Prepared By Dr. N.THENMOZHI M.C.A., M.S., M.Phil., Ph.D.,

Govt, Arts College (Autonomous), Coimbatore-18

Department of Information Technology

OPEN SOURCE TOOLS (18MIT33C) -----II MSc

UNIT-III: PHP: Introduction – Identifier -Variables - Constants – Data types – Operators - Statements – loops. Advanced PHP –Arrays – Get and Post – Object oriented concepts – Strings –File handling and data storage. MySQL Databases – Setting –Starting, terminating and writing own SQL programs – record selection technology, strings functions, date and time – starting query – generating summary – working with metadata – using sequences – PHP and MySQL databases.

TEXT BOOKS

1. M.N. Rao, "Fundamentals of open source software", PHI Learning Private Limited, 2015.

REFERENCE BOOKS

1. Elliot White III, Jonathan.D.Eisenhamer, "PHP 5 in practice" pearson Education,2007.
2. Paul Du Bois,O Reilly Publishers,"My SQL- Cookbook",Second Edition,2010.

UNIT-III: PHP: Introduction – Identifier -Variables - Constants – Data types – Operators - Statements – loops. Advanced PHP –Arrays – Get and Post – Object oriented concepts – Strings –File handling and data storage. MySQL Databases – Setting –Starting, terminating and writing own SQL programs – record selection technology, strings functions, date and time – starting query – generating summary – working with metadata – using sequences – PHP and MySQL databases.

3.1 INTRODUCTION

PHP is an open source scripting language designed for web development to produce dynamic web pages. As it is an open source scripting language, anyone can view, modify and redistribute the source code. PHP was created by Rasmus Lerdrof in 1995. PHP stands for Hypertext Preprocessor; originally, it was called Personal Home Page Tools.

Just like java script, PHP is a loosely typed language as it can automatically convert the variable to a correct data type depending on its value. PHP commands are based on PERL, java, and C languages which provide a good platform for creating dynamic content. PHP commands end with semicolon (;), and can make up an entire file or can be embedded in HTML. PHP server recognizes embedded script and executes the PHP program. The PHP program result (which is generated by the PHP parser) will be passed to the browser through web server, but the source code is not visible. PHP can be deployed on web servers, operating systems and platforms. Therefore, the PHP program can be used for many relational databases.

3.1.1 Setting up Environment

Applications can be executed by setting up the following PHP environment in the system:

Prerequisite

Install XAMPP server which contains a PHP module on the top of the web server– Apache with MySQL for windows operating system.

Installation Procedure for XAMPP

XAMPP installation includes the following steps:
Step 1: To download XAMPP visit the website. www.apachefriends.org/en/xampp.htm/
Step 2: Click on the 'Installer' in the above mentioned website.
Step 3: 'XAMPP-win32-1. 7.0-Installer. exe' file is to be installed.
Step 4: Installation wizard to install XAMPP will appear on the screen.
Step 5: Choose the destination folder in any of the local drive(c:\XAMPP), then click next.
Step 6: Select 'Install XAMPP, Install MySQL', and click on install.
Step 7: Click on finish, your XAMPP server is installed on the system.

3.1.2 PHP Script Execution: Interaction of Web Browser and Web Server

- A web browser requests a page and that request is sent to the web server.
- If the page contains PHP, it is sent to the PHP parsing engine.
- The PHP parsing engine executes the PHP code and places the result in the page.
- The web server sends the HTML page back to the web browser.
- The browser displays the page.

Advantages of PHP

- It is easy to learn and understand.
- It is highly flexible and user-friendly.
- It can handle many complex web applications.
- It is cross-platform, since it is used on various major operating platforms, like Unix, Linux, RISC OS, Mac OS X, Microsoft Windows, etc.
- It is an open source.
- It is most compatible language.
- It includes various security functions to provide security to the developed sites.
- It helps programmers to create dynamic websites, thereby leading to high profits.
- It supports both structured and object-oriented programming.
- It provides powerful built-in functions.

3.1.3 Basic Syntax for PHP

A PHP script should start with. <? PHP and ends with?>. The general form of the PHP script is as follows:

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

In HTML Document, PHP script can be placed anywhere.

3.2 IDENTIFIERS

An identifier is a name that identifies either a unique object or a unique class of objects, where the object may be an idea, physical object or substance. Identifiers can be of any length and can consist of letters, numbers, words and underscores (_). Function names are not case sensitive and can be either in upper case or lower case. Identifiers are used to refer the following objects:

- Variables (when prefixed with $)
- Classes
- Constants
- Functions
- Formal parameters

3.2.1 Limitations of Identifiers

- Reserved words cannot be used as identifiers.
- Syntactic keywords are treated as reserved words in PHP.
- Different elements refer to the same identifier by using different scopes.
- It is better to avoid identifiers beginning with an underscore (_).
- It cannot begin with a digit.
- These are case sensitive. For example, $v and $V are two different variables.

3.2.2 Keywords

Following are the keywords that are used in PHP scripting:

abstract, and, array(), as, break, callable(PHP 5.4), case, catch, class, clone, const, continue, declare, default, die(), do, echo, else, elseif, empty(), enddeclare, endfor, endforeach, endif, endswitch, endwhile, eval(), exit(), extends, final, for, foreach, function, global, goto(PHP 5.3), if, implements, include, include_once, instanceof, insteadof(PHP 5.4), interface, isset(), list(), namespace (PHP 5.3), new, or, print, private, protected, public, require, require_once, return, static, switch, throw, trait(PHP 5.4), try, unset(), use, var, while, xor.

3.3 VARIABLES

A variable is a symbolic name associated with a value. The associated value may be changed. A variable type refers to the kind of data stored in it. Dollar sign is used before the name of the variable in PHP. Here the variable name is case sensitive. Rule for a valid variable name starts with a letter or underscore, followed by any number of numbers, letters, or underscores. The variable can be assigned with null value too, if not assigned it is undeclared value. A variable can have short names or descriptive names, for example v , I , o , customer name, and student address. Following are the simple examples for the valid and invalid variables:

```
$4PWD // invalid variable; starts with a number
$_4PWD // valid variable; starts with an underscore
$v = 12;
$v = $w + $x ;
```

Value Assignment

Assignment by value simply involves copying the value of the assigned expression to the variable assignee. For example, $day = “Sunday”;

Example 3.1 Demonstration of the value assigned for variables:

```
<HTML>
        <BODY>
                <?php
                        $a=1;
                        $b=2;
                        print ‘sum=’;
                        print $a+$b;
                ?>
         </BODY>
</HTML>
```

The output generated by this program is sum=3

3.3.1 Scope of Variables

PHP has four different scopes of variables:

Local

Global

Static

Function Parameter

Local Variable

A variable declared within a PHP function is local variable, and can only be accessed within that function.

Example 3.2 Local variable:

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

The output generated by this program is: 5

Global Variable

A variable that is defined outside of any function, has a global scope.

Example 3.3 Global variable:

```
<?php
        $a = 7; // global scope
        function Example ()
        {
                $a=5;
                echo $a;
                global $a;
                echo $a ;
         }
        Example ();
?>
5  ------ local variable o/p
7 ------global
```

Static Variable

Static keyword is used when a function is completed and all of its variables are normally deleted. However, sometimes you want a local variable not to be deleted.

Example 3.4 Static variable:

```
<?php
        function Example()
        {
        static $a=0;
        echo $a ;
        $a++;
        }
        Example ();
        Example ();
        Example ();
        Example ();
?>
```

The output generated by this program is: 0123

Function Parameters

A parameter is a local variable whose value is passed to the function by the calling code.

Example 3.5 Function parameters:

```
<?php
        Function Example ($a)
        {
                echo $a ;
        }
        Example (8);
?>
```

The output generated by this program is: 8

3.4 CONSTANTS

A constant can be defined as an identifier (name) for a simple value. As the name suggests, that the value will not change throughout the execution of the script. A constant name is case sensitive. By convention, constant identifiers are generally uppercase. There are two types of PHP constants:

- User-defined constants
- Built-in PHP constants

Defining PHP Constants

To define a PHP constant use define (“IDENTIFIER”, VALUE);

Example 3.6 User defined and built-in PHP constants:

```
define (“PIE”,3.14159265359); // User defined constants
define (“PASSWORD”,“CSE”); // Built-in PHP constants
```

3.4.1 Comments in PHP

Comments are used in the PHP script to explain the coded statements. It is very important to have comments in order to understand, debug and update PHP script in the later stage. There is no need to delimit the comments as PHP statements, because the new line (“/n”) character fulfills this need. Comments will not be displayed in the output, they are used to explain the source code. They are useful in the development of software projects. Single-line comments should start with //, a multi-line comments start with /* and ends with */.

Example 3.7 Single line and multi-line comments:

```
<?php
        /* Multiple line comments */
        print “helloworld”; // Single line comment

?>
```
The output generated by this program is: helloworld

3.4.2 Output Functions

PHP offers different functions for displaying the output. Output data to the browser is done using the echo(), print(), printf() and sprintf() statements. The echo() function outputs one or more strings. It is slightly faster than print() . The print() function outputs one or more strings. The print() function is not actually a function. Therefore, parentheses are not used with it. The printf() function is ideal for displaying a combination of static text and dynamic information stored within one or several variables. The sprintf() function is identical to printf(), except that the output is assigned to a string.

Example 3.8 Echo, print, printf and sprintf output statements: Echo “hi” Print “hi”

```
 <?php
        printf (“Hello Roll Number:%d”,501); // %d is the type specifier for integer
?>
```
The output generated by the program is: Hello Roll Number: 501

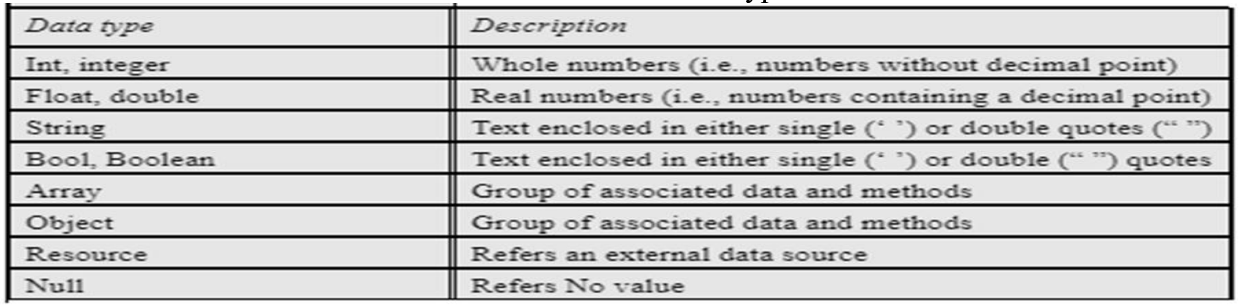
```
<?php
        $first = 23;
        $second = sprintf(“%f”,$first);
        echo $second;
?>
```

The output generated by this program is: 23.000000

3.5 DATA TYPES

In computer programming, a data type or simple type is a division identifying one of various types of data, such as real-valued , integer or boolean . It also exhibits the way in which the values of that type can be stored. PHP is a dynamically typed language, there is no need to declare the variable previously. The type of a variable is determined by the value allocated to it. List of data types are given in Table 3.1.

Table 3.1 List of Data Types



Data type	*Description*
Int, integer	Whole numbers (i.e., numbers without decimal point)
Float, double	Real numbers (i.e., numbers containing a decimal point)
String	Text enclosed in either single (‘ ’) or double quotes (“ ”)
Bool, Boolean	Text enclosed in either single (‘ ’) or double (“ ”) quotes
Array	Group of associated data and methods
Object	Group of associated data and methods
Resource	Refers an external data source
Null	Refers No value

Example 3.9 Different data type

$price = 20.50;
$items = 10;
$name = ‘mathews’;
As 10 is assigned to the items, it is now an integer type variable. Similarly, price is assigned as 20.50. Therefore, it is of type float . The name is assigned to ‘mathews’, its type will be of string type.

PHP supports eight primitive types, which are shown in Table 4.1

<u>Four Scalar Data Types</u>
Boolean : This data type supports only two values, one for true and another for false, it is case insensitive, zero is used to represent false and nonzero is used to represent true.
Integer : This data type is used to represent whole numbers, or in other words, to represent a number which does not contain fractional parts.
Float: This data type is also called double or real data type which is used to represent a number that contains fractional parts.
String : This data type is used for a sequence of characters put in a contiguous group. Strings are delimited by single quotes, double quotes or string interpolation section.
PHP treats strings in the same fashion as of arrays. The specific characters in the array can be accessed or processed using the offset value of the array.

Example 3.10 Strings:

$name = “Delhi”;
You can retrieve a particular character of string:
$letter = $color[2]; // assigns ‘o’ to $letter

Two Compound Data Types
The array: First one is an array of complex data type which handles a group of elements. Elements in an array can be accessed through an index. In PHP, arrays are different, and treated as lists or dictionaries. The keyword array is used to create a group of elements.

Example 3.11 Array data type:

$fruits = array (“Mango”, “Apple”, “Banana”, “Grapes”);

An object: An object is an instance of a class. Objects are user defined data types. Developers can make their data types that fit their domain.

Two Special Types
Resource: We can say that the resources are special data types, as they grasp a reference to an external resource, and special functions are used to create them. Resources are handlers to open files, image canvas areas of database connections.

Null: It is another special data type. Basically, the meaning of the null data type is non-existent and not known or empty.

A variable in PHP is null in three cases:

- When it was not assigned a value.
- When it was assigned a special null constraint.
- It was in set with the unset() function.

3.5.1 Typecasting

We often work with multiple data types at the same time. Converting one data type to a different data type is a common work in programming. Type conversion or typecasting is changing an object of one data type to another data type. There are two types of conversion– implicit and explicit conversions.

Example 3.12 Type conversion from string to integer:

```
<?php
        $var123 = “5bar”; // string
        settype($var123,“integer”); // $var123 is now set to 5 (integer)
        echo $var123;
?>
```

The output generated from this program is: 5

In Example 3.12 $var123 is assigned to string value, so, now it is a string variable. Now, $var123 is enforced to type cast to integer by using settype() function.

3.5.2 Embedding PHP in HTML

Example 3.13 shows how to embed a PHP code in HTML page. As PHP is a server side script which is executed only on the server and returns equivalent HTML code at the browser end, the browser can interpret HTML and Java script.

Example 3.13 Hello World script in PHP:

```
 <html>
        <body>
                <?php
                        print “Hello world”;
                ?>
        </body>
</html>
```

The output generated from the program is: Hello world

3.5.3 Expression

An expression is a combination of explicit values , constants , variables , operators and functions that are interpreted according to the particular rules of precedence and associativity for a particular programming language, which computes and then produces another value. The value can be of many types Such as, numerical, string and logical.

Example 3.14 Expression
$d = 25;
$name = “jack”;
$price = 30 + $rate;
$counter++;

3.6 OPERATORS

In an expression, the operator tells us what action to be performed on the operands. Following are the different types of operators in PHP:

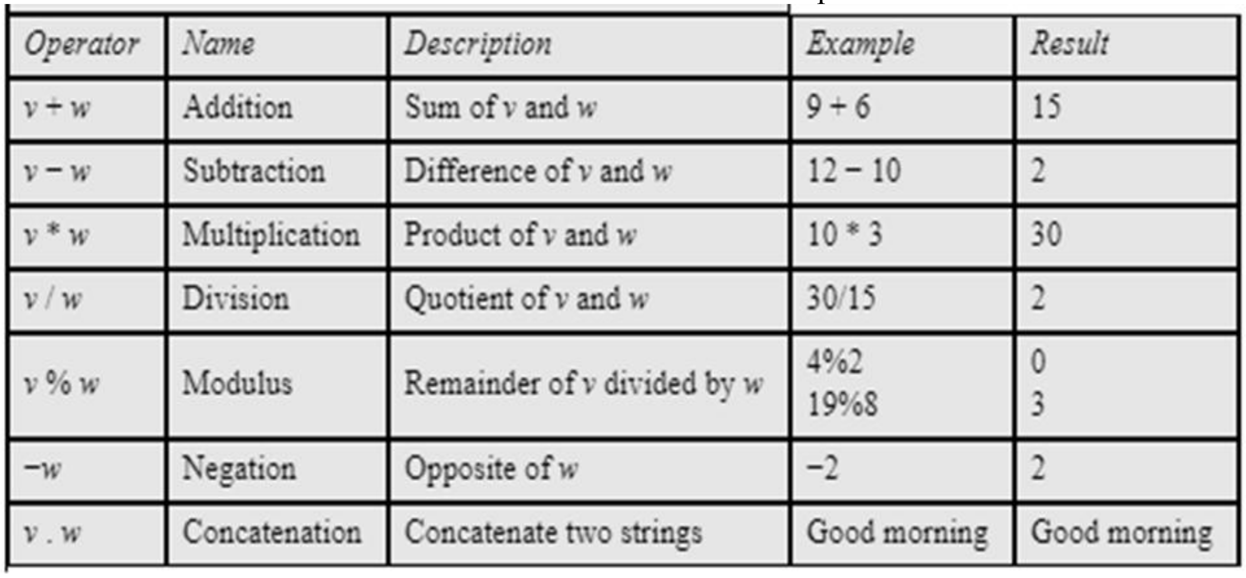
- Arithmetic Operators
- Logical Operators
- Assignment Operators
- Comparison Operators
- Bitwise Operators
- Incrementing/Decrementing Operators
- String Operators
- Array Operators

3.6.1 Arithmetic Operators

The arithmetic operators are used to perform various arithmetic operations on the variables and values.

