UNIT – 1 PHP Basic

BCA SEM – 2

WEB PROGRAMMING

Code : CS-09

Presented By : Dhruvita Savaliya

Topics :

- Introduction to PHP
- PHP configuration in IIS & Apache Web server
- Understanding of PHP.INI file
- Understanding of PHP .htaccess file
- PHP Variable
- Static & global variable
- GET & POST method
- PHP Operator
- Conditional Structure & Looping Structure
- Array
- User Defined Functions:
  - argument function
  - default argument
  - variable function
  - return function
- Variable Length Argument Function
  - func_num_args , func_get_arg, func_get_args
- Built in Functions –
  - Variable Functions - String Function - Math Function - Date Function - Array Function - Miscellaneous Function - File handling Function

History of PHP :

- PHP was conceived sometime in the fall of **1994 by Rasmus Lerdorf (**Common Gateway Interface**(CGI))**. Early non-released versions were used on his home page to keep track of who was looking at his online resume.
- The first version used by others was available sometime in early 1995 and was known as the **Personal Home Page Tools**.
- It consisted of a very simplistic parser engine that only understood a few special macros and a number of utilities that were in common use on home pages back then. A guestbook, a counter and some other stuff.
- The parser was rewritten in mid-1995 and named PHP/FI Version 2. The FI came from another package Rasmus had written which interpreted html form data. (**PHP/FI** : Personal Home Page /Forms Interpreter)
- He combined the Personal Home Page tools scripts with the Form Interpreter and added mySQL support and PHP/FI was born. PHP/FI grew at an amazing pace and people started contributing code to it.

Logo of PHP :

Version Of PHP

Version of PHP	Released date
1.0	8th June 1995
2.0	1st Nov. 1997
3.0	6th June 1998
4.0	22nd May 2000
4.1	10th Dec. 2001
4.2	22nd April 2002
4.3	27th Dec. 2002
4.4	11th July 2005
5.0	13th July 2004
5.1	24th Nov 2005
5.2	2nd Nov 2006

Version of PHP	Released date
5.3	30th June 2009
5.4	1st March 2012
5.5	20th June 2013
5.6	28th August 2014
6.X	Not released
7.0	3rd Dec. 2015
7.1	1st Dec. 2016
7.2	30th Nov. 2017
7.4	28th Nov. 2019
8.0	26th Nov. 2020
8.1	25th Nov. 2021
8.2	24th Nov. 2022

Introduction to PHP:

- PHP started out as a small open source project that evolved as more and more people found out how useful it was. **Rasmus Lerdorf** unleashed the first version of PHP way back in **1994**.
- PHP is a recursive acronym for "**PHP: Hypertext Preprocessor**".
- PHP is a server side scripting language that is embedded in HTML.
- It is used to create dynamic website.
- It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as
- POP3(Post Office Protocol), IMAP(Internet Messaging Access Protocol) and LDAP(Lightweight Directory Access Protocol).
- PHP4 added support for Java and distributed object architectures (COM and COBRA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

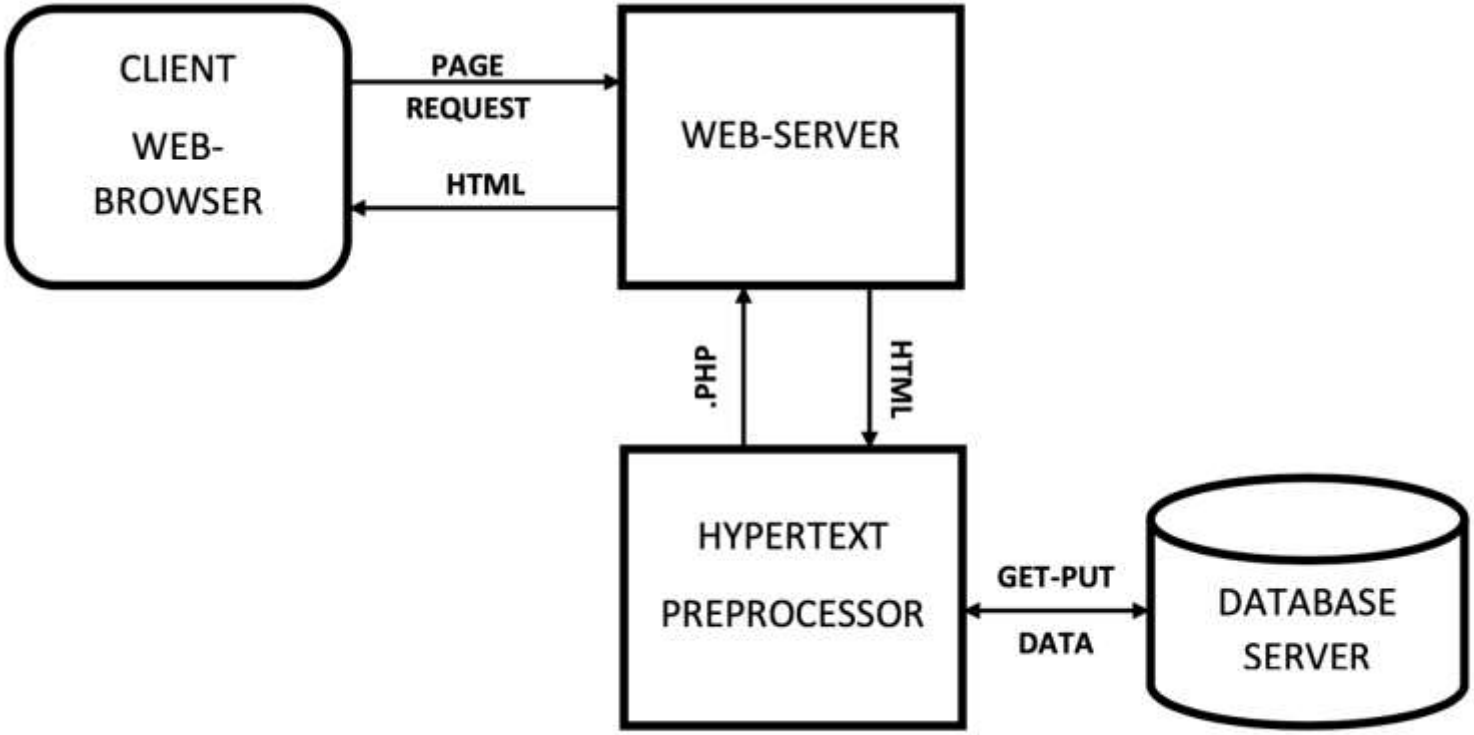
Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP

- Five important characteristics make PHP's practical nature possible −
- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

How PHP Works



What is Web Server :

- File servers, database servers, mail servers, and web servers use different kinds of server software. Each of these applications can access files stored on a physical server and use them for various purposes.
- The job of a web server is to serve websites on the internet. To achieve that goal, it acts as a middleman between the server and client machines. It pulls content from the server on each user request and delivers it to the web.
- The biggest challenge of a web server is to serve many different web users at the same time — each of whom is requesting different pages. Web servers process files written in different **programming languages** such as **PHP**, Python, Java, and others.

Presented By : Dhruvita Savaliya

What is IIS :

- IIS (Internet Information Services) is Microsoft's web server and has been around since 1995. The latest version, IIS 10, comes bundled with Windows and as such, IIS is a free product.
- There is a lightweight version of IIS called IIS Express that can be installed separately, but is intended for development purposes only. IIS Express only supports HTTP, HTTPS ,FTP, FTPS, SMTP, NNTP protocols and by default, merely supports local requests.

What is Apache web server :

- Apache is a free and open-source software that allows users to deploy their websites on the internet.
- It is one of the oldest and most reliable web server software maintained by the Apache Software Foundation, with the first version released in 1995.

