CHAPTER 1

UNIT I

Expressions and Control Statements in PHP

Syllabus

1.1 Introduction

Q. Define PHP?	**(2 Marks)**

– PHP stands for **H**ypertext **P**reprocessor. PHP is a powerful and widely-used open source server-side scripting language to write dynamically generated web pages. PHP scripts are executed on the server and the result is sent to the browser as plain HTML.

– PHP can be integrated with the number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.

– PHP is an "HTML-embedded scripting language" primarily used for dynamic Web applications.

– PHP can be embedded within a normal HTML web pages. That means inside your HTML documents you'll have PHP statements

– PHP takes most of its syntax from C, Java, and Perl. It is an open source technology and runs on most operating systems and with most Web servers. PHP was written in the C programming language by Rasmus Lerdorf in 1994 for use in monitoring his online resume and related personal information. For this reason, PHP originally stood for "Personal Home Page".

– PHP files have a file extension of ".php", ".php3", or ".phtml"

1.1.1 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.

1.1.2 Advantages of PHP

Q.	Describe advantages of PHP	**(4 Marks)**
Q.	Why PHP is known as scripting language?	**(2 Marks)**

- **Open Source :** PHP is open source and free of cost, which helps developers to install it quickly and readily available for use. There are a lot of PHP frameworks and developer can choose any of the frameworks to work. All the features and tools will be provided to the developer for that framework easily. Open Source means you never need to rely on the manufacturer to release the next version if something doesn't work or pay for expensive upgrades.
- **Platform Independent:** PHP is mainly supported by all the operating systems like Windows, Unix, Linux etc. The PHP based developed web applications can be easily run on any platform. It can be integrated with other programming language and database easily and there is no requirement of re-development. It helps in saving a lot of effort and cost.
- **Simple and Easy :** PHP is simple and easy to learn and code. The command functions of PHP can easily learn and understood. The syntax is simple and flexible to use.
- **Database :** PHP is easily connected with the database and make the connection securely with databases. It has a built-in module that is used to connect to the database easily. Multiple databases can be integrated with PHP.
- **Fast :** PHP is known as the fastest Programming language as compared to another. PHP applications can be easily loaded over the slow Internet and data speed.
- **Support :** This advantage of PHP has great online support and community, which helps the new developers to help in writing the code and developing the web applications. Compatible with servers like IIS and APACHE.
- **Security :** PHP frameworks built-in feature and tools make it easier to protect the web applications from the outer attacks and security threats.
- **Maintenance :** PHP framework is mainly used to make the web application development easier and maintain the code automatically. Low development and maintenance cost with very high performance and reliability
- **Fast Performance :** Scripts written in PHP usually execute faster than those written in other scripting languages like ASP.NET or JSP.
- **Vast Community :** Since PHP is supported by the worldwide community, finding help or documentation for PHP online is extremely easy. PHP is so popular that it's quite easy for you to get support on PHP.
- **Scripting language :** A scripting language or script language is a programming language that supports scripts. If code of programming language can embed with other language or integrate with other language or script called scripting language. PHP is Scripting language because we can embed PHP code into HTML. PHP is server side language because PHP requires server to run a code. Code of PHP gets executed on server and result of execution is return to the browser, that's why PHP is called script language and server side language.

1.1.3 Syntax of PHP

Q.	What these tags specify in PHP : <?php and ?>	**(2 Marks)**

- A PHP script starts with the <?php and ends with the ?> tag.
- The PHP delimiter <?php and ?> in the following example simply tells the PHP engine to treat the enclosed code block as PHP code, rather than simple HTML.

– On servers with shorthand support enabled, you can start a scripting block with <? and end with ?>.

Syntax :

```
<?php
  echo 'Hello world';
?>
```

Example :

```
<html>
<body>
<?php echo "Hello World";
?>
</body>
</html>
```

Each code line in PHP must end with a semicolon. The semicolon is a separator and is used to distinguish one set of instructions from another.

There are two basic statements to output text with PHP: **echo** and **print**. In the example above we have used the echo statement to output the text "Hello World".

1.1.4 Embedding PHP within HTML

Q.	Can we run PHP from HTML file? If yes, how?	**(2 Marks)**
Q.	How can PHP and HTML interact?	**(2 Marks)**

PHP files are plain text files with .php extension. Inside a PHP file you can write HTML like you do in regular HTML pages as well as embed PHP codes for server side execution.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>A Simple PHP File</title>
</head>
<body>
<h1><?php echo "Hello, world!"; ?></h1>
</body>
</html>
```

The above example shows how you can embed PHP codes within HTML to create well-formed dynamic web pages. If you view the source code of the resulting web page in your browser, the only difference you will see is the PHP code <?php echo "Hello, world!"; ?> has been replaced with the output "Hello, world!".

What happened here is? when you run this code the PHP engine executed the instructions between the <?php … ?> tags and leave rest of the thing as it is. At the end the web server send the final output back to your browser which is completely in HTML.

Example :

```
<html>
<body class="page_bg">
Hello, today is <?php echo date('l, F jS, Y'); ?>.
</body>
</html>
```

Output :

```
Hello, today is Wednesday, November 6th, 2019.
```

1.1.5 PHP Comments

Q. How can we define comments in PHP? **(2 Marks)**

A comment is simply text that is ignored by the PHP engine. The purpose of comments is to make the code more readable. It may help other developer (or you in the future when you edit the source code) to understand what you were trying to do with the PHP.

PHP support single-line as well as multi-line comments. To write a single-line comment either start the line with either two slashes (//) or a hash symbol (#).

Example :

```
<?php
    // This is a single line comment
    # This is also a single line comment
  echo 'Hello World!';
?>
```

However to write multi-line comments, start the comment with a slash followed by an asterisk (/*) and end the comment with an asterisk followed by a slash (*/), like this :

```
<?php
/*
This is a multiple line comment block
that spans across more than
one line
*/
echo "Hello, world!";
```

```
?>
```

1.1.6 PHP echo and print

- In PHP, there are two ways to get output or print output: echo and print.
- They are both used to output data to the screen.

The PHP echo Statement :

- The echo statement can be used with or without parentheses : echo or echo().
- The echo statement can display anything that can be displayed to the browser, such as string, numbers, variables values, the results of expressions etc.

Example :

```
<?php
// Displaying strings
echo "Hello, Welcome to PHP Programming";
echo "<br/>";

//Displaying Strings as Multiple arguments
echo "Hello", " Welcome", " PHP";
echo "<br/>";

//Displaying variables
$s="Hello, PHP";
$x=10;
$y=20;
echo "$s <br/>";
echo $x."+".$y."=";
echo $x + $y;
?>
```

Output :

```
Hello, Welcome to PHP Programming
Hello Welcome PHP
Hello, PHP
10+20=30
```

The PHP print Statement :

The PHP **print** statement is similar to the echo statement and can be used alternative to echo at many times. It is also language construct and so we may not use parenthesis : print or print(). print statement can also be used to print strings and variables.

Example :

```
<?php
// Displaying strings
print "Hello, Welcome to PHP Programming";
print "<br/>";

//Displaying variables
$s="Hello, PHP";
$x=10;
$y=20;
print "$s <br/>";
print $x."+".$y."=";
print $x + $y;
?>
```

Output :