The operators given in Table 3.2 are used frequently to perform various arithmetic operations in the PHP programs. PHP also provides various mathematical functions like calculating logarithms, square values, etc.

Table 3.2 List of Arithmetic Operators



Operator	*Name*	*Description*	*Example*	*Result*
v + *w*	Addition	Sum of *v* and *w*	9 + 6	15
v − *w*	Subtraction	Difference of *v* and *w*	12 − 10	2
v * *w*	Multiplication	Product of *v* and *w*	10 * 3	30
v / *w*	Division	Quotient of *v* and *w*	30/15	2
v % *w*	Modulus	Remainder of *v* divided by *w*	4%2 19%8	0 3
−*w*	Negation	Opposite of *w*	−2	2
v . *w*	Concatenation	Concatenate two strings	Good morning	Good morning

Example 3.15 Arithmetic operators

```
<?php
$a=10;
$b=3;
print $a%$b; //Modulus Operator [%]
?>
```

The output generated from this program is:

```
1
```

Example 3.16 Arithmetic operators

```
<?php
// Set a couple of sample integer variables
$v = 7;
$w = 9;
// Addition >>> Sum of $v and $w
echo $v + $w;
echo “<br />”;
// Subtraction >>> Difference of $v and $w
echo $v - $w;

echo “<br />”;
// Multiplication >>> Product of $v and $w
echo $v * $w;
echo “<br />”;
// Division >>> Quotient of $v and $w
echo $v / $w;
echo “<br />”;
//Modulus >>> Remainder of $v divided by $w
echo $v % $w;
echo “<br />”;
//Negation >>> Opposite of $v
echo -$v;
?>
```

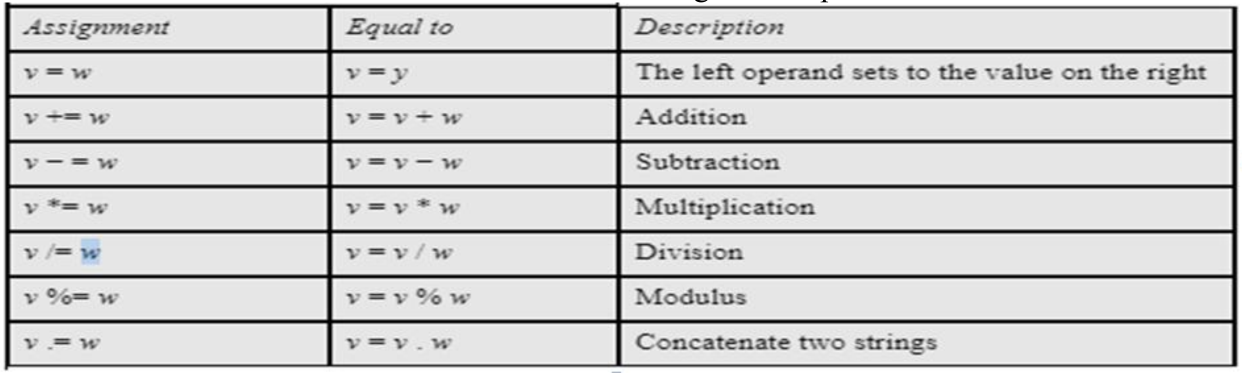
The output generated by this program is:

```
16
−2
63
0.77777777777778
7
−7
```

3.6.2 Assignment Operators

The assignment operator is used to set a value to a variable or set a variable to another variable's value. The assignment operator '=' is set to left hand variable with a right hand value. Assignment operators are shown in Table 3.3.

Table 3.3 List of Assignment Operators



Assignment	*Equal to*	*Description*
v = w	*v = y*	The left operand sets to the value on the right
v += w	*v = v + w*	Addition
v – = w	*v = v – w*	Subtraction
*v *= w*	*v = v * w*	Multiplication
v /= w	*v = v / w*	Division
v %= w	*v = v % w*	Modulus
v .= w	*v = v . w*	Concatenate two strings

Example 3.17 Assignment Operators

```
<?php
$v = 11;
echo $v;
?>
```

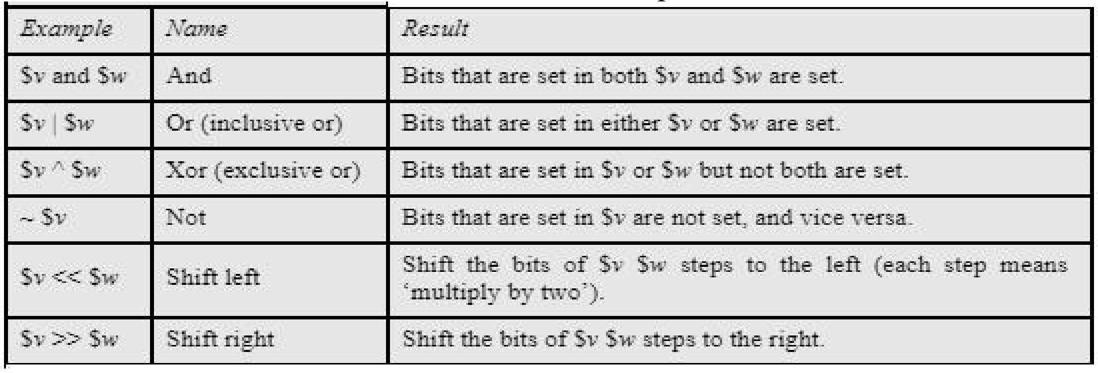
The output generated by this program is:

```
11
```

3.6.3 Bitwise Operator

Bitwise operators examine and manipulate integer values on the level of individual bits that make the integer value. These operators are shown in Table 3.4.

Table 3.4 Bitwise Operator



Example	*Name*	*Result*
$*v* and $*w*	And	Bits that are set in both $*v* and $*w* are set.
$*v* | $*w*	Or (inclusive or)	Bits that are set in either $*v* or $*w* are set.
$*v* ^ $*w*	Xor (exclusive or)	Bits that are set in $*v* or $*w* but not both are set.
~ $*v*	Not	Bits that are set in $*v* are not set, and vice versa.
$*v* << $*w*	Shift left	Shift the bits of $*v* $*w* steps to the left (each step means 'multiply by two').
$*v* >> $*w*	Shift right	Shift the bits of $*v* $*w* steps to the right.

Example 3.18 Bitwise Operators

```
<?php
$a=10;
$b=2;
echo "\$a & \$b = ".($a & $b)."<br/>";
?>
```

The output generated by this program is:

```
$a and $b = 2
```

3.6.4 Incrementing/Decrementing Operators

Increment and decrement operators are unary operators. The increment/decrement operator is useful for addition and reduction of principal value by applying pre- and post-principle. PHP supports C language style post and pre-increment/decrement operators. These operators operate only on variables and not on any value. They do not upset Boolean values. The incrementing/decrementing operators are shown in Table 3.5.

Table 3.5 List of incrementing /decrementing operators

Operator	*Name*	*Description*
++ *v*	Pre-increment	Increments *v* by one, then returns *v*.
v ++	Post-increment	Returns *v*, then increments *v* by one.
-- *v*	Pre-decrement	Decrements *v* by one, then returns *v*.
v --	Post-decrement	Returns *v*, then decrements *v* by one.

Example 3.19 Incrementing/Decrementing operators

```
<?php
$v = 9;
// Use Post-increment to increment the value by 1
$v++;
// Now display to browser or use its new value in the script
echo $v;
?>
```

The output generated by this program is:

```
10
```

3.6.5 Comparison Operators

Comparison operators are used to compare two similar values. The comparison is read from left to right by the PHP engine. If an integer is compared with a string, the string is converted to a number, and if two numerical strings are compared, they are compared as integers. The operators given in Table 3.6 are used frequently to perform various comparison operations in the PHP programs.

Table 3.6 List of Comparison Operator

Operator	*Name*	*Description*	*Example*
v == *w*	Equal	True, if *v* is equal to *w*	15==6 returns, false
v === *w*	Identical	True, if *v* is equal to *w*, and they are of the same type	5===“5” returns, false
v != *w*	Not equal	True, if *v* is not equal to *w*	1!=9 returns, true
v <> *w*	Not equal	True, if *v* is not equal to *w*	4<>7 returns, true
v !== *w*	Non-identical	True, if *v* is not equal to *w*, or they are not the similar type	4!==“4” returns, true
v > *w*	Greater than	True, if *v* is greater than *w*	2>8 returns, false
v < *w*	Less than	True, if *v* is less than *w*	7<8 returns, true
v >= *w*	Greater than or equal to	True, if *v* is greater than or equal to *w*	3>=9 returns, false
v <= *w*	Less than or equal to	True, if *v* is less than or equal to *w*	5<=8 returns, true

Example 3.20 Comparison Operator

```
<?php
$v = 9;
$w = 3;
// Is Greater Than Comparison
echo $v > $w; // should output a "1" meaning TRUE
?>
```

The output generated by this program is:

```
1
```

3.6.6 Logical Operators

Logical operators are used to combine operations and expressions into a set of logical comparisons. They are usually used in conjunction with 'if' and 'else' statements to layout the dynamic logic which we need in many situations. The operators given in Table 3.7 are used frequently to perform various logical operations in the PHP programs.

Table 3.7 List of Logical operators

Operator	*Name*	*Description*	*Example*
v and *w*	And	True, if both *v* and *w* are true	*v*=9 ; *w*=7 ; (*v* < 13 and *w* > 3) returns, true
v or *w*	Or	True, if either or both *v* and *w* are true	*v*=9 ; *w*=7 ; (*v*==9 or *w*==7) returns, true
s xor *t*	Xor	True, if either *x* or *y* is true, but not both	*s*=9 ; *t*=7 ; (*s*==9 xor *t*==7) returns, false
v && *w*	And	True, if both *v* and *w* are true	*v*=9 ; *w*=7 ; (*v* < 10 && *w* > 1) returns, true
v || *w*	Or	True, if either or both *v* and *w* are true	*v*=9 ; *w*=7; (*v*==5 || *w*==5) returns, false
! *v*	Not	True, if *x* is not true	*v*=9 ; *w*=7 ; !(*v*==*w*) returns, true

Example 3.21 Logical operator

```
<?php
$v = 5;
$w = 8;
if (($v == 1) && ($w == 4))
{
   echo "The values are equal.";
}
else
   {
   echo "The values are not equal.";
}
?>
```

The output generated by this program is:

```
The values are not equal.
```

3.6.7 String Operators

PHP's string operators provide a convenient way in which to concatenate strings together. There are two such operators, including the concatenation operator (.) and the concatenation assignment operator ".=", here concatenate combines couple of objects together. Following are the examples for string operations:

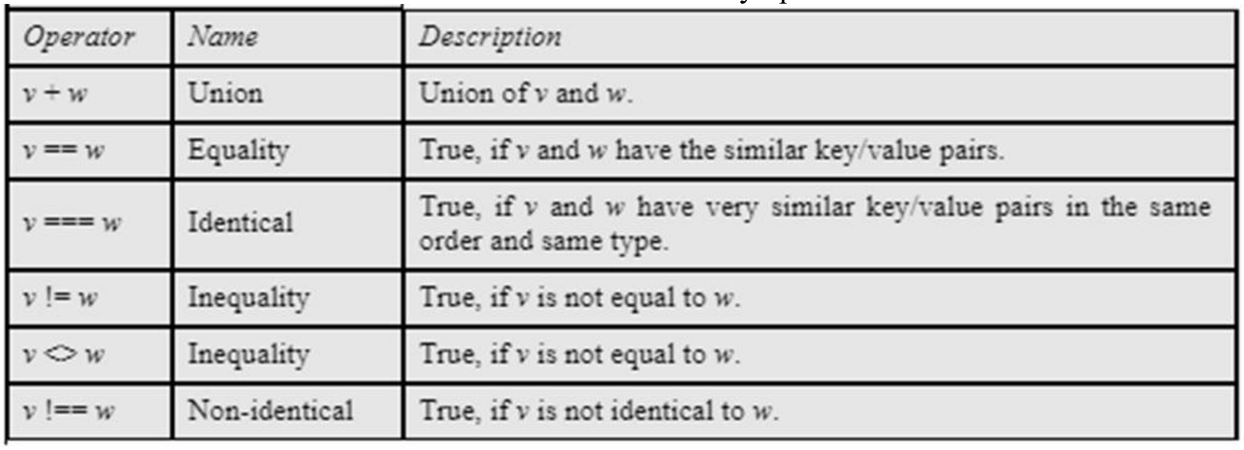
$z= "rst"."uvw"; Concatenation $z is assigned the string "rstuvw"

$x .= "stuvw"; Concatenation-assignment $x equals its current value concatenated with "stuvw"

3.6.8 Array Operators

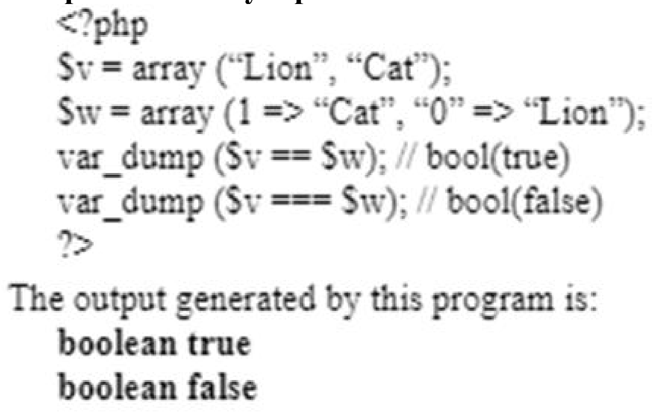
The array operators enable you to access array elements. We use '=>' operator in some contexts to represent array elements. The array operators are listed in Table 3.8.

Table 3.8 List of Array operators



Operator	*Name*	*Description*
v + *w*	Union	Union of *v* and *w*.
v == *w*	Equality	True, if *v* and *w* have the similar key/value pairs.
v === *w*	Identical	True, if *v* and *w* have very similar key/value pairs in the same order and same type.
v != *w*	Inequality	True, if *v* is not equal to *w*.
v <> *w*	Inequality	True, if *v* is not equal to *w*.
v !== *w*	Non-identical	True, if *v* is not identical to *w*.