What is php.ini File :

- **php.ini** was a special file provided as a default configuration file.
- **Purpose :**
- It's a very essential configuration file that controls what a user can or cannot do with the website.
- Each time PHP is initialized, the php.ini file is read by the system.
- Sometimes you need to change the behavior of PHP at runtime, then this configuration file is to use.
- All the settings related to registering global variables, uploading maximum size, display log errors, resource limits, the maximum time to execute a PHP script and others are written in a file as a set of directives that helps in declaring changes.
- The php.ini file is the default configuration file for running applications that require PHP. It is used to control variables such as upload sizes, file timeouts, and resource limits.
- Keys in the file are case-sensitive, keyword values are not spaces and lines starting with a semicolon are ignored. The file is well commented on. The Boolean values are represented by **On/Off, 1/0, True/False, Yes/No.**

What is php .htaccess file :

- **.htaccess** is a configuration file for use on web servers running on the web apache server software. when a **.htaccess** file is placed in a directory which in turn loaded via the Apache web server, then the .htaccess file detected and executed by the Apache server software.
- **.htaccess** files can be utilized to modify the setup of the Apache server software to empower additional functionality and fetures that the apache web server softwatre brings to the table. We can use the .htaccess file for alteration various configuration in apache web server software.
- Like :

  ErrorDocuments

  Password protection

  Redirection

  Deny visitors by IP address

  Adding MIME type

PHP File structure/syntax :

- A PHP script can be placed anywhere in the document.
- A PHP script starts with **<?php** and ends with **?>**
- The default file extension for PHP files is "**.php**".
- A PHP file normally contains HTML tags, and some PHP scripting code.
- In PHP, keywords (e.g. if, else, while, echo, etc.), classes, functions, and user-defined functions are not case-sensitive but all variable names are case-sensitive!

```
<html>
   <body>

   <h1>My first PHP
   page</h1>

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

   </body>
</html>
```

PHP Tag Syntax :

- **Full tags / Escaping To PHP / Canonical PHP Tags :**

  <? PHP PHP code ?>

- **Short tags / SGML Tags** (Standard Generalized Markup Language) **:**

  (This will only work by setting the *short_open_tag* setting in the *php.ini* file to 'on')

  <? PHP code ?>

- **Script tags :**

  <script language = "PHP">…</script>

- **ASP Style Tags** :

  (This will only work by setting the setting in the *php.ini*)

  <%…%>

Comments in PHP :

- **Single Line Comment**: As the name suggests these are single line or short relevant explanations that one can add to their code. To add this, we need to begin the line with (//) or (#).
- **Multi-line or Multiple line Comment**: These are used to accommodate multiple lines with a single tag and can be extended to many lines as required by the user. To add this, we need to begin and end the line with (/*...*/)

Variables in PHP :

- Variables in a program are used to store some values or data that can be used later in a program.
- The variables are also like containers that store character values, numeric values, memory addresses, and strings. PHP has its own way of declaring and storing variables.

Rules :

- Any variables declared in PHP must begin with a dollar sign ($), followed by the variable name.
- A variable can have long descriptive names (like $factorial, $even_nos) or short names (like $n or $f or $x)
- A variable name can only contain alphanumeric characters and underscores (i.e., 'a-z', 'A-Z', '0-9, and '_') in their name. Even it cannot start with a number.
- A constant is used as a variable for a simple value that cannot be changed. It is also case-sensitive.
- Assignment of variables is done with the assignment operator, "equal to (=)". The variable names are on the left of equal and the expression or values are to the right of the assignment operator '='.
- One must keep in mind that variable names in PHP names must start with a letter or underscore and no numbers.
- PHP is a loosely typed language, and we do not require to declare the data types of variables, rather PHP assumes it automatically by analyzing the values. The same happens while conversion. No variables are declared before they are used. It automatically converts types from one type to another whenever required.
- PHP variables are case-sensitive, i.e., $sum and $SUM are treated differently.

Data types used by PHP to declare or construct variables:

- Integers
- Doubles
- NULL
- Strings
- Booleans
- Arrays
- Objects
- Resources

PHP: loosely typed language :

- PHP automatically associates a data type to the variable, depending on its value. Since the data types are not set in a strict sense, you can do things like adding a string to an integer without causing an error.
- In PHP 7, type declarations were added. This gives an option to specify the data type expected when declaring a function, and by enabling the strict requirement, it will throw a "Fatal Error" on a type mismatch.
- You will learn more about strict and non-strict requirements, and data type declarations in the PHP Functions chapter.

Example :

```
<?php
   $a=10;
   $b="abc";
   $_x='a';
   $f=12.1544544;
   echo $a,$b,$_x,$f;
   echo "hello";
?>
```

Variable Scopes :

- Scope of a variable is defined as its extent in a program within which it can be accessed, i.e. the scope of a variable is the portion of the program within which it is visible or can be accessed.

  Depending on the scopes, PHP has three variable scopes:

  1)Local Variable

  2)Global Variable

  3)Static Variable

Local variables:

- The variables declared within a function are called local variables to that function and have their scope only in that particular function.
- In simple words, it cannot be accessed outside that function.
- Any declaration of a variable outside the function with the same name as that of the one within the function is a completely different variable.

EXAMPLE :

```
<?php
$num = 60;
    function local_var()
    {
        // This $num is local to this function
        // the variable $num outside this function
        // is a completely different variable
        $num = 50;
        echo "local num = $num <br>";
    }
local_var();
// $num outside function local_var() is a completely different Variable than that of
// inside local_var()
echo "Variable num outside local_var() is $num <br>";
?>
```

OUTPUT :

local num = 50

Variable num outside local_var() is 60

Global variables:

- The variables declared outside a function are called global variables.
- These variables can be accessed directly outside a function.
- To get access within a function we need to use the “global” keyword before the variable to refer to the global variable.

EXAMPLE :

```
<?php
$num = 20;
// function to demonstrate use of global variable
function global_var()
{
    // we have to use global keyword before
    // the variable $num to access within
    // the function
    global $num;
    echo "Variable num inside function : $num <br>";
}
global_var();
echo "Variable num outside function : $num <br>";
?>
```

OUTPUT :

Variable num inside function : 20

Variable num outside function : 20

Static variable :

- It is the characteristic of PHP to delete the variable, once it completes its execution and the memory is freed.
- But sometimes we need to store the variables even after the completion of function execution.
- To do this we use the static keywords and the variables are then called static variables.
- PHP associates a data type depending on the value for the variable.

EXAMPLE :

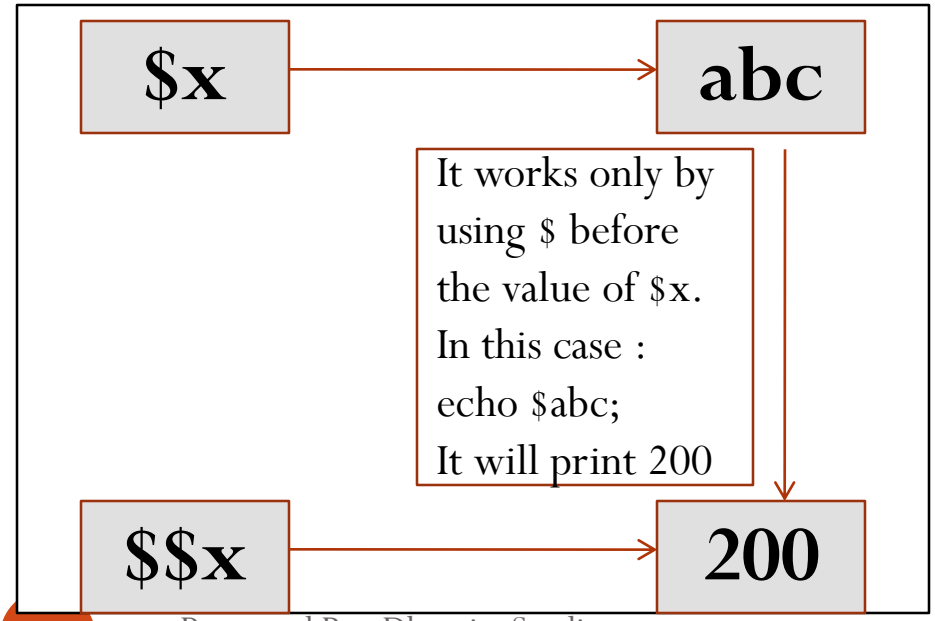
```
<?php
// function to demonstrate static
   variables
function static_var()
{
   static $num = 5;
   // static variable
   $sum = 2;
   $sum++;
   $num++;

   echo $num, "<br>";
   echo $sum, "<br>";
}
static_var(); // first function call
static_var(); // second function
   call
?>
```

OUTPUT :

```
6
3
7
3
```

PHP $ and $$ Variables :

- The **$x** (single dollar) is a normal variable with the name x that stores any value like string, integer, float, etc.
- The **$$x** (double dollar) is a reference variable that stores the value of the $variable inside it.



- **Example :**

```
<?php
    $x = "abc";
    $$x = 200;
    echo $x."<br/>";
    echo $$x."<br/>";
    echo $abc;
?>
```

- **Output :**

```
abc
200
200
```

PHP echo statement :

- It is a language construct and never behaves like a function, hence no parenthesis is required. But the developer can use parenthesis if they want.
- The end of the *echo* statement is identified by the semi-colon (';').
- It output one or more strings. We can use '*echo*' to output strings, numbers, variables, values, and results of expressions. Below is some usage of echo statements in PHP:
- **Displaying Strings**: We can simply use the keyword *echo* followed by the string to be displayed within quotes. The below example shows how to display strings with PHP.
- The **(.)** operator in the below code can be used to concatenate two strings in PHP and the "
" is used for a new line and is also known as line-break.