```
Hello, Welcome to PHP Programming
Hello, PHP
10+20=30
```

Difference between echo and print

Q. Give difference between echo() and print(). **(4 Marks)**

Echo	print
echo outputs one or more strings or arguments.	print output only one string or argument.
echo has no return value.	print has a return value of 1 so it can be used in expressions.
Echo is faster than print	Print is slower than echo.
Not written with parenthesis.	It can or cannot be written with parenthesis.
Syntax: echo($arg1, $arg2, ….)	Print ($arg)

The print_r() function in PHP

The print_r() function is used to print human-readable information about a variable.

– **Syntax :** print_r($variable, $isStore)

– **$variable** : This parameter specifies the variable to be printed and is a mandatory parameter.

– **$isStore** : This an option parameter. This parameter is of boolean type whose default value is FALSE and is used to store the output of the print_r() function in a variable rather than printing it. If this parameter is set to TRUE then the print_r() function will return the output which it is supposed to print.

– **Return Value** : If the $variable is an integer or a float or a string the function prints the value of the variable. If the variable is an array the function prints the array in a format which displays the keys as well as values, a similar notation is used for objects. If the parameter $isStore is set to TRUE then the print_r() function will return a string containing the information which it is supposed to print.

Example :

```
<?php
// string variable
$var1 = "Hello PHP";
// integer variable
$var2 = 101;
// array variable
$arr = array('0' => "Welcome", '1' => "to", '2' => "PHP");
// printing the variables
print_r($var1);
echo "<br/>";
print_r($var2);
echo "<br/>";
print_r($arr);

?>
```

Output :

```
Hello PHP
101
Array ( [0] => Welcome [1] => to [2] => PHP )
```

1.2 Variables

Q. How PHP variables are different from other language's variables? **(2 Marks)**

– Variables are used to store data, like string of text, numbers, etc. Variable values can change over the course of a script.

– In PHP, a variable does not need to be declared before adding a value to it. PHP automatically converts the variable to the correct data type, depending on its value.

- After declaring a variable it can be reused throughout the code.
- In PHP, a variable starts with the **$** sign, followed by the name of the variable :
- The assignment operator (**=**) used to assign value to a variable.
- In PHP variable can be declared as : **$var_name = value;**

Example :

```
<html>
<body>
<?php
// Declaration of variables
$txt = "Hello World!";
$number = 10;
// Display variables value
echo $txt;
echo "<br>";
echo $number;
?>
</body>
</html>
```

Output :

```
Hello World!
10
```

Rules for Declaring PHP Variables :

Q. Write down the rules to know about the variables in PHP? **(4 Marks)**

- 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 should not contain spaces. If a variable name is more than one word, it should be separated with an underscore ($first_name), or with capitalisation ($firstName).
- Variables used before they are assigned have default values.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters (A-Z, a-z) and underscores.
- Variable names are case-sensitive ($name and $NAME are two different variables)
- You able to use variable over and over again in your PHP script after declaring it.

- Variables can, but do not need, to be declared before assignment. PHP automatically converts the variable to the correct data type, depending on its value.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- PHP variables are Perl-like.

PHP is Loosely Typed Language :

Q. Why PHP is called loosely Typed Language?

- In PHP, a variable does not need to be declared before adding a value to it.

For example :

```
<?php
$str="Hello World!";
$x=16;
?>
```

- In above example, you do not have to tell PHP which data type the variable is.
- PHP automatically converts the variable to the correct data type, depending on its value.
- In a strongly typed programming language, you have to declare (define) the type and name of the variable before using it.
- In PHP, the variable is declared automatically when you use it.

1.2.1 Variable Scope

Q.	Describe scope of variables	**(4 Marks)**
Q.	How to define global variables in PHP?	**(4 Marks)**
Q.	Describe static, local and global variable. Explain it.	**(6 Marks)**

Scope of a variable is defined as its extent in program within which it can be accessed, i.e. the scope of a variable is the portion of the program with in which it is visible or can be accessed. There are three different variable scope:

1. **Local Variables :** The variables declared within a function are called local variables to that function and has its scope only in that particular function. Local variables cannot be accessed outside that function. Any declaration of a variable outside the function with same name as that of the one within the function is a complete different variable.

Example :

```
<?php
//the variable $num declare outside this function is different variable
$num = 20;

function local_var()
{
    $num = 10;
    echo "local num = $num<br/>";
}
local_var();

//echo "num Outside Local_var()=$num<br/>";
echo "Variable num outside local_var()=$num<br/>";
?>
```

Output :

```
local num = 10
Variable num outside local_var() = 20
```

2. **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
//declare global variable
$num = 20;

function local_var()
{
global $num;
    echo "Access global variable within function=$num<br/>";
}

local_var();

//echo "global variable Outside Local_var()=$num<br/>";
echo "globalvariablenum outside of local_var()=$num<br/>";
?>
```

Output :

```
Access global variable within function=20
globalvariablenum outside of local_var()=20
```

3. Static Variables

Q. Define static variable. Explain with example **(4 Marks)**

It is the characteristic of PHP to delete the variable, ones 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 static keyword and the variables are then called as static variables.

Example :

```
<?php
function static_var()
{
static $x=1;
$y=1;
$x++;
$y++;
echo "x=$x <br/>";
echo "y=$y <br/>";
}

static_var();
static_var();
static_var();
?>
```

Output :

```
x=2
y=2
x=3
y=2
x=4
y=2
```

Note : $x is regularly increments even after the first function call but $y doesn't. This is because $y is not static and it's memory is freed after execution of first function call.

1.2.2 PHP $ and $$ Variables

Q. State the difference between $ and $$ in PHP. **(2 Marks)**

- The $var (single dollar) is a normal variable with the name var that stores any value like string, integer, float, etc. Example : $x="Hello";
- The $$var (double dollar) is a reference variable that stores the value of the $variable inside it (variable of a variable). Example : $$x=100
- $x represents variables whereas $$x represents a variable with the content of $x

$$x = $($x) = $(“Hello”) = $Hello = PHP

– Data stored in $ is fixed while data stored in $$(double dollar sign) can be changed dynamically.

```
    $Hello = "Vidyalankar"
    $var = "Hello"
    echo $var
    echo $$var
Output : Hello
         Vidyalankar
```

Example :

```
<?php
$x="Hello";
$$x=”PHP”;
echo $x."<br/>";
echo $$x."<br/>";
echo $Hello;
?>
```

Output : $$x = $($x) = $(“Hello”) = $Hello = PHP

```
Hello
PHP
PHP
```

1.2.3 Data Types

Q.	Describe various data types available in PHP.	**(4 Marks)**

– Data types are used to hold different types of data or values.

– PHP supports 8 primitive data types that can be categorized in 3 types :

Scalar Type	Compound Type	Special Type
Integer	Array	Resource
Float	Object	NULL

String		
Boolean		

1. Scalar Type (Single value types) :

There are 4 scalar data types in PHP.

(i) Integer : This data type holds only numeric values. Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or decimal point. The range of integer values are -2,147,483,648 to +2,147,483,647. Integers can be defined indecimal(base 10), hexadecimal(base 16), octal(base 8) or binary(base 2) notation.The default base is decimal (base 10). The octal integers can be declared with leading 0 and the hexadecimal can be declared with leading 0x. The PHP **var_dump()** function returns the data type and value.