Example 3.22 Array Operator



```
<?php
$v = array ("Lion", "Cat");
$w = array (1 => "Cat", "0" => "Lion");
var_dump ($v == $w); // bool(true)
var_dump ($v === $w); // bool(false)
?>
```

The output generated by this program is:

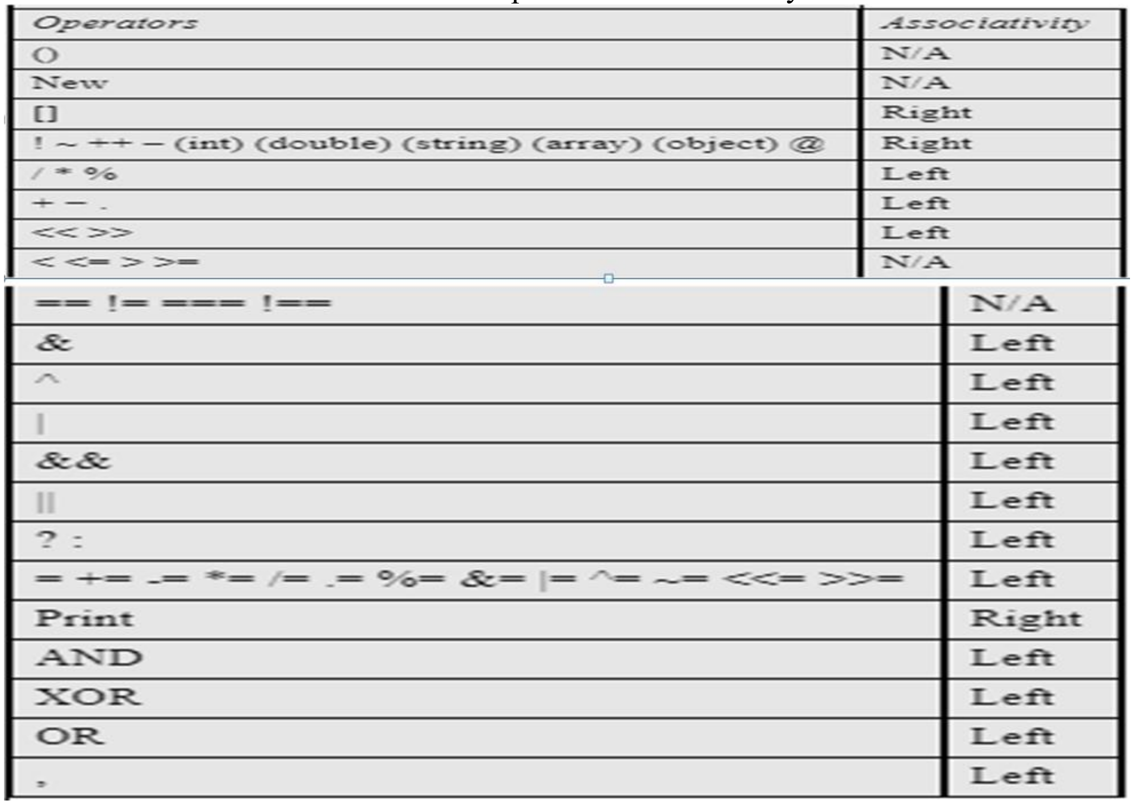
```
boolean true
boolean false
```

3.6.9 Operator Precedence and Associativity

PHP includes a set of rules. They enable us to determine in which order the operators are to be evaluated in complicated expressions. If the expression contains operators of same precedence then the rules will specify the processing order either left to right or right to left or not associative. In simple words, precedence tells the priority of the operator, whereas associativity tells the direction of evaluation of the expression, if the expression contains operators of the same precedence. Table 3.9 shows the operators and their associativity.

If the expression is evaluated from left to right, then it is called left associativity and if the expression is evaluated from right to left, then it is called right associativity. There is one more associativity–non associativity, which is neither left associativity nor right associativity. The best example for non-associativity is a comparison operator.

Table 3.9 Operator's Associativity



Operators	Associativity
()	N/A
New	N/A
[]	Right
! ~ ++ – (int) (double) (string) (array) (object) @	Right
/ * %	Left
+ – .	Left
<< >>	Left
< <= > >=	N/A
== != === !==	N/A
&	Left
^	Left
|	Left
&&	Left
||	Left
? :	Left
= += -= *= /= .= %= &= |= ^= ~= <<= >>=	Left
Print	Right
AND	Left
XOR	Left
OR	Left
,	Left

Example 3.23 Operator precedence and associativity:

Consider the expression 2+3*4/3. This expression contains the operators '*' and '/' which are having the same precedence, and in addition these operators are given higher priority than the'+' operator. So, the evaluation of expression should be from left to right according to Table 4.9.

Step 1: 3*4=12 now expression becomes 2+12/3.

Step 2: 12/3=4 now expression becomes 2+4.

Step 3: 2+4=6 now result of the expression is 6.

3.7 STATEMENTS

Statements are the control structures which are useful in controlling the flow of the code within an application. They are used to define characteristics, such as, how many times a particular statement will execute and when the control will come out of the execution.

3.7.1 Conditional Statements in PHP

Conditional statements will execute a statement or a group of statements depending on the input given by the user. There are four types of conditional statements as follows:

1. If statement
2. If... else statement
3. If...elseif.... Else statement
4. Switch Statement If Statement

If statement

The ‘if’ statement is used to execute some code only if a specified condition is true . The condition in the ‘if’ statement is enclosed in parenthesis. The general form of if statement is:

```
if(condition)
{
        Code to be executed if the condition is true;
}
```

Example 3.24 Simple ‘if’ statement:

```
<?php
$a=2;
if($a==2)
echo “the value of a is 2”;
?>
```

The output generated by this program is: the value of a is 2

The variable a is assigned with the value 2, and then ‘if’ statement is used to compare the value of variable a with the value 2. Since, the condition specified within the parenthesis of ‘if’ statement is true, the output is printed as shown above.

Example 3.25 Simple ‘if’ statement:

```
<?php
$str1=’hai’;
$str2=’bye’;
if($str1 == ‘hai’)
{
$str2= ‘fine’;
}
echo $str2;
?>
```

The output generated by this program is: fine

The condition specified in the ‘if’ statement, that is, if str1 is equal to ‘hai’, then the block of code within the braces will be executed, and therefore the output will be printed as shown above. If the condition specified in the ‘if’ statement is not true then the output is ‘bye’.

If... else statement

The ‘if else’ statement consists of two blocks of code. One is ‘if’ block which contains the code to be executed when the condition specified is true, and ‘else’ block of code gets executed when the condition specified is false.

The general form of ‘if else’ statement:

```
if(condition)
{
Code to be executed if the condition is true;
}
 else
{
Code to be executed if the condition is false;
}
```

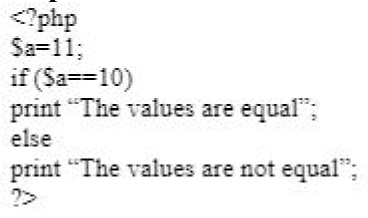
Example 3.26 Uses ‘if... else’ statement:

```
<? php
$a=10;
if ($a == 10)
    print(“The values are equal”);
else
    print(“The values are not equal”);
?>
```

Output is

```
The values are equal
```

Example 3.27 ‘If’ statement:



```
<?php
$a=11;
if ($a==10)
print “The values are equal”;
else
print “The values are not equal”;
?>
```

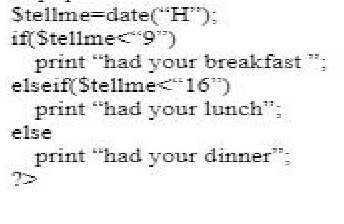
The output generated by this program is: The values are not equal

if...elseif.... else statement

The ‘if else’ statement consists of three blocks of code. One is ‘if’ block which contains the code to be executed when the condition specified is true and ‘elseif’ block of code gets executed when the condition specified is true and ‘if that is not else’ block is executed. The general form of ‘if elseif else’ statement is:

```
if(condition)
{
Code to be executed if the condition is true;
}
elseif(condition)
        {
        Code-to-beexecuted if it’s true;
        }
        else
        {
         Code to be executed if the condition is false;
        }
```

Example 3.28 ‘If...elseif.... else statement:



```
<?php
$tellme=date(“H”);
if($tellme<“9”)
  print “had your breakfast ”;
elseif($tellme<“16”)
  print “had your lunch”;
else
  print “had your dinner”;
?>
```

The output generated by this program is: had your lunch

Ternary Operators

In addition to the traditional 'if...else' statement, there is a shorthand 'if...else' using ternary operators (?:). The general form of 'if...else' statement using ternary operators is as follows:

(Condition? result (when condition is true): result (when condition is false)).

Example 3.29 Ternary operators:

```
$a = ((5>2)? 5:2);
```

The important point to remember is that if the blocks include single command or a statement, it is good enough to use shorthand 'if...else' using ternary operator. For example, if more than a single command or a statement have to be written in blocks, then traditional way of using 'if...else' statement works out well.

Following are the advantages of using ternary logic

- Code becomes shorter
- The code can be maintained easily and quickly
- It is effective for variable initialization when used properly
- Logic of 'if...else' statement is made in-line with output

The condition in the 'if' statement may contain two expressions or statements.

Example 3.30 Two expressions in the 'if' condition:

```
if($age >=22 && $age <=28)
        echo "you are eligible for government job";
else
        echo "you are not eligible for government job";
```

In these cases the evaluation of 'if' is done from left to right. The two expressions in the 'if' statement are connected by using logical AND (&&) operator whose resultant expression will be true, if and only if, both the expressions evaluates to true (as per the logic of 'logical AND operator'). So, in the Example 3.23, if the variable age specified is greater than 22 and also less than 28, then only the 'if' block of code will be executed. Therefore, the output 'you are eligible for government job' is displayed. The 'else' block of code in the above sample program will be executed if any one of the expression specified in the condition evaluates to false.

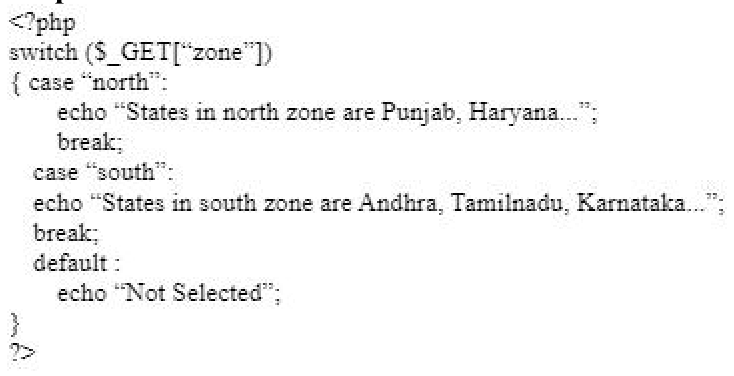
Switch Statement

There may be cases for a program to print different values for different conditions by checking a single instance or a single input. The long block of 'if...else' or simple 'if' statements will help us to code this type of program. But this is not the best approach as it will be difficult to code and understand. The switch statement is the best and efficient way to code this type of programs. A switch is a speedy checking process. In simple words, we can also say that 'switch' statement is more useful when there is a need to test a single variable for different values and will execute some action for each different value if and only if it is matched with that variable. The PHP switch statement will execute faster than multiple ‘elseif’. The general form of switch statement is:

```
switch(n) {
Case label1: Code to be executed if n=label1;
             break;
Case label2: Code to be executed if n=label2;
             break;
       .
       .
       .
default    : Code to be executed if n is different from both label1 and label2;
 }
```

The input expression or the condition will be matched with every case provided in the switch block and for the case which is matched with that particular condition provided the corresponding action will be displayed as output. If no case is matched, the action provided with the default case will be executed. Note, that the default case is optional. If the default case is not existed, the statements will undergo execution till the end of switch. Remember, that ‘default’ is a special keyword, where the word case will not come before that word. The ‘break’ should be provided at the end of each case. Placing ‘break’ at the end of each case prevents the code execution from the very next case after the case which is matched.

Example 3.31 Switch statement:



```
<?php
switch ($_GET["zone"])
{ case "north":
     echo "States in north zone are Punjab, Haryana...";
     break;
  case "south":
  echo "States in south zone are Andhra, Tamilnadu, Karnataka...";
  break;
  default :
     echo "Not Selected";
}
?>
```

Type this URL in the browser ‘ http://localhost/index.php?zone=south ’

The output generated by this program is:

```
States in south zone are Andhra, Tamilnadu, Karnataka...
```

Type this URL in the browser ‘ http://localhost/index.php?zone=north ’

The output generated by this program is:

```
States in north zone are Punjab, Haryana...
```

3.8 PHP LOOPS

Looping is a process of executing a block of statements repeatedly until some condition becomes false or until some condition is met. Some programs may require a group of statements need to be executed a particular number of times. In those cases; instead of writing that group of statements several times in a program, loop concept in PHP can be used to save time and space.

A block can be understood as a group of statements which are written between the opening brace "{" and the closing brace "}". If a block is going to be executed by some particular number of times it can be treated as a loop. Following are the different types of PHP loops:

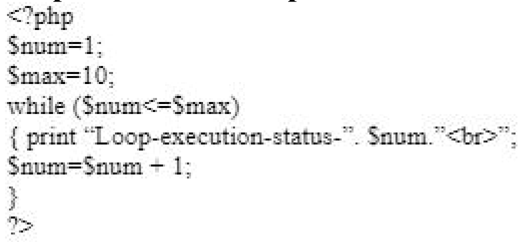
- while loop
- do-while loop
- for loop
- for each loop
- break statement
- continue statement

3.8.1 while Loop

While loop is the simplest type of loop in PHP. The function of the while loop is to do a job over and over with the understanding that the specified conditional statement is true. While loop is used to execute a group or a block of statements repeatedly (called loop) until the expression or condition specified in the while becomes false. This logical check is same as it is done in the 'if' statement to determine if it is true or untrue. The general form of while loop is:

```
while (conditional statement is always true)
{
//perform this code;
}
```

Example 3.32 while loop:



```
<?php
$num=1;
$max=10;
while ($num<=$max)
{ print "Loop-execution-status-". $num."<br>";
$num=$num + 1;
}
?>
```

The output generated by this program is:

```
Loop-execution- status-1
Loop-execution-status-2
Loop-execution-status-3
Loop-execution-status-4
Loop-execution-status-5
Loop-execution-status-6
Loop-execution-status-7
Loop-execution-status-8
Loop-execution-status-9
Loop-execution-status-10
```

Example 3.33 while loop:

```
<?php
$num=12;
$max=9;
while ($num<=$max){
print "Loop execution status-".$num."<br>";
$num=$num + 1;
}
```

For the above script no output will be displayed.

3.8.2. do-while Loop

A 'do-while' loop is an improved form of while loop. In case of while loop, the conditional statement is checked, if the condition is true then the code inside the while loop is executed and if the conditional statement is untrue then the code inside the loop is not executed. A do-while loop always performs its block of code at least once. This happens due to not testing the conditional statement until the contained code has been executed. The general form of do-while loop is:

```
do
{
//do this code;
} while (conditional statement);
```

Example 3.34 do-while loop

```
<?php
$num=12;
$max=9;
do {
print "Loop execution status-".$num."<br>";
$num=$num + 1;
}while($num<=$max);
?>
```

The output generated by this program is:

```
Loop execution status-12
```

3.8.3. for Loop

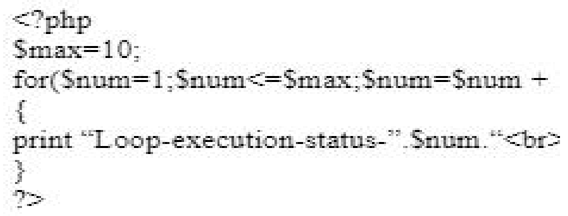
For loop is modestly a while loop with a more code joined to it. The for loop permits one to define the following steps in one line of code.

- Allot a counter variable to the initial value which you require.
- Find, if the conditional statement is always true.
- Execute the script within the loop.
- Increment the counter variable at the completion of each iteration through the loop

The general form of for loop is:

```
for (initialise a counter; conditional statement; increment the counter)
{
  perform this code;
}
```

Example 3.35 for loop



```
<?php
$max=10;
for($num=1;$num<=$max;$num=$num +
{
print "Loop-execution-status-".$num."<br>
}
?>
```

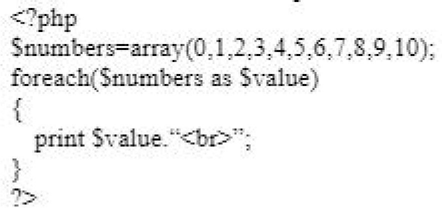
The output generated by this program is:

```
Loop-execution- status-1
Loop-execution-status-2
Loop-execution-status-3
Loop-execution-status-4
Loop-execution-status-5
Loop-execution-status-6
Loop-execution-status-7
Loop-execution-status-8
Loop-execution-status-9
Loop-execution-status-10
```

3.8.4. foreach loop

PHP provides an easier way to use each element of an array using the 'foreach' statement. For each item in the specified array executes this code. In case of while loop and for loop, will work till some condition fails. The foreach loop will remain till it has gone through every item in the array. The operator "=>" denotes the association between a key and value.