EXAMPLE :

```
<?php
// Defining the variables
$text = "Hello, World!";
$num1 = 10;
$num2 = 20;
// Echoing the variables
echo $text."<br>";
echo $num1."+".$num2."=";
echo $num1 + $num2;
?>
```

OUTPUT :

```
Hello, World!
10+20=30
```

PHP print statement :

- The PHP *print* statement is similar to the *echo* statement and can be used alternative to *echo* many times. It is also a language construct, and so we may not use parenthesis i.e *print* or *print()*.
- The main difference between the *print* and *echo* statement is that *echo* does not behave like a function whereas *print* behaves like a function.
  The *print* statement can have only one argument at a time and thus can print a single string. Also, the *print* statement always returns a value of 1. Like an *echo*, the *print* statement can also be used to print strings and variables. Below are some examples of using *print* statements in PHP:
- **Displaying String of Text**: We can display strings with the *print* statement in the same way we did with *echo* statements. The only difference is we cannot display multiple strings separated by comma(,) with a single print statement. The below example shows how to display strings with the help of a PHP *print* statement.

EXAMPLE :

```
<?php
// Defining the variables
$text = "Hello, World!";
$num1 = 10;
$num2 = 20;
// Echoing the variables
print $text."<br>";
print $num1."+".$num2."=";
print $num1 + $num2;
?>
```

OUTPUT :

Hello, World!

10+20=30

Difference between echo and print statements in PHP:

echo statement	print statement
echo accepts a list of arguments (multiple arguments can be passed), separated by commas.	print accepts only one argument at a time.
It returns no value or returns void.	It returns the value 1.
It displays the outputs of one or more strings separated by commas.	The print outputs only the strings.
It is comparatively faster than the print statement.	It is slower than an echo statement.

Operators of PHP :

- **What is Operator?** Simple answer can be given using expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator. PHP language supports following type of operators.
- **Types of Operators :**
  - Arithmetic operators
  - Assignment operators
  - Comparison operators
  - Increment/Decrement operators
  - Logical operators
  - String operators
  - Conditional assignment operators
  - Bitwise Operators

PHP Arithmetic Operators :

- The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc.

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

PHP Assignment Operators :

- The PHP assignment operators are used with numeric values to write a value to a variable.
- The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

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

PHP Comparison Operators :

The PHP comparison operators are used to compare two values (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
<=>	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.

PHP Increment / Decrement Operators:

- The PHP increment operators are used to increment a variable's value.
- The PHP decrement operators are used to decrement a variable's value.

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

PHP Logical Operators :

- The PHP logical operators are used to combine conditional statements.

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

PHP String Operators :

- PHP has two operators that are specially designed for strings.

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

PHP Conditional Assignment Operators:

- The PHP conditional assignment operators are used to set a value depending on conditions:

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
??	Null coalescing	$x = expr1 ?? expr2	Returns the value of $x. The value of $x is expr1 if expr1 exists, and is not NULL. If expr1 does not exist, or is NULL, the value of $x is expr2. Introduced in PHP 7

Decision Making/ Conditional Structure /Flow Control Statements :

- PHP allows us to perform actions based on some type of conditions that may be logical or comparative. Based on the result of these conditions i.e., either TRUE or FALSE, an action would be performed as asked by the user. It's just like a two- way path. If you want something then go this way or else turn that way. To use this feature, PHP provides us with four conditional statements:
- **if** statement
- **if…else** statement
- **if…elseif…else** statement
- **switch** statement

if Statement :

- This statement allows us to set a condition. On being TRUE, the following block of code enclosed within the if clause will be executed.
- **Syntax** :

  if (condition){ // if TRUE then execute this code }

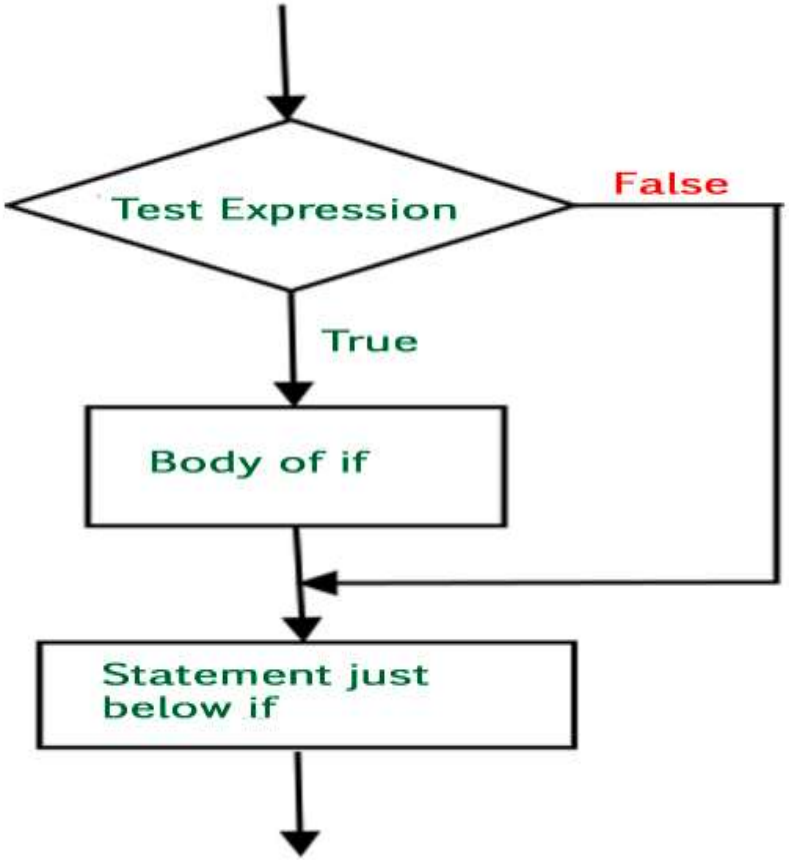
- **Example :**

```
<?php
$x = 12;

if ($x > 0) {
    echo "The number is positive";
}
?>
```

- **Output :**

The number is positive

Flowchart :


Test Expression
False
True
Body of if
Statement just
below if

if...else Statement :

- We understood that if a condition will hold i.e., TRUE, then the block of code within if will be executed. But what if the condition is not TRUE and we want to perform an action? This is where else comes into play. If a condition is TRUE then if block gets executed, otherwise else block gets executed.
- **Syntax**:

```
if (condition)
{
        // if TRUE then execute this code
}
else
{
        // if FALSE then execute this code
}
```

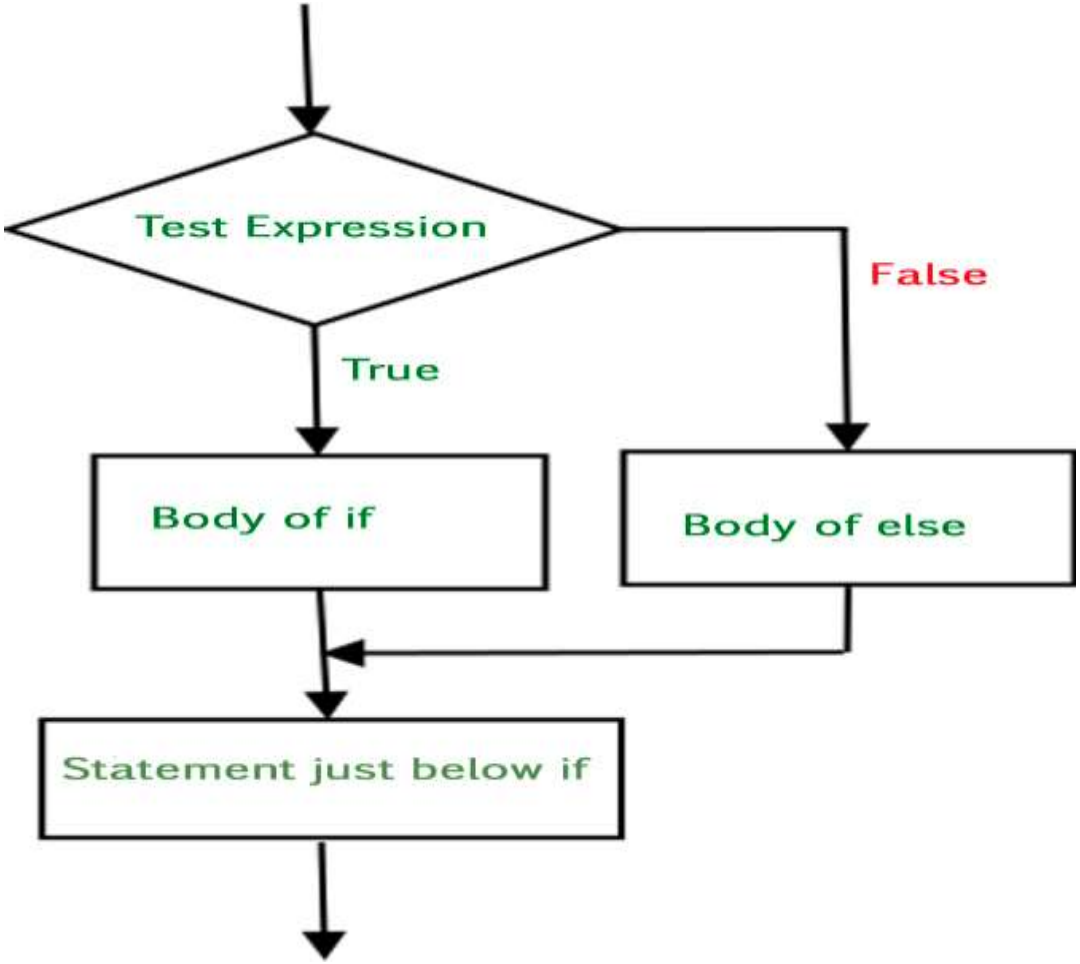
- **Example :**

```
<?php
$x = -12;
if ($x > 0) {
   echo "The number is positive";
}
else{
   echo "The number is negative";
}
?>
```

- **Output :**

The number is negative

Flowchart :


Test Expression
False
True
Body of if
Body of else
Statement just below if

if...elseif...else Statement :

- This allows us to use multiple if…else statements. We use this when there are multiple conditions of TRUE cases.