Decimal Values : Decimal values are represented by a sequence of digits without leading zeros. The sequence may begin with a plus(+) or minus (-) sign. It considers positive numbers, if no sign is assigned. Example: 50, -11, +33.

Octal Numbers : Octal numbers are represented by leading O and sequence of digits from 0 to 7. Octal numbers can be prefixed with a plus or minus. Example: O53, +O10

Hexadecimal Values: Hexadecimal values begins with Ox, followed by sequence of digits (0-9) or letters(A-F). Example: Ox10

Rules for declaring integer values :

– 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

(ii) Float : Floating point numbers represents numeric values with decimal points (real numbers) or a number in exponential form. The range of floating point values are 1.7E-308 to 1.7E+308. Example: 3.14, 0.25, -5.5, 2E-4, 1.2e2

(iii) String : A string is a sequence of characters. A string are declares using single quotes or double quotes. Example: "Hello PHP", 'Hello PHP'

(iv) Boolean : The Boolean data types represents two values, true(1) or false(0). Example: $a=True

Example :

```
<?php
$a=11;
$b=11.22;
$c="Hello PHP";
$d=True;
var_dump($a);
var_dump($b);
var_dump($c);
var_dump($d);
?>
```

Output :

```
int(11)
float(11.22)
string(9) "Hello PHP"
bool(true)
```

2. Compound Type :

There are 2 compound data types in PHP: Array and object.

(i) **Array:** An array stores multiple values in one single variable and each value is identify by position (zero is the first position). The array is a collection of heterogeneous (dissimilar) data types. PHP is a loosely typed language that's why we can store any type of values in arrays.

Syntax : Variable_name = array (element1, element2, element3, element4......)

Example 1:

```
<?php
$MyArray = array( 10, 20 , 30);
echo "First Element: $MyArray[0] <br/>";
echo "Second Element: $MyArray[1] <br/>";
echo "Third Element: $MyArray[2] <br/>";
?>
```

Output :

```
First Element: 10
Second Element: 20
Third Element: 30
```

Example 2 :

```
<?php
  $a = [10, 20, 30, 40];
print_r($a);
?>
```

Output :

```
Array ( [0] => 10 [1] => 20 [2] => 30 [3] => 40 )
```

(ii) Object :

Objects are defined as instances of user defined classes that can hold both values and functions. First we declare class of object using keyword class. A class is a structure that contain properties(variables) and methods(functions).

Example :

```
<?php
   class vehicle
     {
        function car()
        {
        echo "City Honda";
        }
     }
     $obj1 = new vehicle;
     $obj1->car();
  ?>
```

Output :

```
City Honda
```

3. Special Type

(i) Resource : The special resource type is not an actual data type. It is the storing of a reference to functions and resources external to PHP. Consider a function which connect to the database, a function to send a query to the database, a function to close the connection of database. Resource variables hold special handles to opened files, database connections, streams etc.

Example :

```
<?php
// Open a file for reading
$f1=fopen("note.txt","r");
var_dump($f1);
echo"<br>";


// Connect to MySQL database server with default setting
$link=mysql_connect("localhost","root","");
var_dump($link);
```

```
?>
```

(ii) **Null :** Null is a special data type which can have only one value NULL i.e. a variable that has no value assigned to it. If a variable is created without a value, it is automatically assigned a value of NULL.

Syntax : $var_name= NULL

Example :

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

Output :

```
NULL
```

1.2.3(A) Type Juggling

Q. Define term Juggling.	**(2 Marks)**

– Type Juggling means dealing with a variable type. In PHP a variables type is determined by the context in which it is used. If an integer value is assigned to a variable, it becomes an integer. If a string value is assigned to variable, it becomes a string.

– PHP does not require (or it support) explicit type definition in variable declaration; a variable's type is determined by the context in which the variable is used.

– PHP variable types may be modified without any explicit conversion process. This conversion process, whether explicit or implicit, is known as **type juggling.**

Example

```
$var1=1;                          // $var1 is an integer
$var2="20";                                  //$var2 is a string
$var3=$var1+$var2;                // $var3 is now an integer
$var1=$var1 + 1.3;                 // $var1 is now a float
$var1=5 * "10 small birds";       // $var1 is integer (50)
$var1=5 + "10 Small puppies";     // $var1 is integer (15)
```

1.2.3(B) Type Casting

Q. Describe Type casting in PHP.	**(4 Marks)**

Typecasting is a way to convert one data type variable into the different data type. A type can be cast by inserting one of the casts in front of the variable.

Cast Operator	Conversion

(int) or (integer)	Integer
(bool) or (boolean)	Boolean
(float) or (double) or (real)	Float
(string)	String
(array)	Array
(object)	Object

Example :

```
$var1 = 10;                          // $variable1 is assigned the integer value 10.
$var2 = (double) $variable1;         // $variable2 is assigned the double value 10.0
$var3=(string)$var1                  //variable3 is assigned string "10"
$var1=(boolean)$var1                 //variable1 is assigned boolean value
```

– PHP provides datatype functions to get or set the datatype of a variable or to check whether a variable is a particular datatype. Those functions are *gettype()*, *settype()*.

– *gettype() is used to check data type of variable and settype()*is used to change a variables datatype.

Example:

```
<?php
$count = "5";
echo gettype($count);          // before datatype conversion: $count is a string
settype($count,'int');
echo gettype($count);          // after datatype conversion: $count is an integer
?>
```

Output :

```
String
integer
```

1.2.4 Expressions and Operators

Q.	What are the different operators available in PHP?	**(4 Marks)**
Q.	Describe any two comparison and two Assignment operators.	**(4 Marks)**

Operators are symbols used to manipulate data stored in variables. A value operated on by an operator is referred to as an *operand*. The combination of operands with an operator to produce a result is called an *expression*.

For Example : (1 + 2) Here integer value 1 and 2 are operands and + is the addition operator, operates on operands to produce the integer result 3.

In PHP, operators are used to perform many operations on variables or operands or values. These operators are nothing but symbols needed to perform operations of various types. PHP operators can be classified into a number of categories :

– Arithmetic Operators
– Assignment Operators
– Comparison Operators
– Increment/Decrement Operators
– Logical or Relational Operators
– String Operators
– Array Operators
– Conditional or Ternary Operators
– Spaceship Operators (Introduced in PHP 7)
– Bitwise Operators

1.2.4(A) Arithmetic Operators

Arithmetic operators are used in mathematical expression in the same way like addition, subtraction, multiplication etc, that they are used in algebra.

Operator	Description	Syntax	Operation
+	Addition	$a + $b	Sum the operands
-	Subtraction	$a- $b	Differences the operands
*	Multiplication	$a* $b	Product of the operands
/	Division	$a/ $b	Quotient of the operands
**	Exponentiation	$a** $b	$x raised to the power $y
%	modulus	$a% $b	Remainder of the operands

Example :

```
<?php
$a = 20;
$b = 10;
echo "First number:$a <br/>";
echo "Second number:$b <br/>";
echo "<br/>";
$c = $a + $b;
echo "Addtion: $c <br/>";
```

```
$c = $a - $b;
echo "Substraction: $c <br/>";
$c = $a * $b;
echo "Multiplication: $c <br/>";
$c = $a / $b;
echo "Division: $c <br/>";
$c = $a % $b;
echo "Modulus: $c <br/>";
?>
```

Output :

```
First number:20
Second number:10
Addtion: 30
Substraction: 10
Multiplication: 200
Division: 2
Modulus: 0
```

1.2.4(B) Assignment Operators

Assignment operators are used to assign the value of an expression to a variable.