Example 3.36 foreach loop:



```
<?php
$numbers=array(0,1,2,3,4,5,6,7,8,9,10);
foreach($numbers as $value)
{
  print $value."<br>";
}
?>
```

The output generated by this program is:

```
0
1
2
3
4
5
6
7
8
9
10
```

3.8.5 Break Statement

Sometimes, a situation arises where we want to exit from a loop without waiting to go back to the conditional statement. The ‘break’ keyword ends execution of foreach , while , for, do-while or switch structure. When the ‘break’ keyword executed exclusively for a loop, then the control automatically passes to the first statement outside the loop. A break is usually linked with the ‘ if ’.

Example 3.37 Break statement:

```
<?php
for ($count = 0; $count < 10; $count++)
{
$randomNumber = rand (1,100);
if ($randomNumber < 20)
break;
else
echo “Number greater than 20: $randomNumber<br />”;
}
echo “Number less than 20: $randomNumber<br />”;
?>
```

The output generated by this program is:

```
Number greater than 20: 40
Number less than 20: 14
```

3.8.6. continue Statement

Sometimes, a situation arises where we want to take the control of the program to the beginning of the loop (for example, do-while, for, while, etc.) by skipping the rest of the statements inside the loop which have not been executed. Then we use the keyword ‘continue’ for that purpose. When the keyword continues executing inner side of a loop, the control automatically passes to the starting of the loop. Continue is usually associated with the ‘if’ statement.

Example 3.38 continue statement:

```
<?php
for ($count = 0; $count < 10; $count++)
{
if ($count %2)
continue;
echo “Number : $count is even number <br />”;
}
?>
```

The output generated by this program is:

```
Number : 0 is even number
Number : 2 is even number
Number : 4 is even number
Number : 6 is even number
Number : 8 is even number .
```

3.8.7. Jumping or Branching Statements

<u>goto Statement</u>

Goto label is used to jump code execution to a different set of code in a program. The goto statement requires a label to identify the place where the branch to be made. A label is any valid variable name and must be followed by a semicolon (;). A label is immediately placed before the statement where the control is to be transferred. The general form of goto statement is:

```
<?php
 goto label;
 ?>
```

Example 3.39 goto branching statement:

```
<?php
for ($count = 0; $count < 10; $count++)
{
$randomNumber = rand (1,100);
if ($randomNumber < 20)
goto less;
else
echo "Number greater than 20: $randomNumber<br />";
}
less:
echo "Number less than 20: $randomNumber<br />";
?>
```

The output generated by this program is:

```
Number greater than 20: 60
Number greater than 20: 43
Number greater than 20: 97
Number greater than 20: 35
Number greater than 20: 100
Number less  than 20: 5
```

3.9. Advanced PHP

A PHP function is a block of script that can be called from anywhere in the program once it is declared. Also, we can say that the block of statements that can be executed whenever and wherever we need, can be called a function. In simple words, a function is a piece of code in the large program which performs a particular task. Sometimes, we need a group of statements to be executed at different instances in a program, in that case, instead of writing the same group of statements at different places in a program, we will group that statements as a block give a name. So, whenever we need that code to be executed in our program, we simply call that block of code using function name. Code in the function is ignored till that function is called from somewhere in the program. A function can be called by using the function name followed by parenthesis.

3.9.1. Advantages of Using Functions

- Reduces code duplication
- Improves and increases the code clarity
- Provides reusability of code
- Improves readability and provides flexibility
- Maintenance will become easy
- Division of complex problems into simple problems
- Debugging and testing will become easy
- Easy to understand the code involved in the function
- Recursion is possible

3.9.2 Types of Functions

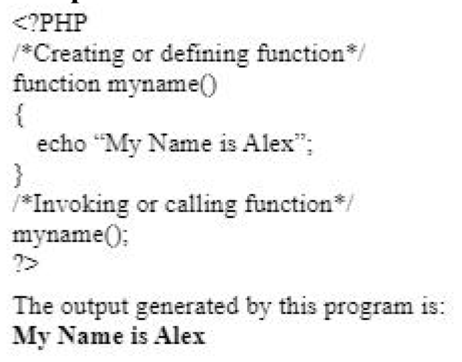
There are mainly two types of functions, namely, built-in functions, and user-defined functions. The functions which are a part of the PHP language comes under built-in functions. Examples of built-in functions are round(), abs(), etc. The functions which are defined by programmers to perform some specific task as per their requirement are called user-defined functions. User-defined functions are created by using the keyword function.

3.9.3 Creating a PHP Function

A function will be executed by a call to the function. The general form of a PHP function is:

```
function functionName()
{
code to be executed;
}
```

Example 3.40 A PHP function:



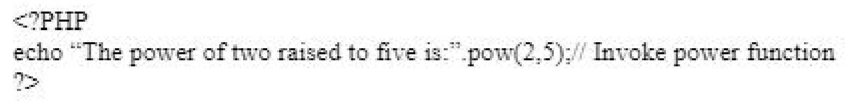
```
<?PHP
/*Creating or defining function*/
function myname()
{
  echo "My Name is Alex";
}
/*Invoking or calling function*/
myname();
?>
```

The output generated by this program is:
My Name is Alex

3.9.4 Invoking a Function

The function must be declared before they are called. By declaring the function, we are creating a prototype of that function. Therefore, the compiler comes to know about the presence of functions. A function call is a function that is called within another function. Invoking a function is nothing, but calling a function by its name. We have two ways to invoke a function. First one is calling by value and the second one is call by reference.

Example 3.41 invoking a function:



```
<?PHP
echo "The power of two raised to five is:".pow(2,5);// Invoke power function
?>
```

The output generated by this program is:
The power of two rose to five is: 32

Call by Value

Call by value means passing the value directly to a function. By default, the parameters of a function can be handled by call by value. To handle the parameters, copy operation is used. If we change the parameter, then the original variable passed to the function will not change. The called function uses the value in a local variable; any changes to it do not affect the source variable. The general form of a call by value is:

```
function by_value($a, $b){      }
```

Example 3.42 Call by value:

```
<?PHP
/*Creating or defining function*/
function yournameis($value)
{
   echo "Your Name is" .$value;
}
/*Invoking or calling function*/
yournameis("Alex");
?>
```

The output generated by this program is:

```
Your Name is Alex
```

Example 3.43 Call by value:

```
<?PHP
/*Creating or defining function*/
function yournameis($value,$age)
{
   echo "Your Name is ".$value." and age is" .$age;
}
/*Invoking or calling function*/
yournameis("Alex",19);
?>
```

The output generated by this program is:

```
Your Name is Alex and age is 19
```

Call by Reference

Call by reference means passing the address of a variable where the actual value is stored. The way of handling parameters is altered in call by reference. A reference is passed as an alias variable. Therefore, altering it will not affect the local copy, but it will affect the original variable. In place of copying the entire data, e.g. objects, call by reference hold back a lot of storage space. The general form of a call by reference is:

```
function by_reference(&$a, &$b){}
```

Example 3.44 Call by reference:

```
<?PHP
/*Creating or defining function*/
function isyourname(&$value)
{
   $value="No not Alex";
}
$value="Is your name Alex";
echo $value."<br>"; //displaying before calling
/*Invoking or calling function*/
isyourname($value);
echo $value; //displaying after calling
?>
```

The output generated by this program is:

```
Is your name Alex
No not Alex
```

3.10 Returning Values

The return statement is used to return the values and the statement is optional. Any type can be returned, including arrays and objects. This return statement causes the function to end its execution instantaneously. Then the control is passed back to the line from where it was called. We can return a value using the return keyword. A function may or may not return a value.

Example 3.45 Return statement:

```
<?PHP
/*Creating or defining function*/
function yourname()
{
  return "My Name is Alex";
}
/*Invoking or calling function*/
echo yourname();
?>
```

The output generated by this program is:
My Name is Alex

3.11. Recursive Functions

Recursive function is a method of defining functions in which the function being defined is applied within its own definition. A recursive function calls itself to do its job. Recursion is a widely used approach to solve many programming tasks.

Example 3.46 Recursive function:

```
<?PHP
function fact($n)
{
  if ($n === 0) {
  return 1;
  }
  else {
  return $n * fact($n-1); // <--calling itself.
  }
}
echo fact(5);
?>
```

The output generated by this program is:

```
120
```

Function Libraries

Functions relate to a great way of reusing the existing code. They are often collectively assembled into libraries and subsequently reused for similar applications. The libraries in PHP are formed via simple aggregation of function definitions in a single file.

Example 3.47 Function libraries:

```
<?php
     function weather($temperature, $pressure)
     {
        function body here
```

```
         }
        function schedule($week,$day, $time)
        {
          // function body here
        }
        function season($seasontype $year)
        {
         // function body here
         }
?>
```

Let us save this function with climate as it will denote the purpose, name as climate.library.PHP'. However, this file has to be saved in the server root with an extension that would affect the web server to permit the file contents unparsed. This will give the possibility for a user to call the file from the browser and review the code, which contains sensitive data. Assume that you have titled this library 'climate.Library.PHP'.

Example 3.48 Script for including functions:

```
<?PHP
require_once("climate.library.PHP");
...
?>
```

The functions included in this library can be called whenever the function is required.

3.12 Get and Post Methods

To send the information from the client to web server we need mainly two methods, Namely, GET and POST. The method parameter in HTML form may use either 'GET' or 'POST' method. There are two variables in PHP which works with this parameter 'method' in HTML form. Those variables are $_GET and $_POST. The 'method' parameter in an HTML form is used to 'GET' all the form variables which can be accessed by using the variable $_GET. In the same way, the 'method' parameter is used to 'POST' all the form variables which can be accessed by using the variable $_POST.

Example 3.49 'GET' method and '$_GET' variable

```
//Consider the following script and save as explget.PHP.
<html>
<head>
<body>
<title> explanation for get method</title>
</head>
<form action ="getexp.PHP" method="get">
Enter your department name : <input name ="deptname" type="text"/>
<input type="submit" name ="submit" value="click this"/>
</form>
</body>
</html>
```

After executing the above form and submitting some text , if the user clicks on the submit button (the user clicks on "clicks this") the URL seems to be http://localhost/getexp.PHP?department=cse. The code regarding the file getexp.PHP:

```
    <?PHP
    $deptname=$_GET['deptname']; //retrieves deptname from web form submitted using GET
method
    echo $deptname;
    ?>
```

The file 'getexp.PHP' will use the $_GET variable to retrieve form data, and therefore the output 'cse' will be displayed.

Points to Remember

- While Using 'GET' Method It is not good for very large variable values (should not exceed 2000 characters).
- It is not secure since the form data (form variables) gets embedded in the URL.
- Should not be used while passing sensitive or secured information.
- Using this method, it is possible to transfer limited data.
- It is used to send only text data not binary data.

Explanation of 'POST' Method and '$_POST' Variable

The major disadvantage of 'GET' method is that the form data gets appended to the URL leading to insecurity of data. To avoid this, 'POST' method will be used to hide the form data submitted and the form data will not be viewed at the URL.

Example 3.50 'POST' method and '$_POST' variable

```
//Consider the following program and save as explpost.PHP.
<html>
<head>
<body>
<title> explanation for get method</title>
</head>
<form action ="postexp.PHP" method="post">
Enter your department name : <input name ="deptname" type="text"/>
<input type="submit" name ="submit" value="click this"/>
</form>
</body>
</html>
```

After executing the above form and submitting some text, if the user clicks on the submit button (the user clicks on 'click this') the URL seems to be http://localhost/postexp.PHP. The code regarding the file "postexp.PHP":

```
<?PHP
$deptname=$_POST['deptname'];
//Retrieves deptname from web form submitted using the POST method
echo $deptname;
?>
```

The file 'postexp.PHP' will use the $_POST variable to retrieve form data. Therefore, the output 'cse' will be displayed.

Points to Remember While Using ‘POST’ Method

- It is good for very large variable values.
- It is secure since the form data (form variables) are not embedded in the URL.
- This ‘POST’ method is more suitable while passing sensitive or secured information.
- This method is useful to transfer large amount of data as there is no restriction on data size.
- Using this method, it is possible to send text data as well as binary data.

3.13 ARRAYS IN PHP

An array is a single variable which is used to store more than one value (belonging to one category) at a time. Arrays are more useful when there is a need to work with large amount of data, such as records from a database or group of similar data together. In PHP, array will be created by using array () function. The general form of an array is:

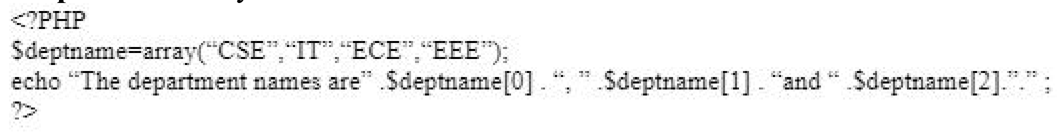
```
$arrayname=array (values);
```

In case, if you want to accumulate 100 numbers, then instead of declaring 100 variables, it is easy to define an array of 100 length.

Uses of Arrays

- Easy to manipulate the array related data.
- Able to work with many values at a time.
- Array-related functions makes the work very easy whenever required.

Example 3.51 Arrays:



```
<?PHP
$deptname=array(“CSE”,“IT”,“ECE”,“EEE”);
echo “The department names are” .$deptname[0] . “, ” .$deptname[1] . “and “ .$deptname[2].”.” ;
?>
```

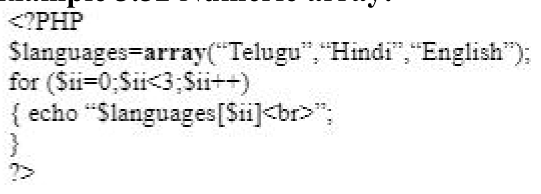
3.13.1. Different Kinds of Arrays

<u>**Numeric (or) Indexed Array**</u>

In numeric (or) indexed array, linear fashion of storing and retrieval of values is observed. Numeric indexing is recognised in the numeric array. The general form of a numeric array is:

```
array(value1,value2,value3);
```

Example 3.52 Numeric array:



```
<?PHP
$languages=array(“Telugu”,“Hindi”,“English”);
for ($ii=0;$ii<3;$ii++)
{ echo “$languages[$ii]<br>”;
}
?>
```

The output generated by this program is:

```
Telugu
```

Hindi
English

Associative Array

An associative array contains strings as an index. This type of array stores element values in association with key values relatively than in a strict linear index order. The general form of an associative array is:

array(key=>value,key=>value,key=>value,......);

Example 3.53 An associative array:

```
    <?PHP
    $languages=array("Mother      Tongue"=>"Telugu","National      language"=>"Hindi","Foreign
language"=>"English");
    echo "Mother Tongue is ".$languages["Mother Tung"]."<br>";
    echo "National language is ".$languages["National language"]."<br>";
    echo "Foreign language is ".$languages["Foreign language"]."<br>"
    ?>
```

The output generated by this program is:

```
Mother Tongue is Telugu
National language is Hindi
Foreign language is English
```

Multidimensional Array

A multidimensional array may contain one or more arrays and values is accessed using multiple indices.