Syntax:

```
if (condition) {
 // if TRUE then execute this code }
 elseif {
 // if TRUE then execute this code }
 elseif {
 // if TRUE then execute this code }
 else {
 // if FALSE then execute this code }
```

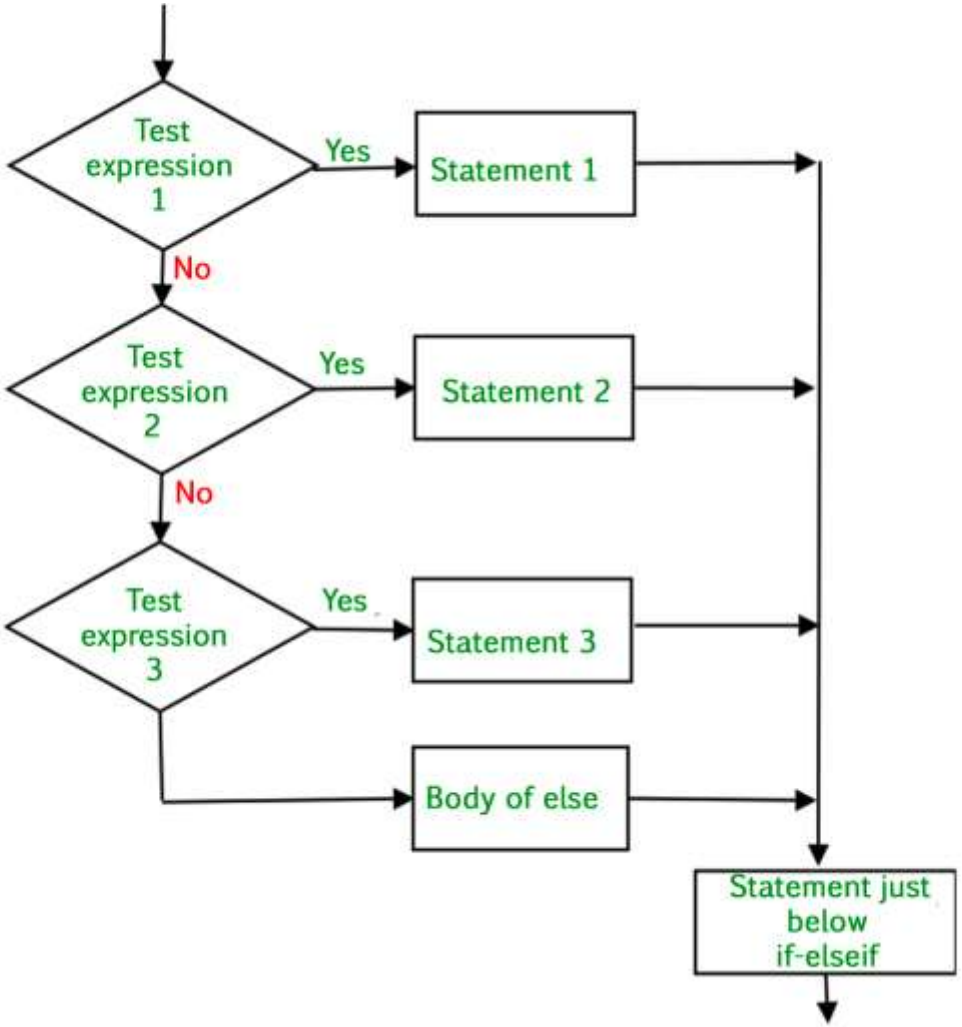
- **Example :**

```
<?php
$x = "August";
if ($x == "January") {
   echo "Happy Republic Day";
}
elseif ($x == "August") {
   echo "Happy Independence Day!!!";
}
else{
   echo "Nothing to show";
}
?>
```

- **Output :**

Happy Independence Day!!!

Flowchart :


Test expression 1
Yes
Statement 1
No
Test expression 2
Yes
Statement 2
No
Test expression 3
Yes
Statement 3
Body of else
Statement just below if-elseif

switch Statement :

- The “switch” performs in various cases i.e., it has various cases to which it matches the condition and appropriately executes a particular case block. It first evaluates an expression and then compares with the values of each case. If a case matches then the same case is executed. To use switch, we need to get familiar with two different keywords namely, **break** and **default**. The **break** statement is used to stop the automatic control flow into the next cases and exit from the switch case.
- The **default** statement contains the code that would execute if none of the cases match.

Syntax :

```
switch(n) {
 case statement1:
              code to be executed if n==statement1;
              break;
 case statement2:
              code to be executed if n==statement2;
              break;
 case statement3:
              code to be executed if n==statement3;
             break;
 ……
 default:
              code to be executed if n != any case;
```

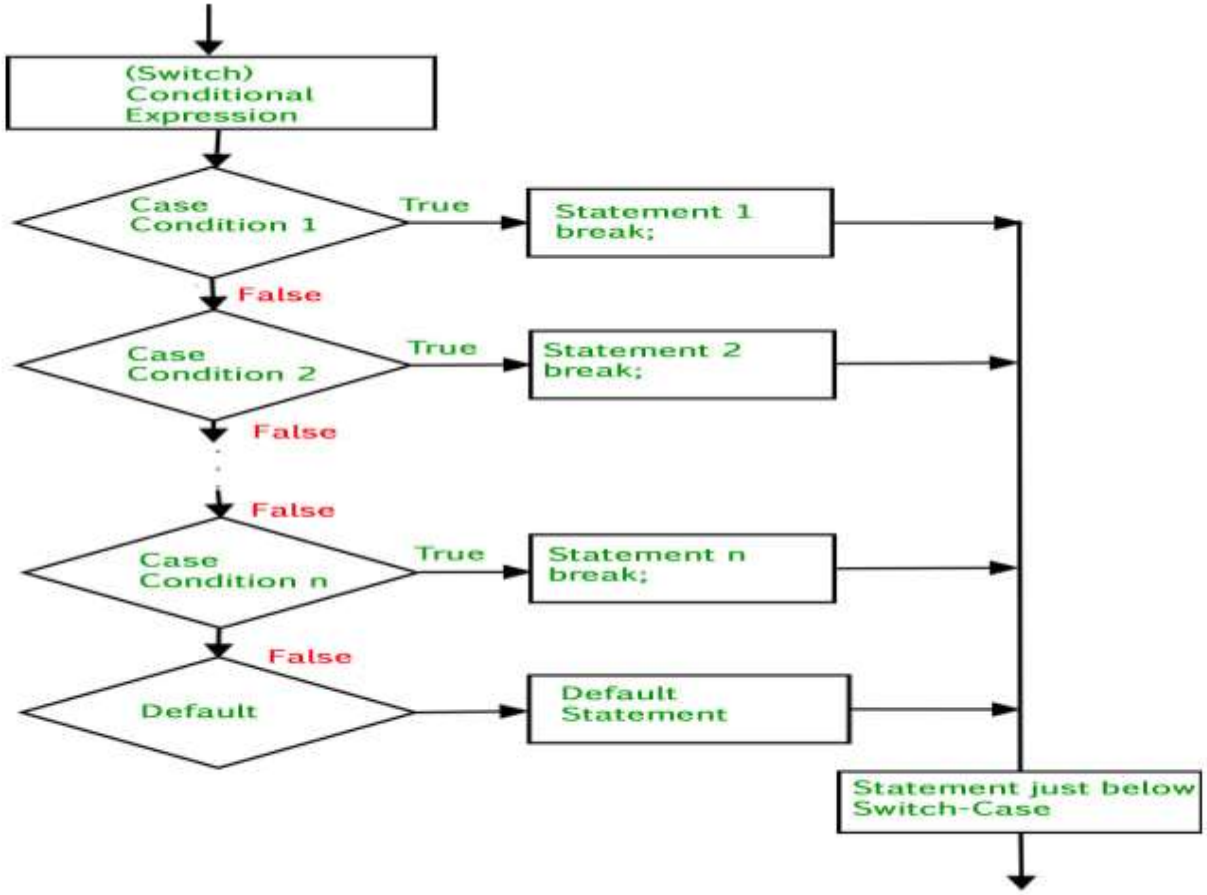
- **Example :**

```
<?php
$n = “A";
switch($n) {
    case “B":
            echo "Its B";
            break;
    case “C":
            echo "Its C";
            break;
    case “A":
            echo "Its A";
            break;
    default:
            echo "Doesn't exist";
}
?>
```