Operator	Description	Syntax	Operation
=	Assign	$a = $b	Operand on the left obtains the value of the operand on right
+=	Add then Assign	$a += $b	Simple Addition same as $a = $a + $b
-=	Subtract then Assign	$a -= $b	Simple subtraction same as $a = $a – $b
*=	Multiply then Assign	$a *= $b	Simple product same as $a = $a * $b
/=	Divide then Assign (quotient)	$a /= $b	Simple division same as $a = $a / $b
%=	Divide then Assign (remainder)	$a %= $b	Simple division same as $a = $a % $b

Example :

```
<?php
$a=20;  // simple assign operator
echo "a=$a <br/>";
//add then assign operator
$a+=10;
```

```
echo "a=a+10 :$a <br/>";

// subtract then assign operator
$a-=10;
echo "a=a-10 :$a <br/>";

// multiply then assign operator
$a*=10;
echo "a=a*10 :$a <br/>";

// Divide then assign(quotient) operator
$a/=10;
echo "a=a/10 :$a <br/>";

// Divide then assign(remainder) operator
$a%=2;
echo "a=a%2 :$a <br/>";
?>
```

Output :

```
a=20
a=a+10  : 30
a=a-10   : 20
a=a*10  : 200
a=a/10   : 20
a=a%2   :  0
```

1.2.4(C) Comparison Operator

These operators are used to compare two operands and outputs the result in Boolean form. The result is always either true, if the comparison is truthful, or false, otherwise.

Operator	Description	Syntax	Operation
==	Equal To	$a == $b	Returns True if both the operands are equal
!=	Not Equal To	$a != $b	Returns True if both the operands are not equal
<>	Not Equal To	$a<> $b	Returns True if both the operands are unequal
===	Identical	$a === $b	Returns True if both the operands are equal and are of the same type
!==	Not Identical	$a !== $b	Returns True if both the operands are unequal and are

			of different types
<	Less Than	$a < $b	Returns True if $a is less than $b
>	Greater Than	$a > $b	Returns True if $a is greater than $b
<=	Less Than or Equal To	$a <= $b	Returns True if $a is less than or equal to $b
>=	Greater Than or Equal To	$a >= $b	Returns True if $a is greater than or equal to $b

Example :

```
<?php
$a=20;
$b=10;
$c=20;
var_dump($a == $c) + "<br/>";
var_dump($a != $b) + "<br/>";
var_dump($a <> $b) + "<br/>";
var_dump($a === $c) + "<br/>";
var_dump($a !== $c) + "<br/>";  //bool(false)
var_dump($a < $b) + "<br/>";    //bool(false)
var_dump($a > $b) + "<br/>";
var_dump($a <= $b) + "<br/>";  //bool(false)


var_dump($a >= $b);
?>
```

Output :

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

1.2.4(D) Increment/Decrement Operators

– Increment operators are called the unary operators as it work on single operands.

– The operator ++ adds one to the operand and -- subtract one from the operand.

Operator	Description	Syntax	Operation

++	Pre-Increment	++$a	First increments $x by one, then return $x
--	Pre-Decrement	--$a	First decrements $x by one, then return $x
++	Post-Increment	$a++	First returns $x, then increment it by one
--	Post-Decrement	$a--	First returns $x, then decrement it by one

Pre-increment and post-increment :

The x++ and ++x means the same thing when they form statements independently; they behave differently when they are used in expressions on the right hand side of an assignment statement.

Initial value of x	Expression	Final value of y	Final value of x
3	y=x++	3	4
3	y=++x	4	4
3	y=x--	3	2
3	y=--x	2	2

Example :

```
<?php
$a = 3;
echo ++$a, " First increments then prints <br/>";
//4 First increments then prints
$a = 3;
echo $a++, " First prints then increments <br/>";
//3 First prints then increments
$a = 3;
echo --$a, " First decrements then prints <br/>";
//2 First decrements then prints
$a = 3;
echo $a--, " First prints then decrements <br/>";
//3 First prints then decrements
?>
```

Output:

```
4 First increments then prints
3 First prints then increments
2 First decrements then prints
3 First prints then decrements
```

1.2.4(E) Logical or Relational Operators

Logical operators are basically used to operate with conditional statements and expressions. Conditional statements are based on conditions. Conditional statement can either be true or false.

Operator	Description	Syntax	Operation
And	Logical AND	$a and $b	True if both the operands are true else false
Or	Logical OR	$a or $b	True if either of the operand is true else false
Xor	Logical XOR	$a xor $b	True if either of the operand is true and false if both are true
&&	Logical AND	$a && $b	True if both the operands are true else false
||	Logical OR	$a || $b	True if either of the operand is true else false
!	Logical NOT	!$a	True if $a is false

Example :

```
<?php
$a = 20;
$b = 10;
if ($a == 20 and $b == 10)
    echo "True <br/>";

if ($a == 20 or $b == 30)
    echo "True <br/>";

if ($a == 20 xor $b == 5)
    echo "True <br/>";

if ($a == 20 && $b == 10)
    echo "True <br/>";

if ($a == 20 || $b == 20)
    echo "True <br/>";
?>
```

Output :

```
True
True
True
True
True
```

1.2.4(F) String Operators

Q. Explain string operators in PHP with example. **(4 Marks)**

There are two string operators. The first is the concatenation operator (‘**.**‘), which returns the concatenation of Two strings i.e join right and left arguments. The second is the concatenating assignment operator (‘**.=**‘), which appends the argument on the right side to the argument on the left side.

Operator	Description	Syntax	Operation
.	Concatenation	$a.$b	Concatenated $a and $b
.=	Concatenation and assignment	$a.=$b	First concatenates then assigns, same as $a = $a.$b

Example :

```
<?php
//first string
$a="Hello";
//second string
$b="World";
//concatenation of string
$c=$a.$b;
echo "$c <br/>";
//$a contains Hello
$a.="PHP";    //$a=
echo "$a";
?>
```

Output :

```
HelloWorld
HelloPHP
```

1.2.4(G) Array Operators

Q. Compare == and = = = operators?

We can perform operations on arrays using Arrays Operators like Union, Equality, Identity, Inequality and Non-identity. Assume $a and $b are arrays.