Example 3.54 Multidimensional array:

```
<?PHP
$mult=array(
     array(1,2,3),
     array(4,5,6),
     array(7,8,9)
);
for ($ii=0;$ii<3;$ii++)
{
for ($jj=0;$jj<3;$jj++)
   {
     echo $mult[$ii][$jj]." ";
   }
     echo "<br>";
}
?>
```

The output generated by this program is:

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

3.13.2. Programs Using Arrays

Adding Values to the Array at the End

```
<?PHP
$states=array("Andhra Pradesh","Tamilnadu","Karnataka");
array_push($states,"kerala","Goa");
foreach ($states as $value)
  echo "$value<br>"
?>
```

The output generated by this program is:

```
Andhra Pradesh
Tamilnadu
Karnataka
Kerala
Goa
```

Adding Values to Array at First

```
<?PHP
$states=array("Andhra Pradesh","Tamilnadu","Karnataka");
array_unshift($states,"kerala","Goa");
foreach ($states as $value)
echo "$value<br>";
?>
```

The output generated by this program is:

```
Kerala
Goa
Andhra Pradesh
Tamilnadu
Karnataka
```

Removing Values from Array Front

```
<?PHP
$states=array("Andhra Pradesh","Tamilnadu","Karnataka");
$front=array_shift($states);
foreach ($states as $value)
  echo "$value<br>";
echo "Removed value :$front";
?>
```

The output generated by this program is:

```
Tamilnadu
Karnataka
Removed value: Andhra Pradesh
```

Remove Values from the End

```
<?PHP
$states=array("Andhra Pradesh","Tamilnadu","Karnataka");
$front=array_pop($states);
foreach ($states as $value)
  Echo "$value<bar>";
echo "Removed value :$front";
?>
```

The output generated by this program is:

```
Andhra Pradesh
Tamilnadu
Removed value: Karnataka
```

Locating Value in the Array

```
<?PHP
$states=array("Andhra Pradesh","Tamilnadu","Karnataka");


 if(in_array("Karnataka",$states))
 echo "The state Karnataka is present in array";
 ?>
```

The output generated by this program is:

```
The state Karnataka is present in the array
```

Retrieving Data from the Array

```
<?PHP
$states=array("Andhra Pradesh","Tamilnadu","Karnataka");
while($state=current($states))
{
echo "$state<br>";
next($states);
}
?>
```

The output generated by this program is:

```
Andhra Pradesh
Tamilnadu
Karnataka
```

Sorting Array in Ascending Order

```
<?PHP
$states=array("Andhra Pradesh","Tamilnadu","Karnataka");
$sort=sort($states);
while($state=current($states))
{
echo "$state<br>";
next($states);
}
?>
```

The output generated by this program is:

```
Andhra Pradesh
Karnataka
Tamilnadu
```

Sorting Array in Descending Order

```
<?PHP
$states=array("Andhra Pradesh","Tamilnadu","Karnataka");
$sort=rsort($states);
while($state=current($states))
{
echo "$state<br>";
next($states);
}
?>
```

The output generated by this program is:

```
Tamilnadu
Karnataka
Andhra Pradesh
```

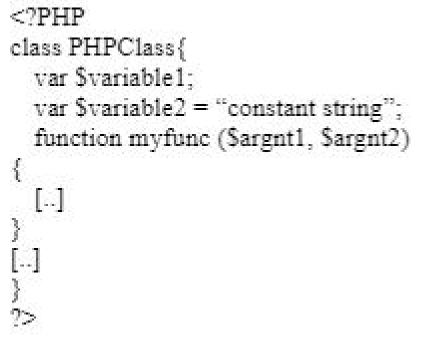
3.14. OBJECT ORIENTED CONCEPTS

Object Oriented Programming (OOP) is a programming paradigm that represents concepts as 'objects'. Objects have data fields (attributes that describe the object) and associated procedures known as methods.

Class

This is a programmer-defined data type. Class can be considered as a template which holds many instances of the similar kind of the object.

Example 3.55 Creation of a class:

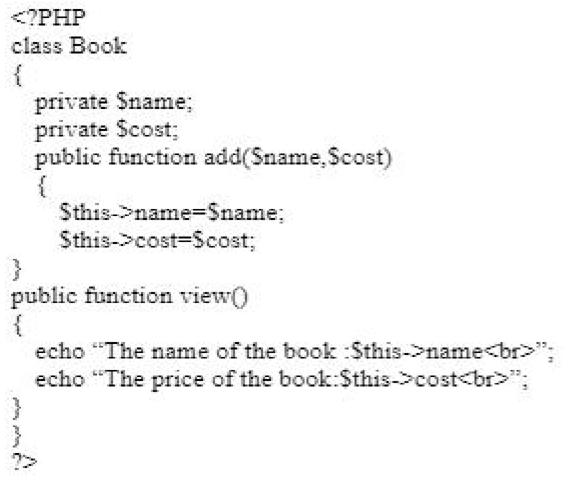


```
<?PHP
class PHPClass{
  var $variable1;
  var $variable2 = “constant string”;
  function myfunc ($argnt1, $argnt2)
{
  [..]
}
[..]
}
?>
```

Explanation for each line of the above program:

- The keyword class, trailed by the name of the class which you want to define.
- In braces set ‘{}’ variable declarations and functions are declared.
- Variable declarations start with the keyword var, followed by $ variable name.
- They can also be assigned initially to a constant value.
- Function definitions are same as standalone PHP functions, but they are local to the class and can be used to set and access the data of the object.

Example 3.56 Creation of a class in PHP:



```
<?PHP
class Book
{
  private $name;
  private $cost;
  public function add($name,$cost)
   {
     $this->name=$name;
     $this->cost=$cost;
}
public function view()
{
  echo “The name of the book :$this->name<br>”;
  echo “The price of the book:$this->cost<br>”;
}
}
?>
```

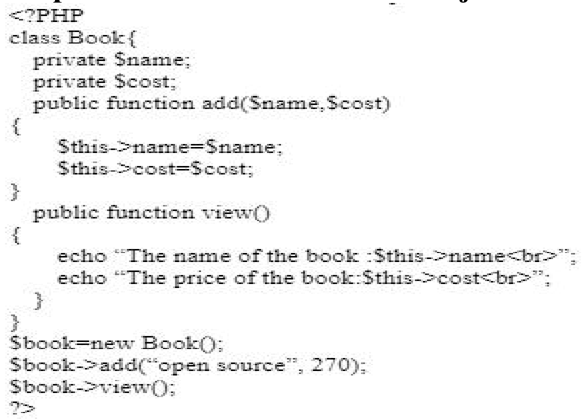
Object

An object is a single instance of the data structure defined by a class. We can define a class once and can create several numbers of objects that belong to a class. An object is also known as an instance of a class. After defining the class several numbers of objects can be created as per the need.

Example 3.57 Creation of an object using the new operator in PHP:

$monday = new day;
$tuesday = new day;
$wednesday = new day;

Example 3.58 Instantiation of an object:



```
<?PHP
class Book{
  private $name;
  private $cost;
  public function add($name,$cost)
{
    $this->name=$name;
    $this->cost=$cost;
}
  public function view()
{
    echo "The name of the book :$this->name<br>";
    echo "The price of the book:$this->cost<br>";
  }
}
$book=new Book();
$book->add("open source", 270);
$book->view();
?>
```

The output generated by this program is:

```
The name of the book: open source
The price of the book: 270
```

Member Variable

The variables defined inside a class are known as member variables . This data can be accessed via member functions and are invisible outside of the class. Once an object is created, these variables are called attributes of that object .

Member Function

To access the data object, we define a function inside a class which is called member function.

Inheritance

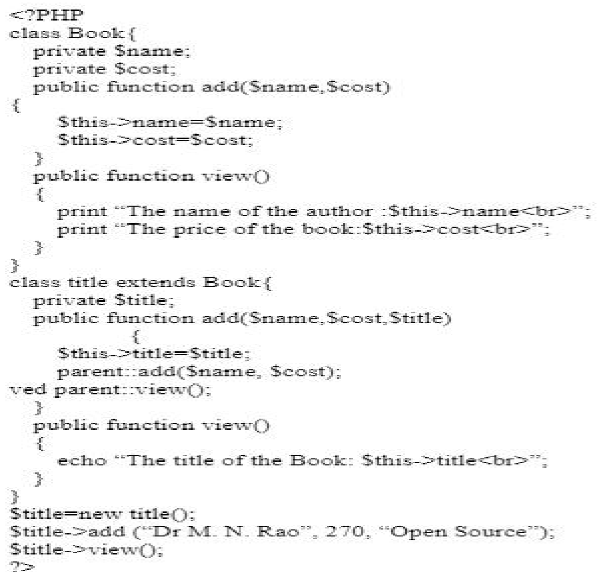
Defining a new class by inheriting the existing functions of a parent class is known as inheritance. There are two types of classes:

Parent class: A new class that is inherited from the existing classes is known as parent class. The existing class is known as a base class or super class.

Child class: A class that is derived from another class is known as child class. This derived class is also known as a subclass or derived class . The derived class will take over the entire or few member functions and variables of a base class. By using extends clause, PHP class defines the inherit property from a parent class. The general form of an inheritance is:

```
class Child extends Parent
{
  <definition body>
}
```

Example 3.59 Inheritance:



```
<?PHP
class Book{
   private $name;
   private $cost;
   public function add($name,$cost)
{
      $this->name=$name;
      $this->cost=$cost;
   }
   public function view()
   {
      print "The name of the author :$this->name<br>";
      print "The price of the book:$this->cost<br>";
   }
}
class title extends Book{
   private $title;
   public function add($name,$cost,$title)
            {
      $this->title=$title;
      parent::add($name, $cost);
ved parent::view();
   }
   public function view()
   {
      echo "The title of the Book: $this->title<br>";
   }
}
$title=new title();
$title->add ("Dr M. N. Rao", 270, "Open Source");
$title->view();
?>
```

The output generated by this program is:

```
The name of the author: Dr.M.N. Rao
The price of the book: 270
The title of the Book: Open Source
```

Open Source Polymorphism

When the same function is used for different purposes in an object oriented concept, then it is called polymorphism. For example, the name of the function will remain same, but it may take different number of arguments and can perform a variety of tasks.

Overloading

Overloading is type of polymorphism. Depending on the types of their arguments some or all operators can have different implementations. One can observe the overloading of functions with distinct implementations.

Data Hiding

The implementation details are hidden. An abstract class can be inherited, but cannot be instantiated. The abstract class is declared with the keyword 'abstract'. The general form of the abstract class:

```
abstract class MyAbstractClass
{
        abstract function myAbstractFunction()
        { }
 }
```

Example 3.60 Data Hiding property:

```
<?PHP
abstract class AbstractClass
{
      abstract protected function prefixName($name);
}
class salutation extends AbstractClass
{
      public function prefixName($name, $separator = ".")
   {
      if ($name == "Ramu")
   {
      $pr]efix = "Mr";
   } elseif ($name == "Sitha")
   {
      $prefix = "Mrs";
   } else {
      $prefix = "";
         }
      return "{$prefix}{$separator} {$name}<br>";
   }
}
$class = new salutation;
echo $class->prefixName("Ramu"), "\n";
echo $class->prefixName("Sitha"), "\n";
?>
```

The output generated by this program is:

```
Mr. Ramu
Mrs. Sitha
```

Encapsulation

The concept of encapsulating all the data and its member functions together to form an object is referred as Encapsulation.

Constructor

Whenever an object is created, it automatically calls for constructor functions. To define a constructor, PHP provides function __construct (). One can pass any number of arguments into the constructor function.

Example 3.61 Constructor

```
<?PHP
class Book{
   private $name;
   private $cost;
   function __construct($name,$cost)
   {
      $this->add($name,$cost);
      $this->view();
   }
      public function add($name,$cost)
   {
      $this->name=$name;
      $this->cost=$cost;
   }
   public function view()
   {
      print "The name of the book :$this->name<br>";
      print "The price of the book:$this->cost<br>";
   }
}
$book=new Book("open source", 270);
?>
```

The output generated by this program is:

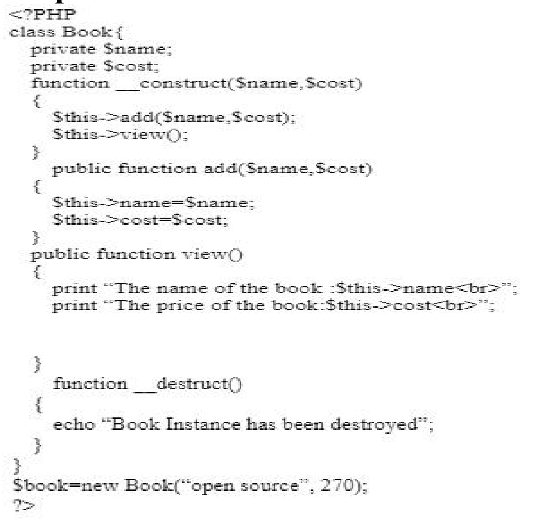
```
The name of the book: open source
```

The price of the book: 270

Destructors

Whenever an object is removed or out of scope, it automatically invokes the destructor function. A destructor function can be defined using function __destruct (). All the resources within a destructor are released.

Example 3.62 Destructor:



```
<?PHP
class Book{
   private $name;
   private $cost;
   function __construct($name,$cost)
   {
      $this->add($name,$cost);
      $this->view();
   }
      public function add($name,$cost)
   {
      $this->name=$name;
      $this->cost=$cost;
   }
   public function view()
   {
      print "The name of the book :$this->name<br>";
      print "The price of the book:$this->cost<br>";


   }
      function __destruct()
   {
      echo "Book Instance has been destroyed";
   }
}
$book=new Book("open source", 270);
?>
```

The output generated by this program is:
The name of the book: open source
The price of the book: 270
Book Instance has been destroyed

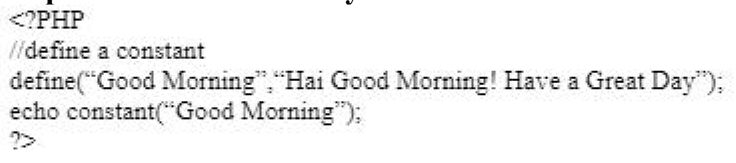
Function Overriding

Child class function definitions can override definitions of parent classes which are having the same name in the parent class. In a child class, the function definition inherited from parent class can be modified.

3.15 CONSTANTS

A constant is a variable that holds a value, but it is similar to a function because a constant is immutable. A constant value does not change.

Example 3.63 A constant keyword:



```
<?PHP
//define a constant
define("Good Morning","Hai Good Morning! Have a Great Day");
echo constant("Good Morning");
?>
```

The output generated by this program is: Hai Good Morning! Have a Great Day

Static Keyword

Whenever class members or methods of a class are declared as static, it means that they are accessible without any creation of object to the class. A static member cannot be accessed with an instantiated class object. For example static int a=10.

Example 3.64 Static keyword:

```
<?PHP
class Count
{
private static $counter = 0;
function __construct()
{
self::$counter++;
}
static function getCount()
{
return self::$counter;
}
}

  // Instantiate the Count class.
  $c1 = new Count();
  echo Count::getCount().“<br />”;
  // Instantiate another Count class.
  $c2 = new Count();
  echo Count::getCount().“<br />”;
  ?>
```

The output generated by this program is: 1 2

<u>**Final Keyword**</u>

The final keyword in PHP 5 prevents child classes from method overriding by defining the class with final keyword. If the class defined itself as final then it cannot be extended.

3.16 STRINGS

The general sequence of characters enclosed in double quotes can be defined as a string . A single letter enclosed in a single quote can be called a character. A string in PHP can be defined as a series of characters, whereas a character is the same as a byte since PHP supports only 256 character set. Strings can be easily converted into numbers and vice-versa in PHP, because of the weakly typed feature. The string can be created by declaring a string variable. The string variable value must be some text enclosed in double quotes. The general form of a string is:

$strname = “string value”; (strname is string variable)

Example 3.65 String:

```
<?PHP
$str1 = “good morning to all.”;
echo $str1
?>
```

The output generated by this program is: good morning to all.

While printing a string, the string variable can be enclosed in double quotes also, which makes no difference in the output. It means that, in the third statement of Example 3.69, the string variable can be written in double quotes. Please note that echoes $str1 can be also rewritten as echo “$str1”.

3.16.1 Creation of a String in Four Ways

<u>Single Quote</u>

This is the simplest way to define a string by enclosing it in single quotes. Every variable and character enclosed by using single quotes is treated literally except in the concept of escape characters.

<u>Double Quotes</u>

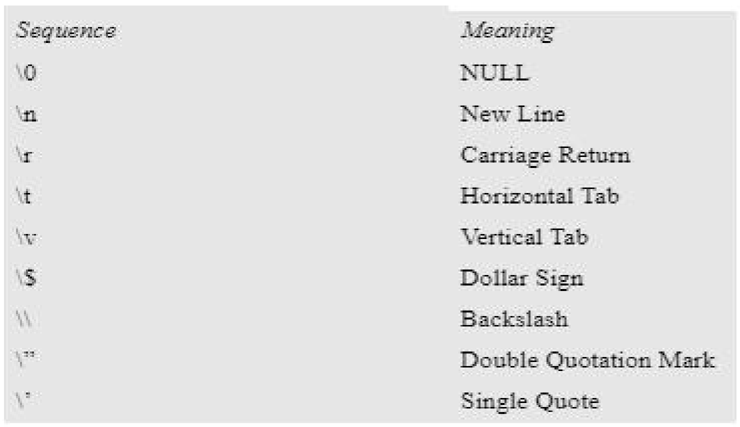
This is the primary way to define a string. Strings which are delimited by double quotes are processed by two ways in PHP, namely, character sequences, and escape sequences.