Output :

Its A

Flowchart :


(Switch)
Conditional
Expression
Case
Condition 1
True
Statement 1
break;
False
Case
Condition 2
True
Statement 2
break;
False
False
Case
Condition n
True
Statement n
break;
False
Default
Default
Statement
Statement just below
Switch-Case

Ternary Operators / Conditional Statement :

- In addition to all this conditional statements, PHP provides a shorthand way of writing if…else, called Ternary Operators. The statement uses a question mark (?) and a colon (:) and takes three operands: a condition to check, a result for TRUE and a result for FALSE.

  Syntax:

- (condition) ? if TRUE execute this : otherwise execute this;

Example :

```
<?php
$x = -12;
if ($x > 0) {
    echo "The number is positive <br>";
}
else {
    echo "The number is negative <br>";
}
// This whole lot can be written in a
// single line using ternary operator
echo ($x > 0) ? 'The number is positive' :'The number is negative';
?>
```

Output :

The number is negative

The number is negative

Looping Structure in PHP :

- Loops in PHP are used to execute the same block of code a specified number of times. PHP supports following four loop types.
- **for** – loops through a block of code a specified number of times.
- **while** – loops through a block of code if and as long as a specified condition is true.
- **do...while** – loops through a block of code once, and then repeats the loop as long as a special condition is true.
- **foreach** – loops through a block of code for each element in an array.

The for loop statement :

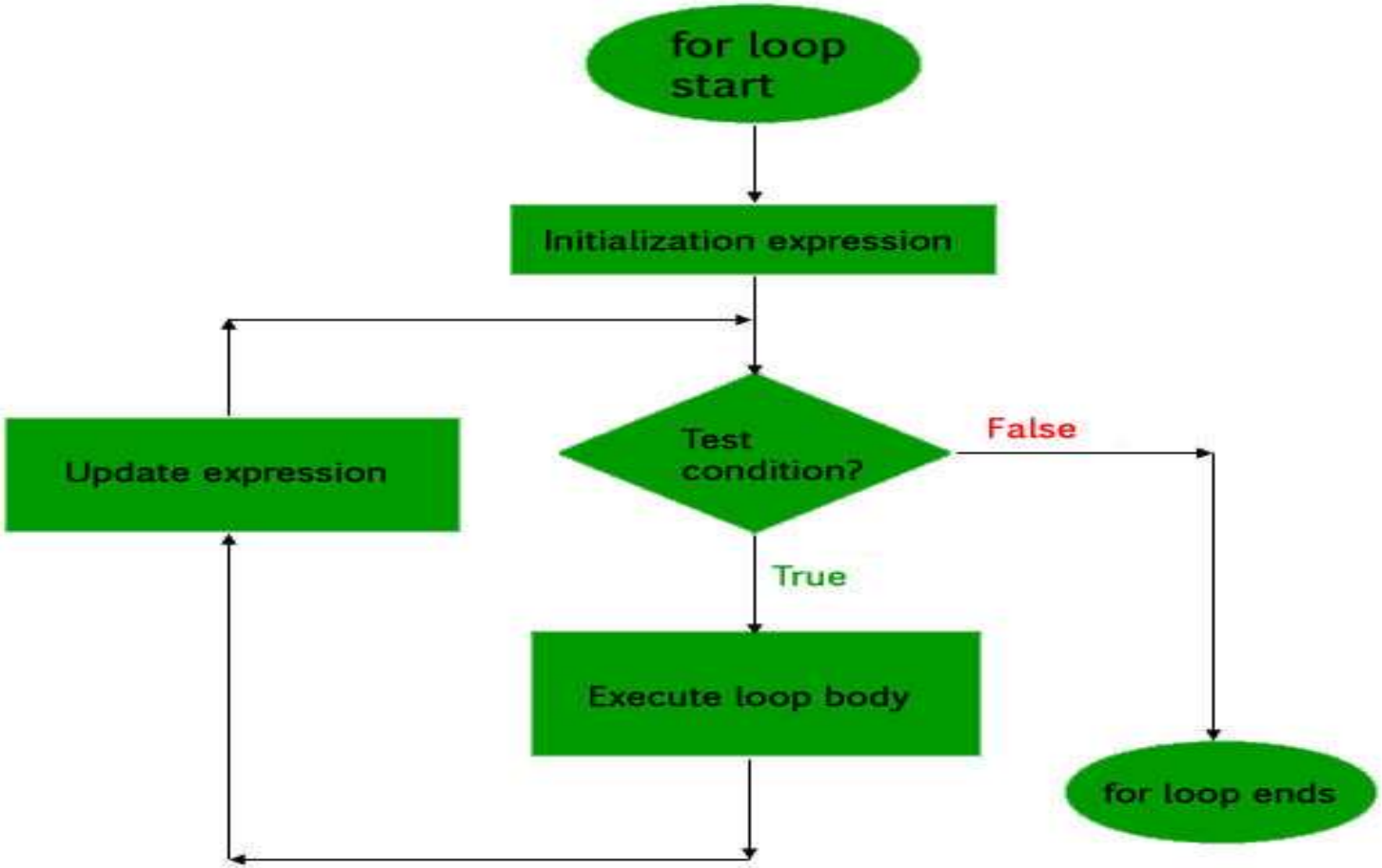
- The for statement is used when you know how many times you want to execute a statement or a block of statements.
- **Syntax :**

  for (initialization expression; test condition; update expression){

  code to be executed;

  }
- In for loop, a loop variable is used to control the loop. First initialize this loop variable to some value, then check whether this variable is less than or greater than counter value. If statement is true, then loop body is executed and loop variable gets updated . Steps are repeated till exit condition comes.
- **Initialization Expression**: In this expression we have to initialize the loop counter to some value. for example: $num = 1;
- **Test Expression**: In this expression we have to test the condition. If the condition evaluates to true then we will execute the body of loop and go to update expression otherwise we will exit from the for loop. For example: $num <= 10;
- **Update Expression**: After executing loop body this expression increments/decrements the loop variable by some value. for example: $num += 2;

Flowchart :


for loop start
Initialization expression
Test condition?
False
True
Update expression
Execute loop body
for loop ends

- **Example :**

```
<?php
// code to illustrate for loop
for ($num = 1; $num <= 10; $num += 2) {
    echo "$num <br>";
}
?>
```