Operator	Description	Syntax	Operation
+	Union	$a + $b	Union of both i.e., $x and $y
==	Equality	$a == $b	Returns true if both has same key-value pair
!=	Inequality	$a != $b	Returns True if both are unequal

===	Identity	$a === $b	Returns True if both has same key-value pair in the same order and of same type
!==	Non-Identity	$a !== $b	Returns True if both are not identical to each other
<>	Inequality	$a <> $b	Returns True if both are unequal

Example :

```
<?php
$a = array("a"=>"c","b"=>"Perl");
$b = array("d"=>"Java","e"=>"Python");
var_dump($a + $b);
var_dump($a == $b) + "<br/>";
var_dump($a != $b) + "<br/>";
var_dump($a <> $b) + "<br/>";
var_dump($a === $b) + "<br>";
var_dump($a !== $b) + "<br>";
 ?>
```

Output :

```
array(4) { ["a"]=> string(1) "c" ["b"]=> string(4) "Perl" ["d"]=> string(4) "Java" ["e"]=> string(6) "Python" }
bool(false)
bool(true)
bool(true)
bool(false)
bool(true)
```

1.2.4(H) Conditional or Ternary Operators

Q. Describe : Ternary Operator in PHP with example. **(4 Marks)**

– The goal of the operator is to decide which value should be assigned to the variable. Ternary operator "?:" basically is used for an if-then-else as shorthand as :

Boolean expression ? operand1: operand2;

– First expression is evaluated. If expression is true then returns operand1; otherwise returns operand2, if the expression is false.

For example : x = (y>z) ? y : z

– X is assigned the value of y if y is greater than z, else x is assigned the value of z.

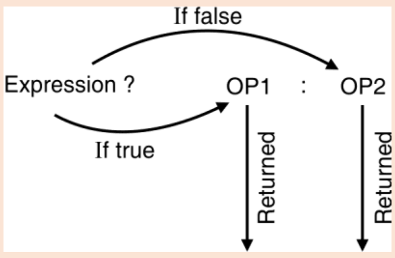


Fig. 1.2.1

Example :

```
<?php
$a=-10;
echo ($a> 0) ? 'The number is positive' : 'The number is negative';
?>
```

Output :

```
The number is negative.
```

1.2.4(I) Spaceship Operators (Introduced in PHP 7)

Q. Describe Spaceship operators available in PHP. **(4 Marks)**

PHP 7 has introduced a new kind of operator called spaceship operator (). These operators are used to compare values but instead of returning Boolean result, 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.

Operator	Syntax	Description
$a< $b	$a<=> $b	Identical to -1 (right is greater)
$a> $b	$a<=> $b	Identical to 1 (left is greater)
$a<= $b	$a<=> $b	Identical to -1 (right is greater) or identical to 0 (if both are equal)
$a>= $b	$a<=> $b	Identical to 1 (if left is greater) or identical to 0 (if both are equal)
$a == $b	$a<=> $b	Identical to 0 (both are equal)
$a != $b	$a<=> $b	Not Identical to 0

Example :

```
<?php
$a=30;
$b=30;
$c=20;
echo $a <=> $b;
echo "<br/>";
echo $a <=> $c;
echo "<br/>";
echo $c <=> $b;
echo "<br/>";
?>
```

Output :

```
 0
 1
-1
```

1.2.4(J) Bitwise Operators

The Bitwise operators is used to perform bit-level operations on the operands.

Operator	Description	Syntax	Operation
&	And	$a & $b	Bits that are set in both *$a* and *$b* are set
|	Or (inclusive or)	$a | $b	Bits that are set in either *$a* or *$b* are set.
^	Xor (exclusive or)	$a ^ $b	Bits that are set in *$a* or *$b* but not both are set.
~	Not	~ $a	Bits that are set in *$a* are not set, and vice versa.
<<	Shift left	$a << $b	Shift the bits of *ab* steps to the left (each step means "multiply by two")
>>	Shift right	$a >> $b	Shift the bits of *ab* steps to the right (each step means "divide by two")

1. **& (Bitwise AND) :** This binary operator works on two operands. Bitwise AND operator in PHP takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.

Example :

```
<?php
$a=5;
$b=3;
echo $a & $b;
echo "<br/>";
?>
```

Output : 1

(Binary representation of 5 is 0101 and 3 is 0011. Therefore, their bitwise & will be 0001)

2. **| (Bitwise OR) :**Bitwise OR operator takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 any of the two bits is 1.

Example :

```
<?php
$a=5;
$b=3;
echo $a | $b;
echo "<br/>";
?>
```

Output : 7

(Binary representation of 5 is 0101 and 3 is 0011.Therefore their bitwise | will be 0111)

3. **^ (Bitwise XOR) :** This is also known as Exclusive OR operator. Bitwise XOR takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.

Example :

```
<?php
$a=5;
$b=3;
echo $a ^ $b;
echo "<br/>";
?>
```

Output : 6

(Binary representation of 5 is 0101 and 3 is 0011. Therefore their bitwise ^ will be 0110)

4. **~ (Bitwise NOT)** : This is a unary operator i.e. works on only one operand. Bitwise NOT operator takes one number and inverts all bits of it.

Example :

```
<?php
$a=5;
echo ~$a;
echo "<br/>";
?>
```

Output : –6

(Binary representation of 5 is 0101. Therefore the bitwise ~ of this will be 1010)

5. **<< (Bitwise Left Shift) :**Bitwise Left Shift operator takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.

Example:

```
<?php
$a=5;
$b=1;
echo $a<<$b;
echo "<br/>";
?>
```

Output : 10

(Binary representation of 5 is 0101 . Therefore, bitwise << will shift the bits of 5 one times towards the left i.e. 01010)

6. **>> (Bitwise Right Shift) :**Bitwise Right Shift operator takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.

Example :

```
<?php
$a=5;
$b=1;
echo $a>>$b;
echo "<br/>";
?>
```

Output : 2

(Binary representation of 5 is 0101 . Therefore, bitwise >> will shift the bits of 5 one times towards the right i.e. 010)

1.2.4(K) Operator Precedence and Associativity

Operator Precedence: The precedence of an operator specifies how "tightly" it binds two expressions together. Operators have a set precedence, or order, in which they are evaluated. For example, in the expression 10 + 5 * 2, the answer is 20 and not 30 because the multiplication * operator has a higher precedence than the addition + operator. Parentheses () may be used to force precedence, if necessary. For instance: (10 + 5) * 2 evaluates to 30.

Operator Associativity :

When operators have equal precedence their associativity decides how the operators are grouped.

Example :

- is left-associative, so *1 - 2 - 3* is grouped as *(1 - 2) - 3* and evaluates to *-4*.

= is right-associative, so *$a = $b = $c* is grouped as $a = ($b = $c).

Following Table shows operator **precedence** and **associativity** in PHP, where operators with the highest precedence are at the top, and precedence decreases as you go down the table.

Associativity	Operators

Associativity	Operators
	Highest Precedence
n/a	()
n/a	new
right	[]
right	! ~ ++ -- (int) (double) (string) (array) (object)
left	* / %
left	+ - .
left	<< >>
n/a	< <= > >=
n/a	== != === !===
left	&
left	^
left	|
left	&&
left	||
left	? :
left	= += -= *= /= .= %= |= ^= ~= <<= >>=
right	print
left	and
left	xor
left	or
left	,
	Lowest Precedence

Example :

```
<?php
  $n1 = 10;
  $n2 = 5;
  $n3 = 2;


  $ans = $n1 + $n2 * $n3;
```

```
    # * has higher precedence than + so first execute $n2 * $n3 the answer is added to
      $n1 which is
    echo "$n1 + $n2 * $n3 = $ans<br />";

    $ans = ($n1 + $n2) * $n3;
    # () has higher precedence than * so bracket execute first ($n1 + $n2) after addition
     answer is multiplied by $n3 */
    echo "($n1 + $n2) * $n3 = $ans<br />";
?>
```

Output :

```
10 + 5 * 2 = 20
(10 + 5) * 2 = 30
```

1.2.5 Constants

Q. How can we declare variable and constant in PHP? Explain it with an example . **(4 Marks)**

– A constant is a name or an identifier for a fixed value.