<u>Character sequences:</u>

Character sequences starting with black slash(\) are replaced by special characters. Variable names starting with $ are replaced with their values. Escape sequences (or) Escape characters:

<u>Escape sequences</u> are the combination of escape character(\) and a letter which means that the character or the letter after the escape character must be treated specially. Table shows the list of sequences.

Table 3.10 Escape sequence in PHP



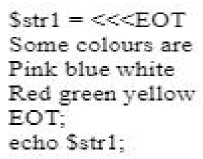
Sequence	*Meaning*
\0	NULL
\n	New Line
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab
\$	Dollar Sign
\\	Backslash
\"	Double Quotation Mark
\'	Single Quote

<u>Heredoc</u>

This way of defining a string helps the programmer to create multi line strings without using quotations. Defining a string using heredoc will be tricky and will create problems if we do not code the string properly. The general form of heredoc is:

```
$stringname = <<< identifier
Multiple lines of text
identifier;
```

<u>Example 3.66 Heredoc:</u>



```
$str1 = <<<EOT
Some colours are
Pink blue white
Red green yellow
EOT;
echo $str1;
```

Following rules are to be followed in Heredoc:

- It should always start with <<< followed by an identifier.

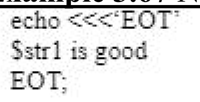
- Then the string should be included.
- The string creation should end with the same identifier followed by semicolons.
- Identifier names are case sensitive.
- The closing identifier must be in a separate line.
- The identifier should contain any alphanumeric character, underscores and must start with a non-numeric character or an underscore.

Nowdoc

This is same as heredoc, but a little bit difference in syntax is that the starting identifier must be enclosed in single quotes. The general form of nowdoc is:

```
$stringname = <<< 'identifier'
Multiple lines of text
identifier;
```

Example 3.67 Nowdoc



```
echo <<<'EOT'
$str1 is good
EOT;
```

The output generated by this program is:

```
$str1 is good
```

Note: If the above program is used with heredoc, $str1 will be replaced by the value assigned to that variable.

Following rules are to be followed in Nowdoc.

- It should always start with <<< followed by identifier enclosed by single quotes.
- Then the string should be included.
- The string creation should end with the same identifier followed by semicolons.
- The closing identifier should not be enclosed with single quotes.
- Identifier names are case sensitive.
- The closing identifier must be in a separate line.
- The identifier should contain any alphanumeric character, underscores and must start with a non-numeric character or an underscore.

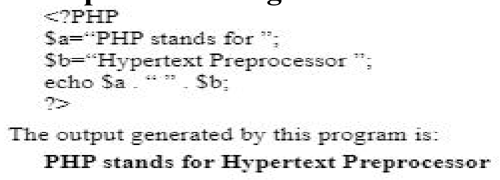
3.16.2 String Manipulation

The functions which allow to manipulate strings are string manipulation.

String Concatenation

A string is defined as any finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks). String concatenation is the operation of joining two character strings end-to-end. The concatenation operator (.) is used to join two string values together.

Example 3.68 String concatenation:



```
<?PHP
$a="PHP stands for ";
$b="Hypertext Preprocessor ";
echo $a . " " . $b;
?>
```

The output generated by this program is:

```
PHP stands for Hypertext Preprocessor
```

Trimming Functions:

The following table shows the list of string trimming functions.

Function	*Description*
trim()	It removes whitespaces at the beginning and at the end of a string.
ltrim()	It removes whitespace at the starting of a string.
rtrim()	It removes whitespace at the ending of a string.

Example 3.69 Trimming Functions

```
<?PHP
$a="\t HypertextPreprocessor \t";
echo ltrim($a);//remove spaces at the beginning only
echo rtrim($a);//remove spaces at the end only
echo trim($a);//remove spaces at the beginning and the end
?>
```

The output generated by this program is:

```
HypertextPreprocessor HypertextPreprocessorHypertextPreprocessor
```

Presentation Function

The following table shows the list of presentation functions.

Function	*Description*
htmlentities()	Escapes all the entities of HTML.
strtoupper()	Converts the given string to uppercase.
strtolower()	Changes the case of a string to lowercase.
ucfirst()	Changes the starting character of a string to uppercase.
ucwords()	Changes the starting character of every word in a string to uppercase.

Example 3.70 Presentation Function

```
<?PHP
$a="Open Source Programming Language";
echo strtolower($a)."<br>";
echo strtoupper($a)."<br>";
echo ucfirst($a)."<br>";
echo ucwords($a)."<br>";
?>
```

The output generated from this program is:

```
open source programming language
OPEN SOURCE PROGRAMMING LANGUAGE
Open Source Programming Language
Open Source Programming Language
```

Converting strings and Arrays

The following table shows the list of string manipulation operation functions.

Function	*Description*
substr(str,pos)	This function returns the substring from the character specified in position pos to the last of the string.
substr(str,-len)	The substring from len characters from the one end of the string to the other end of the string will be returned by this function.

substr(str,pos,len)	A substring starting with the character specified in position pos to len characters will be returned by this function.
substr(str,pos, -len)	Returns a substring starting with the character specified in position pos and removing off the last specified len characters of the string.
str_replace()	Replaces all occurrences of one string with different string.

Example 3.71 string manipulation operations

```
<?PHP
$str="PHP Hypertext Preprocessor";
$substr1=substr($str,4);
$substr2=substr($str,0,3);
echo $substr1."<br>";
echo $substr2;
?>
```

The output generated by this program is:

```
Hypertext Preprocessor
PHP
```

Comparing String

The following table shows the list of functions for comparing string.

Function	*Description*
strcmp()	Compares one string with another string. Returns < 0 if string1 is less than string2, > 0 if string1 is greater than string2, and 0 if both strings are equal.
strcasecmp()	Similar to strcmp(), but it is insensitive to the case.
strlen()	Returns the string length.

Example 3.72 Comparing String function

```
<?PHP
$a="PHP";
$b="HTML";
if(strcmp($a,$b)<0) echo "A string a is less than string b.";
else if(strcmp($a,$b)>0) echo "A string a is greater than string b.";
else echo "Both strings are equal.";
?>
```

The output generated by this program is:

```
A string a is greater than string b.
```

3.17. FILE HANDLING AND DATA STORAGE

The basic requirement for a programmer is to deal with files by performing several operations on it. Implementing several operations on files without any loss of data is known as file handling. Proper care should be taken while dealing with files, because if you do something wrong, it will lead to lots of damage. File handling is more useful to handle a small amount of data where the database is not that much preferred. PHP provides a lot of built-in functions to work with files.

3.17.1. Creating a File

The fopen function, to operate correctly requires two important pieces of information. First, the filename we want to open. Second, the job that we are going to do with that file (i.e., Read from a file, write to file, etc.). Since, we wish to create a new file, we have to supply the name of the file and tell to PHP that we need to write to the file

Example 3.73 File creation:

```
<?PHP
$fp=fopen("hello.txt","r");
if(!$fp)
{
echo "Unable to open the file";
exit;
}
$num=fread($fp,500);
echo "Content in the file: $num";
fclose($fp);
?>
```

The output generated from this program is:

```
Content in the file: PHP is Hypertext Preprocessor.
```

Originally, it is called *Personal Home Pages.*

3.17.2 Open/Close a File

The function fopen() opens the file as a 'stream' and PHP returns a 'handle' to the opened file. Therefore, it can be used as a reference in other functions. Every file opens in a specific mode. An opened file can be closed with fclose() function or when your script ends. The following table shows different modes to open a file.

Modes	*Description*
R	This is read only mode. Starts from the beginning line of the file.
r+	This is read/write mode. Starts from the first line of the file.
W	This is write only mode. Opens and deletes all the displaying contents of the file (or) it can create a new file if the specified file is not existing.
w+	This is read/write mode. Opens and deletes all the displaying contents of the file (or) it can create a new file if the specified file is not existing.
A	This is append mode which opens and writes from the last line of the file or creates a new file if the specified file is not existing.
a+	This is read/append mode. File contents are preserved by writing to the end of the file.
X	This is treated as write only mode which creates a new file and returns FALSE and an error if the specified file is already existing.
x+	This is treated as read/write mode which creates a new file and returns FALSE and an error if the specified file is already existing.

Example 3.74 File opening and file closing function

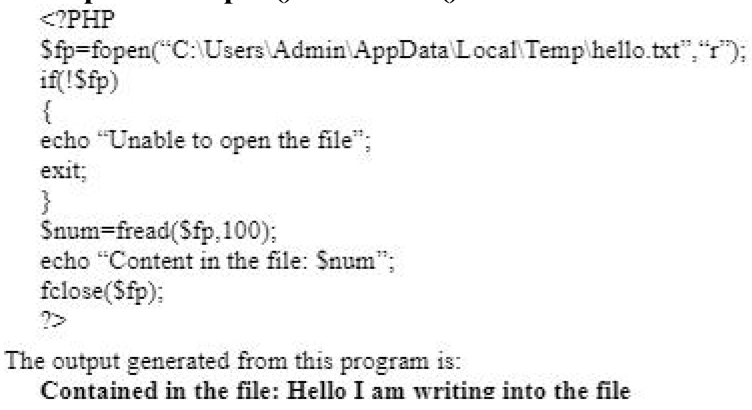
```
<?PHP
// open file to read
$toread = fopen('some/file.ext','r');
// open (possibly new) file to write
$towrite = fopen('some/file.ext','w');
// close both files
fclose($toread);
fclose($towrite);
?>
```

3.17.3 Writing to a File

The fwrite function allows information to be written for any type of file. This function takes file handle as its first parameter and string of the data that is to be written as its second parameter. Use fwrite() function to write data to a file. The general form of fwrite() function is:

```
fwrite($handle,$data)
```

Example 3.75 fopen() and fclose() functions

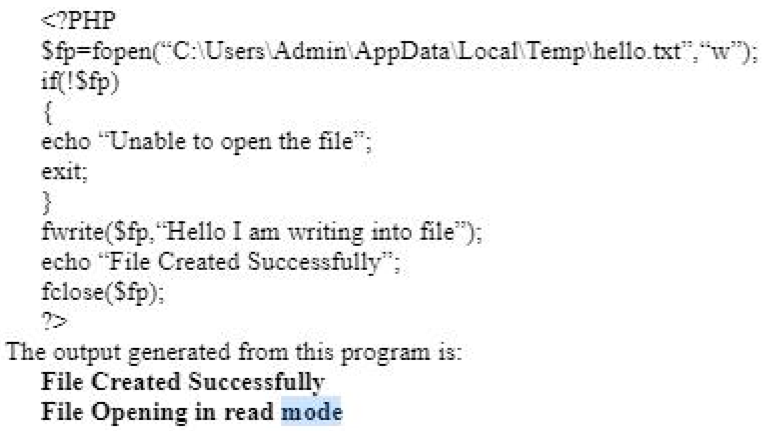


```
<?PHP
$fp=fopen("C:\Users\Admin\AppData\Local\Temp\hello.txt","r");
if(!$fp)
{
echo "Unable to open the file";
exit;
}
$num=fread($fp,100);
echo "Content in the file: $num";
fclose($fp);
?>
```

The output generated from this program is:

```
Contained in the file: Hello I am writing into the file
```

Example 3.76 fopen(), fwrite() and fclose() functions



```
<?PHP
$fp=fopen("C:\Users\Admin\AppData\Local\Temp\hello.txt","w");
if(!$fp)
{
echo "Unable to open the file";
exit;
}
fwrite($fp,"Hello I am writing into file");
echo "File Created Successfully";
fclose($fp);
?>
```

The output generated from this program is:

```
File Created Successfully
File Opening in read mode
```

3.17.4 Reading Data

The function fread reads data from a file. It requires a file handle and an integer which tells the function how many bytes of data supposed to read. One byte is equal to one character. For example, if we want to read the first six characters then one should specify six as the integer. There are two important functions which helps to read data:

- fgets($handle,$bytes)–Reads up to specified $bytes of data and stops at next line or at end of file (EOF).
- fread($handle,$bytes)–Reads up to specified $bytes of data and stops at end of file (EOF).

3.17.5 Appending to a File

Appending to a file means that the data or another file's data will be added to the end of an already existing file without erasing or removing any previously existing data in the original file. When we want to append data to the end of a file, it is necessary that the file must exist. While appending a file to the end of a different file, both the files must exist.

Example 3.77 Appending to a file:

```
<?PHP
$fp=fopen("C:\Users\Admin\AppData\Local\Temp\hello.txt","a");
if(!$fp)
{
echo "Unable to open the file";
exit;
}
fwrite($fp,"<br>This Line is appended to the file");
fclose($fp);
$fp=fopen("/tmp/hello.txt","r");
$num=fread($fp,100);
echo "Content in the file after appending: $num";
fclose($fp);
?>
```

The output generated by this program is:

```
Contained in the file after appending: Hello I am writing into the file
This Line is appended to the file.
```

3.17.6 File Truncate

This function is useful to truncate the file to a given length. It takes the file pointer, handle, and truncates the file to required length and size. While truncating, if the size mentioned is larger than the file, then the file is extended with null bytes. If the size mentioned is smaller than the file, then the file is truncated to that size. Returns TRUE on success or FALSE on failure. The general form of file truncate is:

bool ftruncate (resource $handle , int $size);

Example 3.78 Truncate function in a file:

```
<?PHP
$fname = 'PHP.txt';
$handle = fopen($fname, 'r+');
ftruncate($handle, 3);
rewind($handle);
echo fread($handle, filesize($fname));
fclose($handle);
?>
```

The output generated by this program is:

```
PHP
```

Example 3.78 How to Truncate a file:

```
<?PHP
$fp=fopen("/tmp/hello1.txt","w");
if(!$fp)
{
echo "Unable to open the file";
exit;
}
fwrite($fp,"<br>This Line is written into the file");
fclose($fp);
$fp=fopen("/tmp/hello1.txt", "a");
if(!$fp)
{
   echo "Unable to open the file";
   exit;
}
ftruncate($fp,24);
fclose($fp);
$fp=fopen("/tmp/hello1.txt","r");
$num=fread($fp, 100);
echo "Content after truncating: $num";
fclose($fp);
?>
```

The output generated by this program is:

```
Content after truncating:
This Line is written.
```

3.17.7 File Uploading

The server which is responsible for processing the code will transfer the files to the browser using HTTP. Using HTTP we can transfer any kind of files, like zip files, MPEG files, PDF files and a variety of file types, even FTP is standard for uploading files to a server. PHP supports $_FILES, is_upload_file () and move_uploaded_file (). The file uploading is an understanding between the $_FILES superglobal and a properly coded web form. The file_uploads directive determines whether PHP script can accept file uploads or not. The general form of file uploading is:

```
file_uploads = ON/OFF
Scope PHP_INI_SYSTEM : Default value - ON
```

The max_input_time directive determines the maximum amount of time in seconds. The script will spend attempting to parse input before registering a fatal error.

```
max_input_time = integer
Scope PHP_INI_ALL : Default value – 60
```

The below directive sets an upper limit to upload.

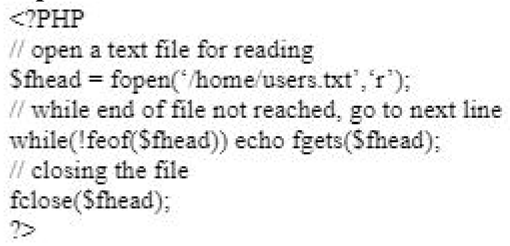
```
max_file_uploads = integer
Scope PHP_INI_SYSTEM ; Default value - 20
```

3.17.8 End-of-File Character

The program requires to identify when the end of file has reached and this is identified by EOF character. It is a built in function for verifying that the control has arrived at the EOF. In PHP, this function is feof(). It is used to find if the file has reached the end point. It is used quite most frequently in file input/output operations. The general form of end-of-file character is:

```
int feof(string resource)
```

Example 3.79 End-of-file character



```
<?PHP
// open a text file for reading
$fhead = fopen('/home/users.txt','r');
// while end of file not reached, go to next line
while(!feof($fhead)) echo fgets($fhead);
// closing the file
fclose($fhead);
?>
```

3.17.9 File Open Shortcuts

Without using the open function there are two 'shortcut' functions:

- $lines = file($filename)–This function reads an entire file into an array such that each line separates entry in the array.
- $str = file_get_contents($filename)–This function reads the entire file into a single string.

