to be executed
}while(condition);
```

Flowchart

Example

```
<?php
$n=1;
do{
echo "$n<br/>";
$n++;
}while($n<=10);
?>
```

- **Output**:

```
1
2
3
4
5
6
7
8
9
10
```

(iii)

PHP For Loop

- PHP for loop can be used to traverse set of code for the specified number of times.
- It should be used if the number of iterations is known otherwise use while loop. This means for loop is used when you already know how many times you want to execute a block of code.
- It allows users to put all the loop related statements in one place. See in the syntax given below:
- **Syntax**

```
for(initialization; condition; increment/decrement){
/code to be executed
}
```

- **Parameters**

  The php for loop is similar to the java/C/C++ for loop. The parameters of for loop have the following meanings:
  - *initialization* - Initialize the loop counter value. The initial value of the for loop is done only once. This parameter is optional.
  - *condition* - Evaluate each iteration value. The loop continuously executes until the condition is false. If TRUE, the loop execution continues, otherwise the execution of the loop ends.
  - *Increment/decrement* - It increments or decrements the value of the variable.

- **Example**

```
<?php
for($n=1;$n<=10;$n++)
{
echo "$n<br/>";
}
?>
```

- **Output**:

```
1
2
3
4
5
6
7
8
9
10
```

(iv)

PHP foreach Loop

- The foreach loop works only on arrays, and is used to loop through each key/value pair in an array.

- **Syntax**

```
foreach ($array as $value)
{
code to be executed;
}
```

- For every loop iteration, the value of the current array element is assigned to $value and the array pointer is moved by one, until it reaches the last array element.

- **Example**

```
<?php
$colors = array("red", "green", "blue", "yellow");
foreach ($colors as $value) {
echo "$value <br>";
}
?>
```

- **Output**

```
red
green
blue
yellow
```

*********END **********

Important Questions :

1) Write Any five Advantages of PHP.
2) State the use of "$" sign in PHP.
3) Write the syntax of PHP.
4) Explain the term :
    a) Variable
    b) Expression
5) Explain any Four data types in PHP.
6) List different types of operators in PHP.
7) Write a program using foreach loop.