– A constant value cannot change during the execution of the script.

– By convention, constant identifiers are always uppercase.

– By default, a constant is case-sensitive.

– A constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

– If you have defined a constant, it can never be changed or undefined.

– Common examples of such data include configuration settings such as database username and password, website's base URL, company name, etc.

– The define() function in PHP is used to create a constant

Syntax : define(name, value, case-insensitive)

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 :

```
<?php
// Creates a case-sensitive constant
define("HELLO", "Hello PHP");
echo HELLO, "<br/>";
 // Creates a case-insensitive constant
```

```
define("HELLO", "Hello PHP", true);
echo hello;
?>
```

Output :

```
Hello PHP
Hello PHP
```

1.3 Decision Making Control Statements

Q.	Explain different loops in PHP.	**(4 Marks)**
Q.	State any four decision making statement along with their syntax in PHP	**(4 Marks)**
Q.	How we use if..else and elseif statement in PHP?	**(4 Marks)**

Generally instructions are executed sequentially. In some cases it is necessary to change the sequence of executions based on certain conditions. For this purpose decision control structure is required.

1.3.1 if statement

The *if* statement is used to execute a block of code only if the specified condition evaluates to true.

Syntax :

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

Concept diagram :

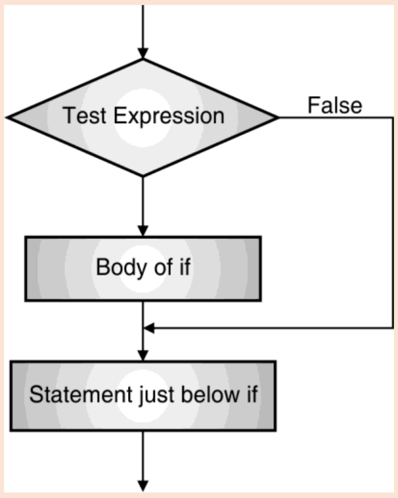


Fig. 1.3.1

Example :

```
<?php
$a=10;
if ($a > 0)
```

```
{
echo "The number is positive";
}
?>
```

Output :

```
The number is positive
```

1.3.2 if-else Statement

If...else statement first checks the condition. If condition is true, then true statement block is executed. If condition is false, then false statement block is executed.

Syntax :

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

Concept Diagram :

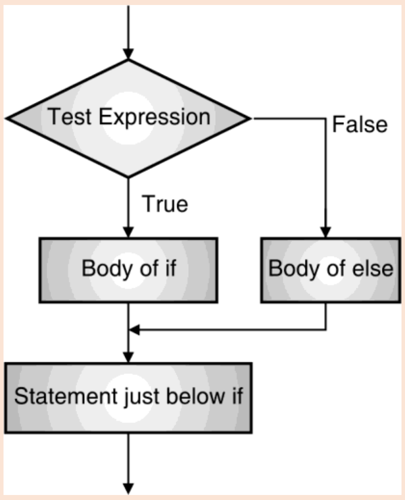


Fig. 1.3.2

Example :

```
<?php
$a=-10;
if ($a > 0)
{
echo "The number is positive";
}
else
{
echo "The number is negative";
}
?>
```

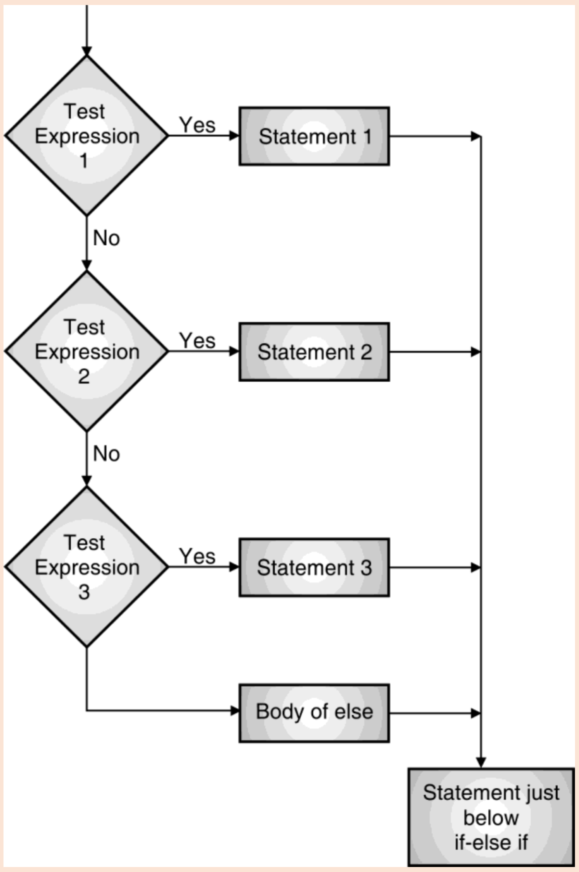
Output :

```
The number is negative
```

1.3.3 Nested-if Statement

Nested if statements mean an if block inside another if block. Nested if else statement used when we have more than two conditions. It is also called if else if statement.

Syntax :	Concept Diagram :

`if(condition1)`
`{`
` // Code to be executed if condition1 is true`
`}`
`elseif(condition2)`
`{`
`   // Code to be executed if the condition1 is false and condition2 is true`
`}`
`else`
`{`
`// Code to be executed if both condition1 and condition2 are false`
`}`
	 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-else if
	Fig. 1.3.3

Example :

```
<?php
    $per=65;
    if ($per >= 75)
    {
        echo "Distinction";
    }
    else if ($per <75 && $per >=60)
    {
        echo "First Class";
    }
    else if ($per <60 && $per >= 45)
    {
        echo "Second Class";
    }
    else if ($per <45 && $per >= 40)
     {
            echo "Pass Class";
    }
    else
    {
        echo "Sorry Fail";
    }
?>
```

Output :

```
First Class
```

1.3.4 Switch Statement

The switch-case statement is an alternative to the if-elseif-else statement, which does almost the same thing. The switch-case statement tests a variable against a series of values until it finds a match, and then executes the block of code corresponding to that match. The switch statement is used to avoid long blocks of if..elseif..else code.