3.18 OTHER FILE OPERATIONS IN PHP

Deleting a File

The unlink () function deletes a file. The general form for deleting a file is:

```
unlink('filename');
```

Example 3.80 Unlink() function:

```
<?PHP
$file = "hello.txt";
if (!unlink($file))
  {
  echo ("Error deleting $file");
  }
else
  {
  echo ("Deleted $file");
  }
```

?>
The output generated by this program is:
Deleted hello.txt

Renaming a File (or) a Directory

The rename() function renames a file or directory. The general form of rename() function is:

rename('old name', 'new name');

Example 3.81 rename() function:

```
<?PHP
$fname = 'PHP.txt';
rename('PHP.txt', 'hai.txt');
?>
```

The output generated by this program is:
The file name 'PHP.txt' will be replaced with the filename 'hai.txt'

Copying a File

The copy() function copies a file. This function returns TRUE on success and FALSE on failure. The general form of copying a file is:

copy('source', 'destination');

Example 3.82 How to copy text from one file to another file:

```
<?PHP
echo copy("hello.txt","images.txt");
?>
```

The output generated by this program is: Previously, the text in hello.txt was 'PHP is Hypertext Preprocessor' after copying the text from images.txt, now the text in the file hello.txt is 'PHP is Hypertext Preprocessor'. Originally, it was called Personal Home Pages.

3.19. MySQL Database

3.19.1 INTRODUCTION

My SQL one of the most powerful and popular Open Source GPL (GNU General Public Licence) software that uses Structured Query Language (SQL). When web developers want to use an application, they can obtain it by using a standard database. By using a database it is easier to store, organize and access the data. In 1994, a commercial Swedish company, named

MySQL AB, established by MySQL developers initiated the development of MySQL, but later on, it was acquired by Sun Microsystems.

MySQL is easy to use, reliable, fast and suitable for applications of any size. To develop web-based applications, MySQL uses Relational DataBase Management System (RDBMS). By using a PERL DataBase Independent interface (DBI), it is easy to integrate MySQL into PERL programs. With the help of C/C++ MySQL was developed. Except for C/C++, other languages, like C#, PHP, Java, Python, PERL, Eiffel, Ruby have MySQL API's.

Advantages of MySQL

- ✓ MySQL has an open source licence, so we can use at free of cost.
- ✓ MySQL performance is very high because of unique storage engine architecture.
- ✓ It is portable, which means it is possible to work on many different platforms.
- ✓ For portability, it uses CMake in MySQL 5.5 and higher versions. But in previous versions it uses Autoconf, GNU Automake and Libtool.
- ✓ It supports multi-threading, which means, if multiple CPU's are available it can use them easily.
- ✓ It supports independent modules with multi-layered design.
- ✓ For the purpose of storage, MySQL provides both transactional and non-transactional engines.
- ✓ It helps the developer to give higher productivity by using views, stored procedures and triggers.
- ✓ Using highly optimized class library, MySQL implements SQL functions. After query Initialization there is no memory allocation at all.
- ✓ The feature of supporting a huge number of embedded applications makes MySQL very flexible.
- ✓ In a networked client/server environment, MySQL provides server as a separate program and also as a library, which can be linked into standalone applications. Such applications can be used in an environment where the network is not available.
- ✓ MySQL supports implementation of in-memory hash tables. These tables can be used as temporary tables.
- ✓ The execution of joins is very fast in MySQL, because it follows an optimized nested-loop join. For memory allocation, MySQL uses thread-based memory allocation system.
- ✓ In MySQL, it is easy to add other storage engines. This is useful when you want to provide your in-house database with a SQL interface.
- ✓ MySQL works quickly and well even with large data sets.
- ✓ It is customisable, which means it allows the program writer to modify the MySQL software suitable for their environment.

Limitations of MySQL

- ✓ MySQL cannot support large size databases efficiently.
- ✓ Even though MySQL is a faster application development database system, it has limited usage in real time applications, and has no OLAP functions.
- ✓ In versions below 5.0, MySQL does not support stored procedures, i.e., COMMIT and ROLE commands.
- ✓ MySQL cannot handle transactions efficiently.

- ✓ In MySQL, it is harder to write stored code because there is no better development and debugging tools.
- ✓ Many database systems support definitions of custom data types, but MySQL does not maintain such functionality.

3.19.2 SETTING UP AN ENVIRONMENT

Installing/Configuring MySQL Database on LINUX

MySQL requires at least Linux kernel versions 2.0 or higher. For installation of MySQL, Linux supports a different number of solutions. It is recommended to use any of the free distributions from Oracle. The options available are as follows:

- ✓ Installing by using Generic Binaries. It means from generic binary package of format .tar.gz.
- ✓ Installing from source. It means obtaining a source distribution of extracting and complied MySQL.
- ✓ Installing by using Redhat Package Manager (RPM) packages.

The installation of MySQL 3.23 on Linux is described. One of the most recommended ways for installing MySQL on a LINUX system is through RPM. Go to the download links of Linux and download both rpm files of client and server programs. The downloaded file appears in local directory. The following are the files:

```
MySQL−3.23.43-1.i386.rpm
MySQL−client-3.23.43-1.i386.rpm
```

To view all the files of the RPM packages, run

```
shell> rpm −qpl MySQL3.23.43-1.i386.rpm
```

Perform the installation as root, run

```
shell> rpm −i MySQL-3.23.43-1.i386.rpm MySQLclient-3.23.43-1.i386.rpm
```

In '/var/lib/mysql' the RPM places the data. It also creates '/etc/rc.d' for appropriate entries and to automatically start the server at boot time.

Once the installation of MySQL is completed, the safe_mysqld script which is in /usr/bin directory starts the database server by means of mysqlddaemon and in the same directory mysql_install_db script create six tables in the MySQL database. The six tables are user, host, db, columns_priv, table_priv, and func. The administrator controls the database. It is necessary because it stores user access privilege information. The server must be running all the time. Run the following commands, to check whether the server is running or not.

```
shell> /usr/bin/mysqladmin version
shell> /usr/bin/mysqladmin variables
```

Installing/Configuring MySQL Database on WINDOWS

Install XAMPP server which contains a PHP module on the top of the web server– Apache with MySQL for windows operating system.

Installation Procedure for XAMPP

Step1: To download XAMPP visit the following website: www.apachefriends.org/en/xampp.htm/
Step 2: Click on the 'Installer' in the above-mentioned website.
Step 3: 'XAMPP-win32-1. 7.0-Installer. exe' file is to be installed.
Step 4: Installation wizard to install XAMPP will appear on the screen.
Step 5: Choose the destination folder in any of the local drive (c:\XAMPP), then click next.
Step 6: Select 'Install XAMPP, Install My SQL' and click on install.
Step 7: Click on finish, XAMPP server is installed on the system.

3.19.3 STARTING, TERMINATING AND WRITING YOUR OWN SQL PROGRAMS

1. Creating New User and Granting Permissions

To create a new user in MySQL, execute the following command at 'mysql' prompt:

CREATE USER 'username'@'localhost' IDENTIFIED BY 'user password';

The newly created user has no privileges to work with the database. Therefore, provide the user with privileges to access the information with the following command:

GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost';

'*' specifies the database and the table that the user can access. The above command allows the user to edit, execute, read and perform all the tasks across all the databases and tables.

To provide specific privilege to the user:

GRANT [type of permission] ON [db_name]. [table_name] TO '[user_name]'@'localhost';

The list of permissions available in MySQL is as follows:

ALL PRIVILEGES: Provide all access or permissions to the user to a selected database.
CREATE: Allows the user to create new databases or tables.
DROP: Allows the user to delete databases or tables.
DELETE: Allows the user to delete the rows of a table.
INSERT: Allows the user to insert the rows into the table.
SELECT: Allows the user to select the data from database by using the SELECT command.
UPDATE: Allows the user to update the data in the rows of a table.

2. Execution of SQL Commands with MySQL Client

One can connect MySQL server and MySQL client with the help of mysql command. By default, the password is set to blank. There is no need to assign any password at that moment. Use the following command:

[root@host] # mysql

One can observe 'mysql>' prompt. It denotes that we connect to MySQL server. Therefore, we can execute SQL commands.

3. Creation and Usage of a Database

To create a new database, first check the current database whether the new database name already exists in the storage. Using SHOW statement can verify what are the databases existing in the server:

mysql> SHOW DATABASES;

Database
mysql
tmp
test

3 rows in set (0.00 sec)

The database list varies from machine to machine. But 'test' and 'mysql' databases exist among them. The 'mysql' database describes user access permissions and the 'test' database provides workspace for users to try out their things.

Use the CREATE DATABASE statement for the creation of new databases.

Example 1. Creation of the database:

mysql> CREATE DATABASE persons;

Query OK, 1 row affected (0.00 sec)

mysql>SHOW DATABASES;

Database
mysql
tmp
persons
test

4 rows in set (0.00 sec)

Note that the SQL commands and sub commands (In Example 1, CREATE is a command and DATABASE is a sub command) are case insensitive. But database names, table names and their fields are case sensitive.

2. USE Command

Explicitly select the database for use after its creation, by using USE command. Note, that USE must be specified on a single line.

Example 2. Usage of USE command:

mysql>USE persons;

Database changed

3. Creation of Tables

After creating databases, tables must be defined and created. A table can be created by using CREATE TABLE command. For example, creating a table named person_information which stores first name, last name and age of an individual or person. MySQL wants to know the type of the data that you are going to store in these fields. In this case, first name and last name are of character type and age is of type integer. The general form to create a table:

```
CREATE TABLE table_name (
        col_name1 datatype(size),
        col_name2 datatype(size),
        ... );
```

```
mysql> CREATE TABLE person_information (
-> plastname CHAR(20),
-> pfirstname CHAR(20),
-> page INT
-> );
Query OK, 0 rows affected (0.00 sec)
```

At the end of execution of the query, it displays OK which means that table is created properly. This can be checked by using SHOW TABLES command:

```
mysql> SHOW TABLES;
```

Tables_in_persons
person_information

1 row in set (0.00 sec)

4. DESCRIBE Command

To get information about fields in a table and to verify whether the table was created in the way you expected, one can use the DESCRIBE command.

```
mysql> DESCRIBE person_information;
```

Field	Type	Null	Key |	Default	Extra
plastname	char(20)	YES	NULL		
pfirstname	char(20)	YES	NULL		
page	int(11)	YES	NULL		

3 rows in set (0.00 sec)

5. INSERT Command

To insert new records into a table, use INSERT command. The general form of insert command:

```
INSERT INTO table_name (col_name1, col_name2, . . .) VALUES (value1, value2, . . .);
```

```
mysql> INSERT INTO person_information
-> (plastname, pfirstname, page)
-> VALUES ('konda', 'tejaswini', 26);
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO person_information
-> (plastname, pfirstname, page)
-> VALUES ('avupati', 'ravi', 21);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> INSERT INTO person_information
-> (plastname, pfirstname, page)
-> VALUES ('gowda', 'ganesh', 23);

Query OK, 1 row affected (0.00 sec)
```

Note, that the values we supply for each column must be in the order of columns listed in the table creation.

3.19.4 RECORD SELECTION TECHNOLOGY

1. **SELECT Command**

To get information from a table, use SELECT command. The general form of select command:

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy
```

what_to_select specifies what you wish to see. This can be a column list, or * which specifies all the columns. which_table specifies the table from where you want to retrieve data. The clause WHERE is optional. It specifies the conditions that rows must satisfy for retrieval. Given below are different ways to work with SELECT statement:

```
SELECT * FROM table_name;
SELECT * FROM table_name [WHERE Clause]
SELECT expressions_and_columns FROM table_name
        [WHERE some_condition_is_true]
        [LIMIT offset, rows]
```

OFFSET attribute tells the SELECT from where to start the retrieval of records. Offset is set to zero by default and the LIMIT attribute, limits the retrieval number of records:

```
SELECT expressions_and_columns FROM table_name
[WHERE some_condition_is_true]
[ORDER BY some_column [ASC | DESC]]

SELECT col_name, aggregate_fun(col_name)
FROM table_name
GROUP BY col_name;
```

Example 3. Usage of a SELECT command:

To select all the information from the table:

mysql> SELECT * FROM person_information;

plastname	pfirstname	page

konda	tejaswini	26
avupati	ravi	21
gowda	ganesh	23

3 rows in set (0.00 sec)

To select a particular row from the table:

```
mysql> SELECT * FROM person_information ->where plastname='gowda';
```

plastname	pfirstname	page
gowda	ganesh	23

1 row in set (0.00 sec)

To select particular columns from the table:

```
mysql> SELECT pfirstname, page FROM person_information -> where page>21 limit 0,2;
```

pfirstname	page
tejaswini	26
ganesh	23

2 row in set (0.00 sec)

2. **UPDATE Command**

With the help of UPDATE command one can update or modify the information in a table.The general form of update command:

```
UPDATE table_name
SET col_name1=value1, col_name2=value2, . . .
[WHERE Clause]
```

Example 4. Usage of UPDATE command:

```
mysql> SELECT * FROM person_information;
```

plastname	pfirstname	page
konda	tejaswini	26
avupati	ravi	21
gowda	ganesh	23

3 rows in set (0.00 sec)

One can observe that the values under column age changes from year to year. So, by using the UPDATE command one can update the values in the age.