- **Output :**

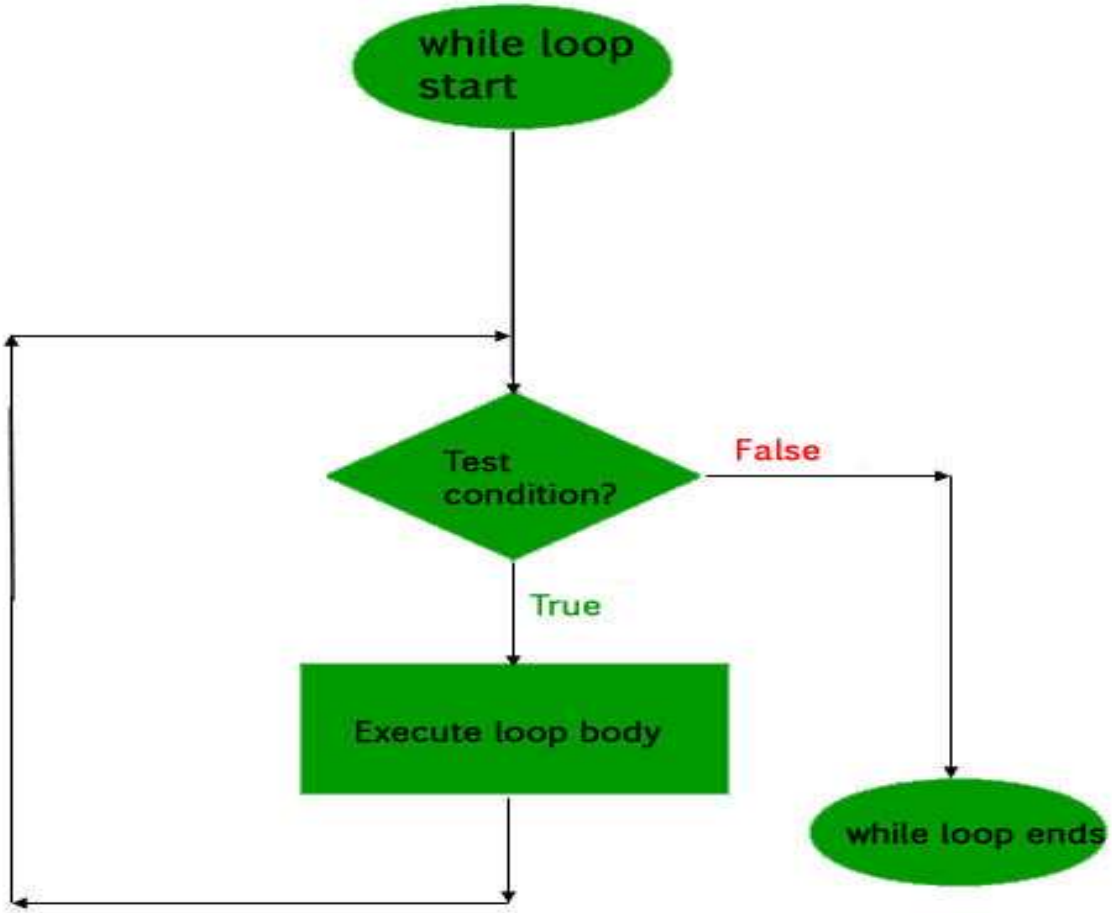
1
3
5
7
9

while loop :

- The while loop is also an entry control loop like for loops i.e., it first checks the condition at the start of the loop and if its true then it enters the loop and executes the block of statements, and goes on executing it as long as the condition holds true.
- **Syntax**:

```
while (if the condition is true)
{
        // code is executed
}
```

Flowchart :


while loop start
Test condition?
False
True
Execute loop body
while loop ends

Example :

```
<?php
// PHP code to illustrate while loops
$num = 2;
while ($num < 12) {
    $num += 2;
    echo $num, "<br>";
}
?>
```

Output :

4

6

8

10

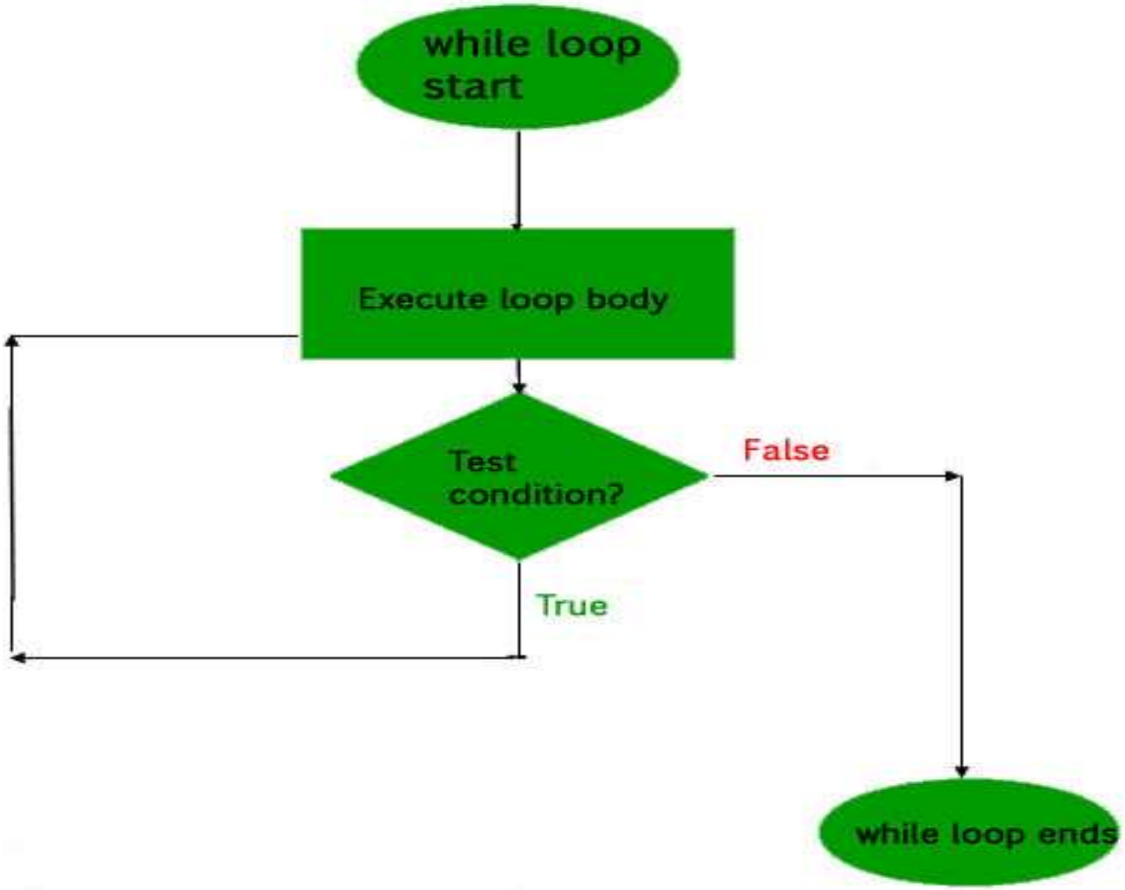
12

do-while loop :

- This is an exit control loop which means that it first enters the loop, executes the statements, and then checks the condition. Therefore, a statement is executed at least once on using the do…while loop. After executing once, the program is executed as long as the condition holds true.
- **Syntax**:

```
do {
        //code is executed
} while (if condition is true);
```

Flowchart :


while loop start
Execute loop body
Test condition?
False
True
while loop ends

- **Example :**

```
<?php
// PHP code to illustrate do...while loops
$num = 2;
do {
    $num += 2;
    echo $num, "<br>";
} while ($num < 12);
?>
```

- **Output :**

4
6
8
10
12

foreach loop :

- This loop is used to iterate over arrays. For every counter of loop, an array element is assigned and the next counter is shifted to the next element.

- **Syntax**:

```
foreach (array_element as value)
 {
        //code to be executed
 }
```

- **Example :**

```
<?php
    $arr = array (10, 20, 30, 40, 50, 60);
    foreach ($arr as $val) {
            echo "$val <br>";
    }
    $arr = array ("Ram", "Laxman", "Sita");
    foreach ($arr as $val) {
            echo "$val <br>";
    }
?>
```

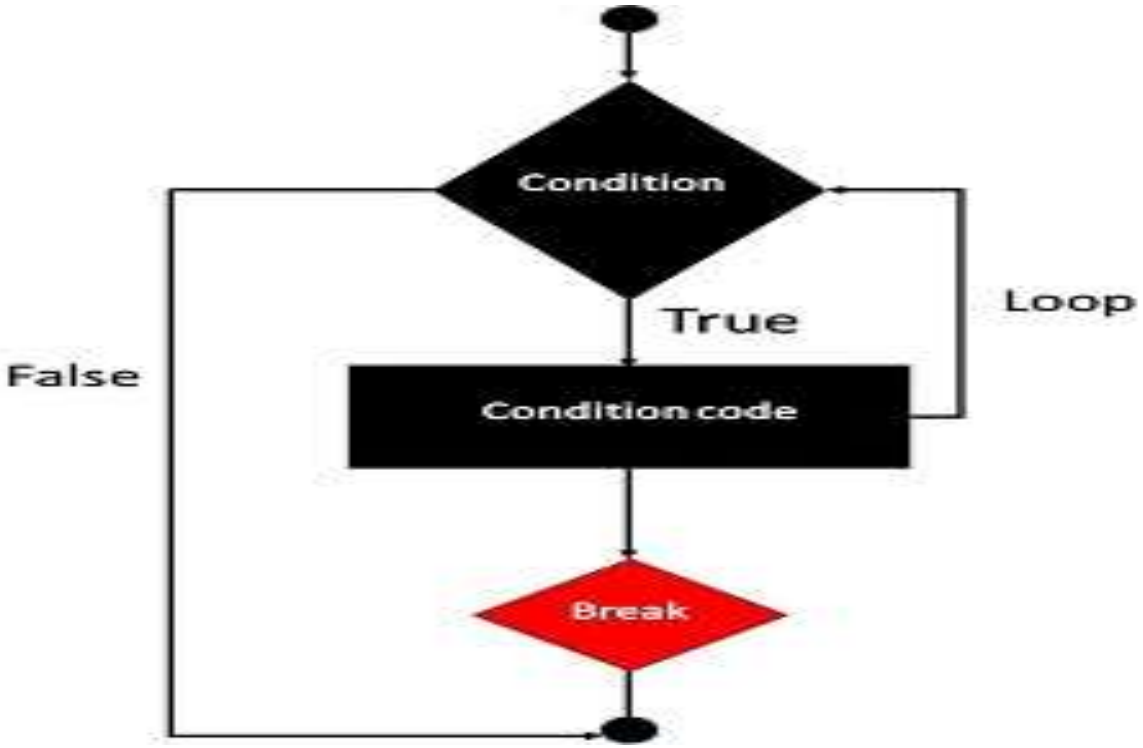
- **Output :**

10
20
30
40
50
60
Ram
Laxman
Sita

break Statement :

- The PHP **break** keyword is used to terminate the execution of a loop prematurely.
- The **break** statement is situated inside the statement block. It gives you full control and whenever you want to exit from the loop you can come out. After coming out of a loop immediate statement to the loop will be executed.