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;

case statement4:
//code to be executed if n==statement4;
break;
......
default:
//code to be executed if n != any case;
}
```

Concept Diagram

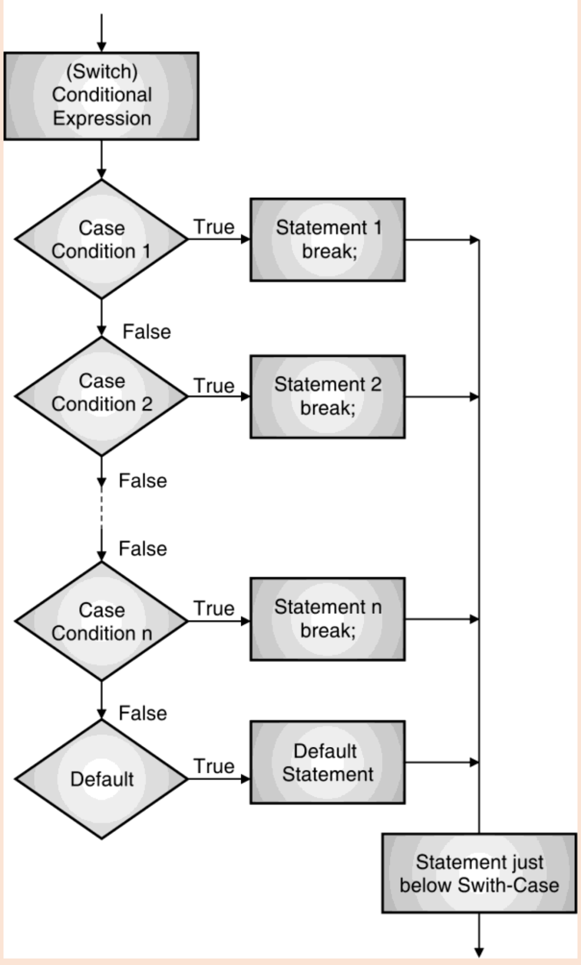


Fig. 1.3.4

Example :

```
<?php
$ch=1;
$a=20;
$b=10;
switch ($ch)
{
case 1:
$c=$a + $b;
echo "Addition=".$c;
break;
case 2:
$c=$a - $b;
echo "Subtraction=".$c;
break;
case 3:
$c=$a * $b;
echo "Multiplication=". $c;
break;
case 4:
$c=$a / $b;
echo "Division=". $c;
break;
default:
echo "Wrong Choice";
}
?>
```

Output :

```
Addition=30
```

1.3.5 Break and Continue Statement

Q.	Describe break and continue statement in PHP	**(4 Marks)**
Q.	Compare break and continue statement in PHP	**(4 Marks)**

There are two special statements that can be used inside loops:

Break and Continue

1. **Break :** The keyword break ends execution of the current for, for each, while, do while or switch structure. When the keyword break executed inside a loop the control automatically passes to the first statement outside the loop. A break is usually associated with the if.

Example :

```
<?php
for($i=1; $i<=5;$i++)
{
echo "$i<br/>";
if($i==3)
{
```

```
break;
}
}


?>
```

Output :

```
1
2
3
```

2. **Continue :** It is used to stop processing the current block of code in the loop and goes to the next iteration. It is used to skip a part of the body of the loop under certain conditions. It causes the loop to be continued with the next iteration after skipping any statement in between. The continue statement tells the compiler ”SKIP THE FOLLOWING STATEMENTS AND CONTINUE WITH THE NEXT ITERATION”.

Example :

```
<?php
for($i=1; $i<=5;$i++)
{
if($i==3)
{
continue;
}
echo "$i<br/>";
}
?>
```

Output :

```
1
2
4
5
```

1.4 Loop Control Statements

Q.	Explain conditional statements of PHP.	**(4 Marks)**

A loop causes a section of a program to be repeated a certain number of times. The repetition continues while the condition set for it remains true. When the condition becomes false, the loop ends and the control is passed to the statement following the loop. loop in PHP is used to execute a statement or a block of statements, multiple times until and unless a specific condition is met. This helps the user to save both time and effort of writing the same code multiple times.

PHP supports four different types of loops: while, do-while, for and for each.

1.4.1 while Statement

The while statement will execute a block of code if and as long as a test condition is true. The while is an entry controlled loop statement. 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
}
```

Concept Diagram :

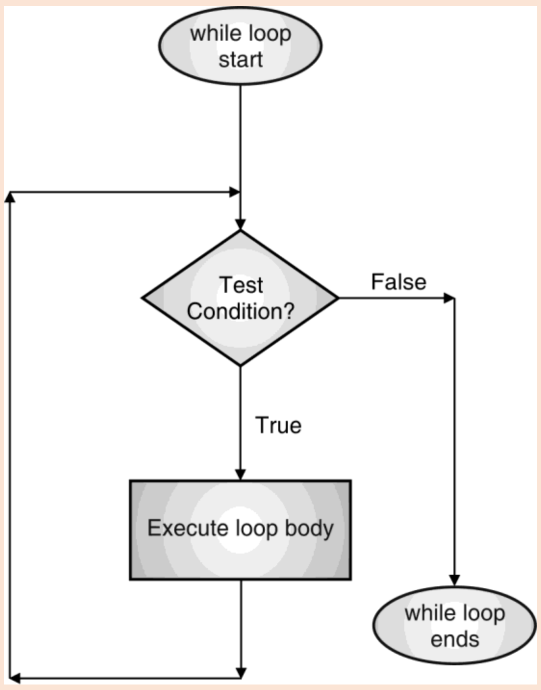


Fig. 1.4.1

Example :

```
<?php
//while loop to print even numbers
$i=0;
while ($i<= 10)
{
echo "$i<br/>";
$i+=2;
}
?>
```

Output :

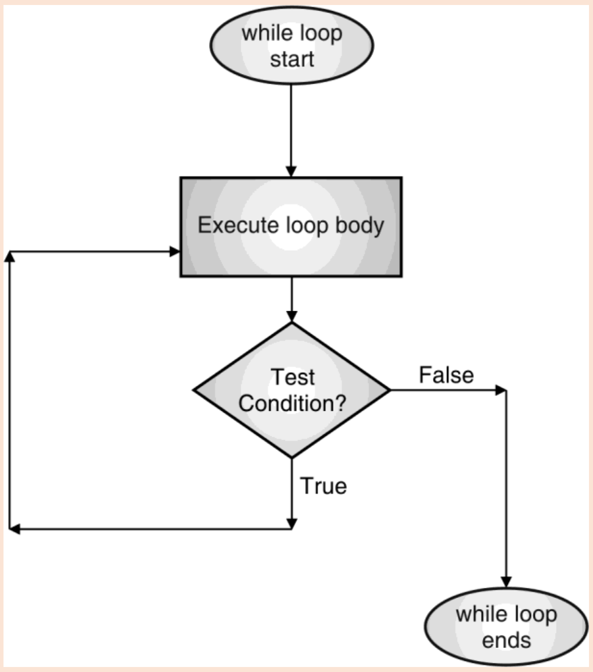
```
0
2
```

```
4
6
8
10
```

1.4.2 do-while Statement

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 :	**Concept Diagram :**

```
do
{
        //code is executed
} while (if condition is true);
```
	 while loop start
	Execute loop body
	Test Condition?
	False
	True
	while loop ends Fig. 1.4.2

Example :

```
<?php
//do-while loop to print odd numbers
$i=1;
do
{
echo "$i<br/>";
$i+=2;
}while ($i< 10);
```

```
?>
```

Output :

```
1
3
5
7
9
```

1.4.3 for Statement :

The for statement is used when you know how many times you want to execute a statement or a block of statements. That is, the number of iterations is known beforehand. These type of loops are also known as entry-controlled loops. There are three main parameters to the code, namely the initialization, the test condition and the counter.

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.

1. **Initialization Expression** : In this expression we have to initialize the loop counter to some value. for example: $i=1;
2. **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: $i<=10;
3. **Update Expression** : After executing loop body this expression increments/decrements the loop variable by some value. for example : $i+= 2;

Syntax :

```
for (initialization expression; test condition; update expression)
{
   // code to be executed
}
```

Concept Diagram :

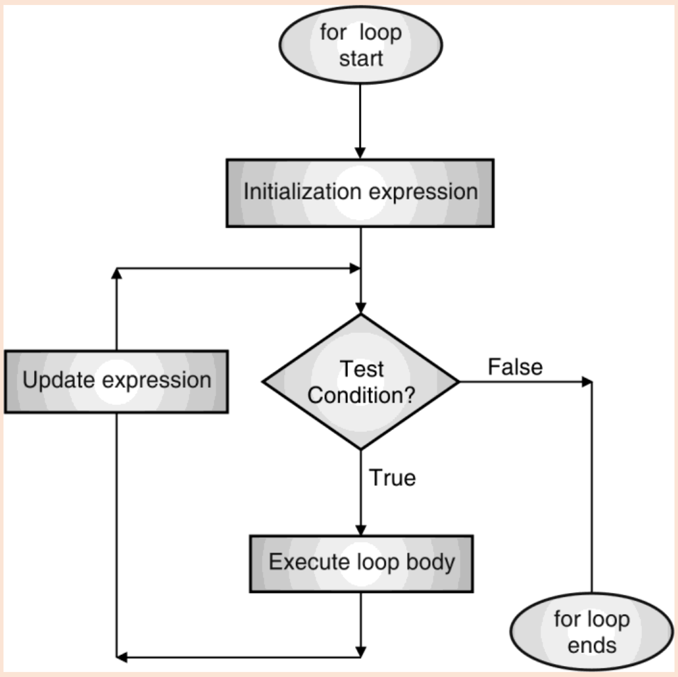


Fig. 1.4.3

Example :

```
<?php
//for loop to print even numbers and sum of them
$sum=0;
for($i=0; $i<=10;$i+=2)
{
echo "$i<br/>";
$sum+=$i;
}
echo "Sum=$sum";
 ?>
```

Output :

```
0
2
4
6
8
10
Sum=30
```

1.4.4 foreach Statement

Q.	Describe the function of foreach construct in PHP?	**(4 Marks)**
Q.	Describe role of for each in PHP	**(4 Marks)**

foreach loop is used for array and objects. 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
$arr = array (10, 20, 30, 40, 50);
foreach ($arr as $i)
{
```

```
echo "$i<br/>";
}
?>
```

Output :

```
10
20
30
40
50
```

Programs :

1. Write a program to print “Welcome to PHP”

```
<?php
echo ("Welcome to PHP");
?>
```

Output :

```
Welcome to PHP
```

2. Write a program in PHP to calculate Square Root of a number.

```
<?php
function my_sqrt($n)
{
 $x = $n;
 $y = 1;
while($x > $y)
 {
  $x = ($x + $y)/2;
  $y = $n/$x;
 }
 return $x;
}
print_r(my_sqrt(16)."<br/>");
print_r(my_sqrt(144)."<br/>");
?>
```

Output :

```
4
12
```

3. PHP program to print factorial of number

```
<?php
$num = 4;
$factorial = 1;
for ($x=$num; $x>=1; $x--)
{
  $factorial = $factorial * $x;
}
echo "Factorial of $num is $factorial";
?>
```

Output :

```
Factorial of 4 is 24
```

4. Write PHP program to print Fibonacci series.

```
<?php
$num = 0;
$n1 = 0;
$n2 = 1;
echo "<h3>Fibonacci series for first 12 numbers: </h3>";
echo "\n";
echo $n1.' '.$n2.' ';
while ($num<10 )
{
    $n3 = $n2 + $n1;
    echo $n3.' ';
    $n1 = $n2;
    $n2 = $n3;
    $num = $num + 1;
}
?>
```

Output :

```
Fibonacci series for first 12 numbers:
0 1 1 2 3 5 8 13 21 34 55 89
```

5. Write a PHP program to print prime number upto n

```
<?php
$count = 0 ;
$number = 2 ;
while ($count <20 )
{
$div_count=0;
for ( $i=1;$i<=$number;$i++)
{
if (($number%$i)==0)
{
$div_count++;
}
}
if ($div_count<3)
{
echo $number." , ";
$count=$count+1;
}
$number=$number+1;
}
?>
```

Output :

```
2 , 3 , 5 , 7 , 11 , 13 , 17 , 19 , 23 , 29 , 31 , 37 , 41 , 43 , 47 , 53 , 59 , 61 , 67 , 71 ,
```

6. Write a PHP program to print sum of digits of a given number.

```
<?php
$num = 12345;
$sum=0; $rem=0;
 for ($i =0; $i<=strlen($num);$i++)
 {
 $rem=$num%10;
```

```
  $sum = $sum + $rem;
  $num=$num/10;
 }
echo "Sum of digits 12345 is $sum";
?>
```

Output :

```
Sum of digits 12345 is 15
```

7. Write PHP program to print reverse of number

```
<?php
$num = 123;
$revnum = 0;
while ($num> 1)
{
$rem = $num % 10;
$revnum = ($revnum * 10) + $rem;
$num = ($num / 10);
}
echo "Reverse number of 123 is: $revnum";
?>
```

Output :

```
Reverse number of 123 is: 321
```

8. Write PHP program to print even numbers upto 10 and sum of them

```
<?php
$sum=0;
for($i=0;$i<=10;$i+=2)
{
echo "$i<br/>";
$sum=$sum+$i;
}
echo "Sum: $sum";
?>
```

Output :

```
0
2
4
6
8
10
Sum : 30
```

9. Write a program to find average of first ten natural numbers using for loop.

```
<?php
$sum=0;
for ($i=1; $i<=10; $i++)
{
echo "$i<br/>";
 $sum=$sum+$i;
}
echo "sum=$sum <br/>";
$average=(float)$sum/$i;
echo "Average=$average"
?>
```

Output :

```
1
2
3
4
5
6
7
8
9
10
sum=55
Average=5
```

Review Questions

Q, 1. Which tag is used to define PHP code?

Q, 2. "Whole PHP code (script) is case sensitive". Justify true or false.

Q, 3. Write features of PHP.

Q, 4. What are the two types of comment tags?

Q, 5. Give difference between foreach and for statement

Q, 6. Example of static, local and global variable. Explain it.

Q, 7. How we use if..else and elseif statement in PHP?

Q, 8. How can we use while, do..while & switch statement.

Q, 9. Explain loop control structure with suitable example

Q, 10. Why PHP is known as Scripting language

Q, 11. Explain identical operator?

Q, 12. Explain the bitwise operator with suitable example.

Q, 13. What is the difference between ++$j and $j++?

Q, 14. How do you convert one variable type to another (say, a string to a number)?

Q, 15. Difference between variables and constants in PHP?

Q, 16. Explain in detail about Control Structures in PHP

Q, 17. “Whole PHP code (script) is case sensitive”. Justify true or false.

Q, 18. Why var_dump( ) is preferable over print_r( ) ?

Q, 19. What is type juggling

Q, 20. Write any 5 web server name

❑❑❑