Flowchart :



- **Example :**

```
<?php
    $i = 0;
   while( $i < 10)
   {
            $i++;
                    if( $i == 3 )
                                break;
    }
 echo ("Loop stopped at i = $i" );
?>
```

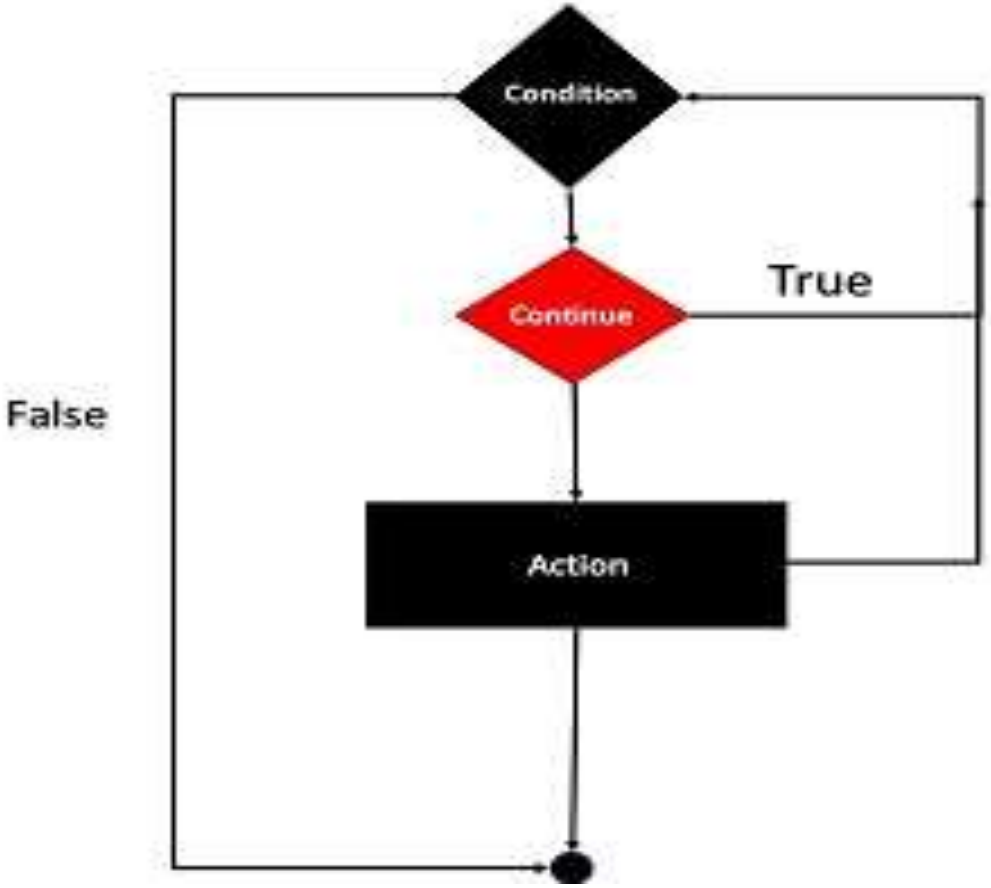
- **Output :**

Loop stopped at i = 3

continue statement :

- The PHP **continue** keyword is used to halt the current iteration of a loop but it does not terminate the loop.
- Just like the **break** statement the **continue** statement is situated inside the statement block containing the code that the loop executes, preceded by a conditional test. For the pass encountering **continue** statement, rest of the loop code is skipped and next pass starts.

Flowchart :



- **Example :**

```
<?php
$value=1;
   while($value<=5)
    {
          $value++;
          if( $value == 3 )
                continue;
          echo "Value is $value <br />";
   }
?>
```

- **Output :**

2
4
5
6

Difference between break and continue :

Break	Continue
The break statement is used to jump out of a loop.	The continue statement is used to skip an iteration from a loop
Its syntax is -: **break;**	Its syntax is -: **continue;**
The break is a keyword present in the language	continue is a keyword present in the language
It can be used with loops ex-: for loop, while loop.	It can be used with loops ex-: for loop, while loop.
The break statement is also used in switch statements	We can use **continue** with a switch statement to skip a case.

Array :

- An array is a special variable that can hold many values under a single name, and you can access the values by referring to an index number or name.
- The arrays are helpful to create a list of elements of similar types, which can be accessed using their index or key.
- Suppose we want to store five names and print them accordingly. This can be easily done by the use of five different string variables. But if instead of five, the number rises to a hundred, then it would be really difficult for the user or developer to create so many different variables. Here array comes into play and helps us to store every element within a single variable and also allows easy access using an index or a key.
- An array is created using an **array()** function in PHP.

➢ There are basically three types of arrays in PHP:

- **Indexed or Numeric Arrays:** An array with a numeric index where values are stored linearly.
- **Associative Arrays:** An array with a string index where instead of linear storage, each value can be assigned a specific key.
- **Multidimensional Arrays:** An array which contains single or multiple array within it and can be accessed via multiple indices.

1) Indexed or Numeric Arrays :

- These type of arrays can be used to store any type of elements, but an index is always a number. By default, the **index starts at zero**. These arrays can be created in two different ways as shown in the following example:

```
<?php
$name_one = array("Zack", "Anthony", "Ram");   // One way to create an indexed array
echo "Accessing the 1st array elements : <br> ";          // Accessing the elements directly
echo $name_one[2], " <br> ";
echo $name_one[0], "<br> ";
$name_two[0] = "Zack";                                    // Second way to create an indexed array
$name_two[1] = "Anthony";
$name_two[2] = "RAM";                                     // Accessing the elements directly
echo "Accessing the 2nd array elements :<br>";
foreach( $name_two as $value )
{
        echo "Value is $value <br>";
}
?>
```

OUTPUT :

Accessing the 1st array elements :

Ram

Zack

Accessing the 2nd array elements :

Zack

Anthony

Ram

2) Associative Arrays :

- These types of arrays are similar to the indexed arrays but instead of linear storage, every value can be assigned with a user-defined key of string type.

```
<?php                    // One way to create an associative array
$name_one = array("a"=>"Zara", "b"=>"Any“,"c"=>"Rani");
echo $name_one['a'],"<br>";
echo $name_one['b'];
// Second way to create an associative array
$name_two["zack"] = "abc";
$name_two["anthony"] = "xyz";
$name_two["ram"] = "pqr";
// Accessing the elements directly
echo "Accessing the elements directly:<br>";
foreach($name_two as $two)
{
    echo "$two";
}
?>
```

OUTPUT :

Accessing the 1st array elements :

Zara

Any

Accessing the 2nd array elements :

abc

xyz

pqr

Multidimensional Arrays :

- Multi-dimensional arrays are such arrays that store another array at each index instead of a single element.
- In other words, we can define multi-dimensional arrays as an array of arrays. As the name suggests, every element in this array can be an array and they can also hold other sub-arrays within.
- Arrays or sub-arrays in multidimensional arrays can be accessed using multiple dimensions.
- Example:

```php
<?php
// Defining a multidimensional array
$favorites = array(
    array(
            "name" => "Dave Punk",
            "mob" => "5689741523",
            "email" => "davepunk@gmail.com",
    ),
    array(
            "name" => "Monty Smith",
            "mob" => "2584369721",
            "email" => "montysmith@gmail.com",
    ),
    array(
            "name" => "John Flinch",
            "mob" => "9875147536",
            "email" => "johnflinch@gmail.com",
    )
);
// Accessing elements
echo "Dave Punk email-id is: " . $favorites[0]["email"], "<br>";
echo "John Flinch mobile number is: " . $favorites[2]["mob"];
?>
```

OUTPUT :

Dave Punk email-id is:
davepunk@gmail.com

John Flinch mobile number is:
9875147536

- We can access all data like this :

```php
echo "<pre>";
        print_r($favorites);
echo "</pre>";
```

```
<?php
// Defining a multidimensional array
$favorites = array(
            "Dave Punk" => array(
            "mob" => "2584369721",
            "email" =>
    "montysmith@gmail.com",
    ),
    "John Flinch" => array(
            "mob" => "9875147536",
            "email" => "johnflinch@gmail.com",
    )
);
echo "<pre>";
    print_r($favorites);
echo "</pre>";
?>
```

- **OUTPUT :**

```
Array ( [Dave Punk] => Array ( [mob] =>
5689741523 [email] => davepunk@gmail.com
) [John Flinch] => Array ( [mob] =>
9875147536 [email] =>
johnflinch@gmail.com ) )
```

- We can also access all data using nested loop like this :

```
// Using for and foreach in nested form
 $keys = array_keys($favorites);
 for($i = 0; $i < count($favorites); $i++) {
 echo $keys[$i] . "<br>";
 foreach($favorites[$keys[$i]] as $key =>
$value) {
        echo $key . " : " . $value . "<br>";
 }
 echo "<br>";
}
```

User Define Functions :

- PHP has a large number of built-in functions such as mathematical, string, date, array functions etc.
- It is also possible to define a function as per specific requirement. Such function is called user defined function.
- A function is a reusable block of statements that performs a specific task. This block is defined with function keyword and is given a name that starts with an alphabet or underscore.
- This function may be called from anywhere within the program any number of times.

- **Syntax**

```
//define a function
function myfunction($arg1,
   $arg2, … $argn)
{
                statement1;  statem
   ent2;  ..  ..  return $val;
}
//call function
myfunction($arg1, $arg2, …
   $argn);
```

- **EXAMPLE :**

```
<?php
 //function definition
function sayHello(){
            echo "Hello World!";
}
//function call sayHello();
?>
```

- **OUTPUT :**

Hello World!

Types of UDF :

1. Argument Function
2. Default Argument
3. Variable Function
4. Return Function

1) function with arguments :

- We can pass the information in PHP function through arguments which is separated by comma.
- **Syntax**

```
//define a function
function myfunction($arg1, $arg2, … $argn)
{
        statement1;  statement2;  ..  ..  return $val;
}
//call function
myfunction($arg1, $arg2, … $argn);
```

- **Example :**

```
<?php
function sayHello($name){
        echo "Hello $name<br/>";
}
sayHello(“Bca");
sayHello(“Sem2");
?>
```

- **Output :**

Hello Bca

Hello Sem2

2) Function with default argument value :

- While defining a function , a default value of argument may be assigned. If value is not assigned to such argument while calling the function, this default will be used for processing inside function
- **Syntax :**

```
function myfunction($arg1=[value], $arg2=[value], … $argn)
{
        statement1;  statement2;  ..  ..  return $val;
}
//call function
myfunction($arg1, $arg2, … $argn);
```

- **Example :**

```
<?php
function welcome($user="Guest")
{
        echo "Hello $user
  <br>";
 }
 //overrides default
 welcome("admin");
 //uses default
 welcome();
?>
```

- **OUTPUT :**

Hello admin

Hello Guest

3) Function with variable number of arguments (call by value) :

- It is possible to define a function with ability to receive variable number of arguments. The name of formal argument in function definition is prefixed by **...** token.
- **Syntax :**

```
//define a function
function myfunction($arg1, $arg2, … $argn)
{
        statement1;  statement2;  ..  ..  return $val;
}
//call function
myfunction($var1, $var2, … $var);
```

- **Example :**

```
<?php
function  Var_fun($str2)
{
   echo $str2;
}
$str = 'Hello';
Var_fun($str);
?>
```

- **Output :**

Hello

4) Function return :

- It just echoed a couple of sentences based on the input. It is entirely up to us to return anything inside any function we define in PHP.
- When you want to return something, you need to use the **return** statement.
- A function can return any type of value in PHP.

- **Example :**

```
<?php
        function sum($n1,$n2){
        return $n1+$n2;
}
echo " Sum is: ".sum(3,5);
?>
```

- **Output :**

Sum is : 8

Other Things About Function :

- Call By Reference
- function within another function
- Recursive function

PHP Call By Reference :

- Value passed to the function doesn't modify the actual value by default (call by value). But we can do so by passing value as a reference.
- By default, value passed to the function is call by value. To pass value as a reference, you need to use ampersand (&) symbol before the argument name.

- **Example :**

```
<?php
function adder(&$str2)
{
   $str2 .= 'Call By Reference';
}
$str = 'Hello ';
echo $str."<br>";
adder($str);
echo  " After function Calling:
          <br>";
echo $str;
?>
```

- **Output :**

Hello

After function Calling :

Call by Reference

function within another function :

- A function may be defined inside another function's body block.
- However, inner function can not be called before outer function has been invoked.

- **Example :**

```
<?php
 function hello()
{
   echo "Hello";

   function welcome(){
      echo “<br>Welcome to
   BCA";
   }
}

 //welcome();
hello();
welcome();
?>
```

- **Output :**

Hello

Welcome to BCA

Now, Remove the comment to call welcome() before hello(). Following error message halts the program −

Fatal error: Uncaught Error: Call to undefined function welcome() in C:\xampp\htdocs\hello.php:77 Stack trace: #0 {main} thrown in **C:\xampp\htdocs\hello.php** on line **12**

Recursive Function / Function Recursion :

- A function that calls itself is called recursive function.
- Calling itself unconditionally creates infinite loop and results in out of memory error because of stack full.

- **Example :**

```
<?php
function rec($n){
  if ($n >= 1) {
    echo $n,"<br>";
    rec($n-1);
  }
}
rec(5);
?>
```

5
4
3
2
1

- **Output :**

- **Variable Length Argument Function**
  1. func_num_args
  2. func_get_arg
  3. func_get_args

1) func_num_args :

- The func_num_args() function can return the number of arguments passed to a function.
- **Syntax :**

  int func_num_args(void)
- The func_num_args() function can return the number of arguments passed into current user-defined function. This function can generate a warning if called from outside of a user-defined function.

- **Example :**

```
<?php
function demo()
{
   echo "Number of arguments: ", func_num_args() ,"<br>";
}
demo(1, 2,"abc", 3);
demo(1, 2,"abc", 3,'x');
?>
```

- **Output :**

```
Number of arguments: 4
Number of arguments: 5
```

2) func_get_arg() Function :

- The **func_get_arg()** function is an inbuilt function in PHP which is used to get a mentioned value from the argument passed as the parameters.
- **Syntax:**

  mixed func_get_arg(*int* $arg)
- **Parameters:** This function accepts a single parameter as mentioned above and described below.
- $arg: This parameter holds the argument offset where the offset of the arguments in the parameter is counted by assuming the first argument to be 0.
- **Return Value:** This method returns the mentioned argument and returns FALSE if an error occurs.

- **Example :**

```
function geeks($a, $b, $c) {
    // Calling func_get_arg() function
    echo "Print second argument: ". func_get_arg(1) . "<br>";
}
// Function call
geeks('hello', 'php', 'geeks');
```

- **Output :**

Print second argument: php

When does any error occur?

The error occurs in two cases.

- If the value of argument offset is more than the actual value of arguments passed as the parameter of the function.
- If this function is not being called from within the user-defined function.
- **Note:** For getting more than one argument **func_get_args()** function can be used instead of **func_get_arg()** function.

3) func_get_args() :

- The func_get_args() function can return an array comprising a function's argument list.
- **Syntax :**

  array func_get_args(void)

- The func_get_args() function can return an array in which each element is a corresponding member of the current user-defined function's argument list. This function can generate a warning if called from outside of function definition.

- **Example :**

```
<?php
function foo1() {
    $numargs = func_num_args();
// return the parameters contained in this function
    echo "number of argumets:"
.$numargs . "<br>";

    $arr = func_get_args();
// return an array to $arr
    print_r ($arr);
// output all parameters of this array
    echo "<br>";
    for($i=0; $i< $numargs; $i++)
{
    echo $arr[$i],"<br>";
    }
}
foo1(1,'a',2,3,4,'bca');
?>
```

- **OUTPUT :**

```
number of argumets:6
Array ( [0] => 1 [1] => a [2] =>
2 [3] => 3 [4] => 4 [5] => bca )
1
a
2
3
4
bca
```

Built in Functions :

1. Variable Functions
2. String Function
3. Math Function
4. Date Function
5. Array Function
6. Miscellaneous Function
7. File handling Function

Assignment Questions :

1. Write down History of PHP.
2. Write down .ini file.
3. Write down .htaccess file.
4. what is web server?
5. write down PHP tag.
6. why PHP is Loosely typed language?
7. Write down all scope of variable with example.
8. Write down difference between $ and $$ variable.
9. Write down difference between echo , print and print_r.
10. Write down difference between break and continue.
11. What is switch statement? explain in detail.
12. What is foreach loop in php? explain with example.
13. Explain all types of array in php with examples.
14. What is UDF? write down types of UDF and explain default argument function.
15. Explain function call by reference.
16. explain recursion in detail.
17. Explain all variable length argument function in detail