```
mysql> UPDATE person_information SET page = 27 ->
         WHERE plastname = 'konda';
         Query OK, 1 row affected (0.00 sec)
         Rows matched: 1 Changed: 1 Warnings: 0
mysql> SELECT * FROM person_information;
```

plastname	pfirstname	page
konda	tejaswini	27
avupati	ravi	21
gowda	ganesh	23

3 rows in set (0.00 sec)

Remember, that one should use WHERE CLAUSE only, otherwise, if one enters UPDATE person_information SET page = 27 then all the values in the column age are updated as 27.

3. **DELETE Command**

If one wish to delete a record from a table then use DELETE command. The general form of delete command:

DELETE FROM table_name [WHERE Clause]

Example 5. Usage of DELETE command:

```
mysql> DELETE FROM person_information
->WHERE plastname = 'gowda';
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM person_information;
```

plastname	pfirstname	page
konda	tejaswini	27
avupati	ravi	21

2 rows in set (0.00 sec)

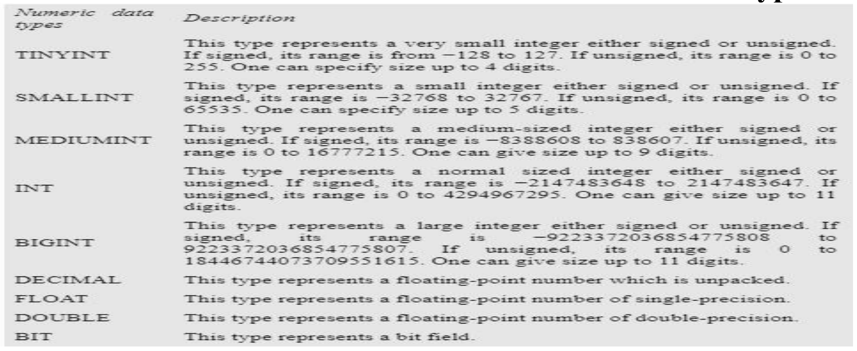
4. Data Types

Every table in a database contains multiple columns with a specific data type, like 'numeric' or 'string'. Other than string or numeric data types MySQL supports other data types, like date and time data types.

5.Numeric Types

MySQL supports all SQL standard numeric types including exact, and approximate value of numeric data types including fixed-point, floating point and integer. For storing bit field values MySQL supports BIT data type. Except BIT data type, all other numeric data types are signed or unsigned. Table 1 gives you the description of numeric data types in MySQL.

Table 1: Numeric data types



Numeric data types	*Description*
TINYINT	This type represents a very small integer either signed or unsigned. If signed, its range is from −128 to 127. If unsigned, its range is 0 to 255. One can specify size up to 4 digits.
SMALLINT	This type represents a small integer either signed or unsigned. If signed, its range is −32768 to 32767. If unsigned, its range is 0 to 65535. One can specify size up to 5 digits.
MEDIUMINT	This type represents a medium-sized integer either signed or unsigned. If signed, its range is −8388608 to 838607. If unsigned, its range is 0 to 16777215. One can give size up to 9 digits.
INT	This type represents a normal sized integer either signed or unsigned. If signed, its range is −2147483648 to 2147483647. If unsigned, its range is 0 to 4294967295. One can give size up to 11 digits.
BIGINT	This type represents a large integer either signed or unsigned. If signed, its range is −9223372036854775808 to 9223372036854775807. If unsigned, its range is 0 to 18446744073709551615. One can give size up to 11 digits.
DECIMAL	This type represents a floating-point number which is unpacked.
FLOAT	This type represents a floating-point number of single-precision.
DOUBLE	This type represents a floating-point number of double-precision.
BIT	This type represents a bit field.

6. String Types

One can store plain text to binary data, like files or images in MySQL using strings. Table 2 provides the description of string types in MySQL. One can use BLOB or TEXT data types in strings. The difference between the two is: BLOB performs case sensitive sorts and comparisons on stored data whereas TEXT do not perform case sensitive sorts and comparisons on stored data.

Table 2 :string data types

String data types	*Description*
CHAR	This type represents a non-binary string of fixed-length. Its length varies between characters 1 and 255. It is not required to specify length, but the default size is 1.
VARCHAR	This type represents a non-binary string of variable-length. Its length varies between characters 1 and 255. One must define the length when creating a VARCHAR field.
BINARY	This type represents a binary string of fixed length.
VARBINARY	This type represents a binary string of variable length.
TINYBLOB	This type represents a very small BLOB (Binary Large Object) or TEXT. Do not give size with TINYTEXT or TINYBLOB. Its maximum length is 255 characters.
BLOB	This type represents a small BLOB to store binary data. Do not give length when defining TEXT or BLOB field. Its maximum length is 65535 characters.
MEDIUMBLOB	This type represents a medium-sized BLOB. Do not give length when defining MEDIUMTEXT or MEDIUMBLOB field. Its maximum length is 16777215 characters.
LONGBLOB	This type represents a large BLOB. Do not give length when defining LONGTEXT or LONGBLOB field. Its maximum length is 429497295 characters.
ENUM	This type helps to create a list item. Each column value is assigned to one enumeration member.
SET	This type helps to create a list of items. Each column value is assigned to zero or more set members.

7. **Date and Time**

Data Types To represent temporary values MySQL provides date and time types. It has TIME, DATE, and DATETIME. In addition, to track the changes in a row of a table MYSQL provides TIMESTAMP data type. One can also store year without date and month by using YEAR data type.

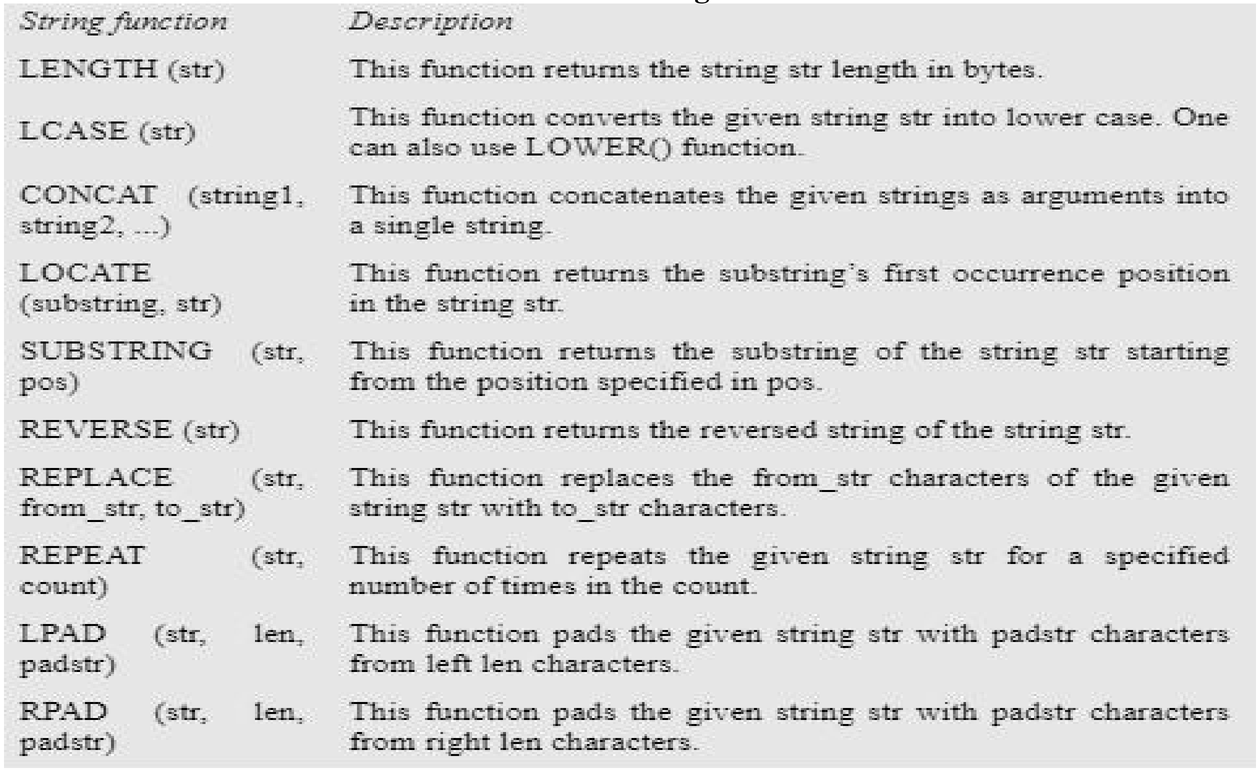
Table 3 : date and time data types

Date and time types	*Description*
TIME	This type stores time in the format of HH:MM:SS.
DATE	This type stores the date in the format of YYYY-MM-DD between 1000-01-01 and 9999-12-31.
DATETIME	This type is a combination of both time and date. It stores both date and time in the format of YYYY-MM-DD HH:MM:SS.
YEAR	This type stores the value of the year in the format of YYYY or YY.
TIMESTAMP	The type timestamp is same as the type DATETIME, but it would not have hyphens in the format. The value of the timestamp is stored in the format of YYMMDDHHMMSS or YYYYMMDDHHMMSS or YYMMDD or YYYYMMDD.

3.19.5. WORKING WITH STRING FUNCTIONS

A function is a set of statements grouped together to perform an operation which is stored in a library. A function may or may not return a value. Functions which perform string operations are known as string functions . In MySQL, there are different types of string functions. Important MySQL string functions are shown here.

Table 4: string functions



String function	*Description*
LENGTH (str)	This function returns the string str length in bytes.
LCASE (str)	This function converts the given string str into lower case. One can also use LOWER() function.
CONCAT (string1, string2, ...)	This function concatenates the given strings as arguments into a single string.
LOCATE (substring, str)	This function returns the substring's first occurrence position in the string str.
SUBSTRING (str, pos)	This function returns the substring of the string str starting from the position specified in pos.
REVERSE (str)	This function returns the reversed string of the string str.
REPLACE (str, from_str, to_str)	This function replaces the from_str characters of the given string str with to_str characters.
REPEAT (str, count)	This function repeats the given string str for a specified number of times in the count.
LPAD (str, len, padstr)	This function pads the given string str with padstr characters from left len characters.
RPAD (str, len, padstr)	This function pads the given string str with padstr characters from right len characters.

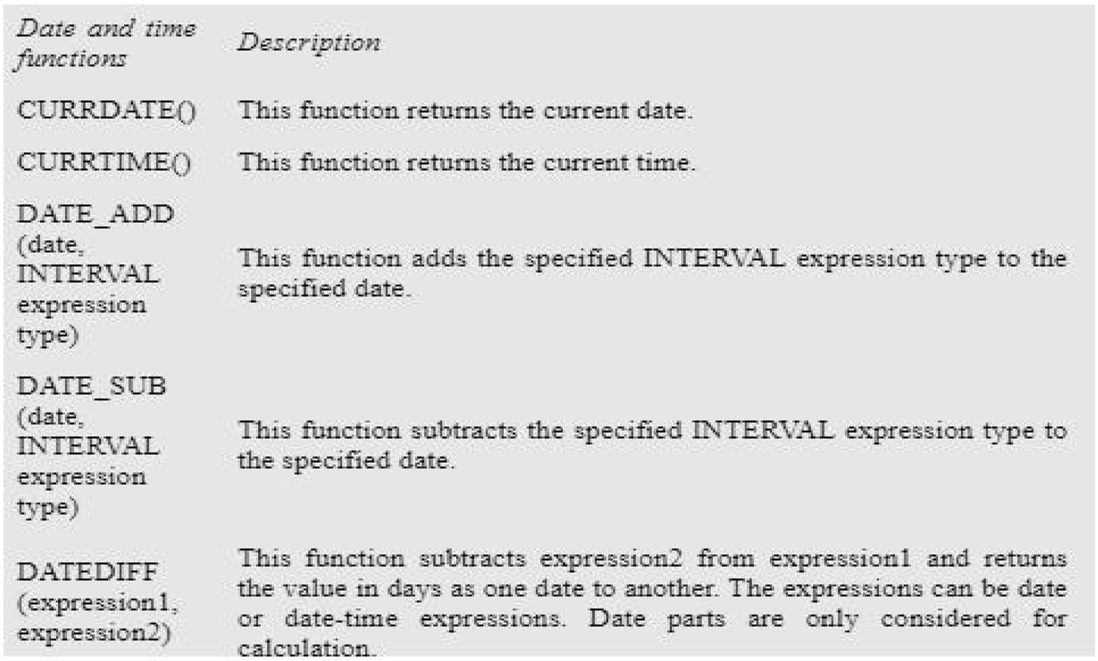
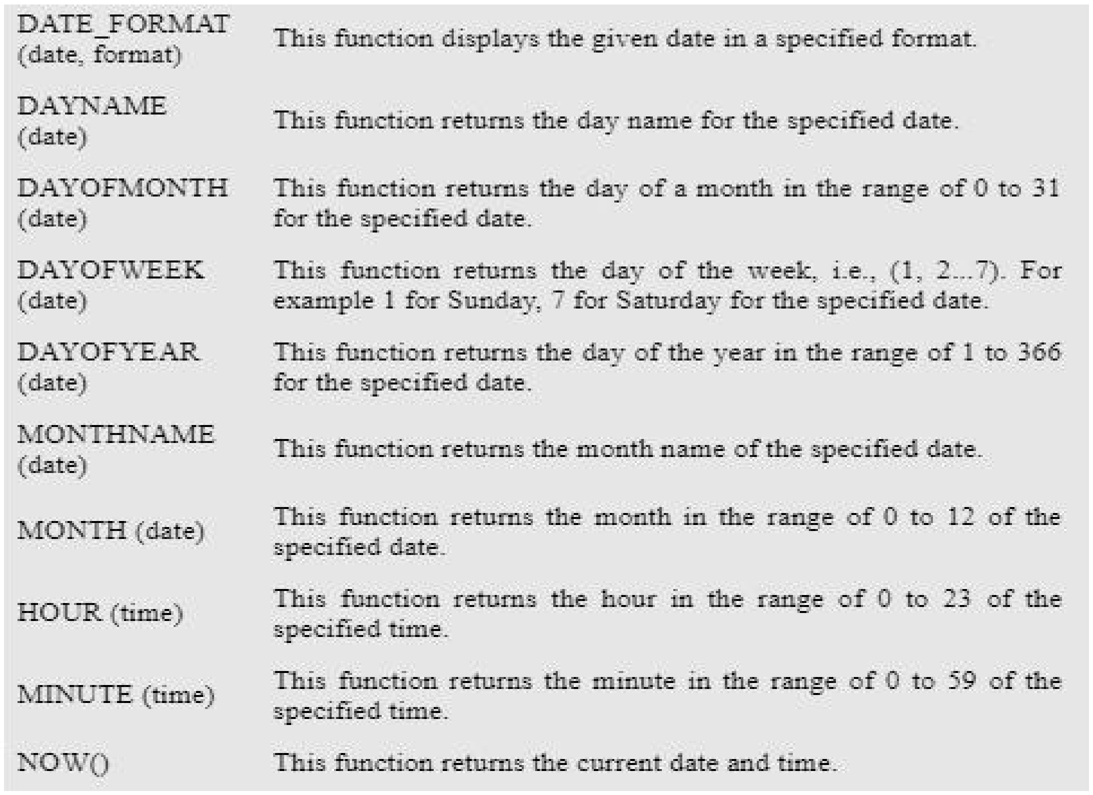
Example 6. Usage of string functions:

```
mysql> select concat ('my', 'sql');
        mysql

mysql> select substring('mysql',3);
        sql

mysql> select repeat('geethagadi',2);
        geethagadigeethagadi

mysql> select rpad('exam',8,'??');
        exam????
```

3.19.6. WORKING WITH DATE AND TIME

Using these functions one can manipulate the displaying format of date and time. In MySQL there are different types of date and time functions. Important MySQL date and time functions are shown in Table 5.

Table 5: MySQL date and time functions

Date and time functions	*Description*
CURRDATE()	This function returns the current date.
CURRTIME()	This function returns the current time.
DATE_ADD (date, INTERVAL expression type)	This function adds the specified INTERVAL expression type to the specified date.
DATE_SUB (date, INTERVAL expression type)	This function subtracts the specified INTERVAL expression type to the specified date.
DATEDIFF (expression1, expression2)	This function subtracts expression2 from expression1 and returns the value in days as one date to another. The expressions can be date or date-time expressions. Date parts are only considered for calculation.
DATE_FORMAT (date, format)	This function displays the given date in a specified format.
DAYNAME (date)	This function returns the day name for the specified date.
DAYOFMONTH (date)	This function returns the day of a month in the range of 0 to 31 for the specified date.
DAYOFWEEK (date)	This function returns the day of the week, i.e., (1, 2...7). For example 1 for Sunday, 7 for Saturday for the specified date.
DAYOFYEAR (date)	This function returns the day of the year in the range of 1 to 366 for the specified date.
MONTHNAME (date)	This function returns the month name of the specified date.
MONTH (date)	This function returns the month in the range of 0 to 12 of the specified date.
HOUR (time)	This function returns the hour in the range of 0 to 23 of the specified time.
MINUTE (time)	This function returns the minute in the range of 0 to 59 of the specified time.
NOW()	This function returns the current date and time.

Example 7. Usage of date() and time() functions:

```
mysql> select date_add('2014-10-19',interval 6 day);
        2014-10-25
mysql> select datediff('2013-10-17','2013-10-16');
         1
mysql> select date_format('2013-09-15','%w');
```

```
        2
mysql> select date_format('2014-10-19 12:25:47','%s');
        47
mysql> select dayofyear('2013-08-17');
        229
```

3.19.7 SORTING QUERY RESULTS

In MySQL, one can fetch the data from a table by using the SELECT command. MySQL server returns the selected rows without any order until one instructs how to sort the result. To sort the resultant set, use ORDER BY clause. The general form of 'order by' clause:

SELECT column1, column2, column3, ...columnN
FROM table_name1, table_name2, table_name3...
ORDER BY column1, [column2, column3...] [ASC/DESC]];

Based on more than one column one can sort the result. ASC or DESC keywords in the syntax represent that one can sort the result in ascending order by using the ASC keyword and in descending order by using the DESC keyword. One can also use WHERE clause in the syntax, if want to specify any condition.

Example 8 : Usage of 'order by' clause:

```
mysql> SELECT * FROM person_information
-> ORDER BY plastname;
```

plastname	**pfirstname**	**page**
avupati	ravi	21
gowda	ganesh	23
konda	tejaswini	27

3 rows in set (0.00 sec)

In Example 8, the table is sorted based on the plastname column in the table. To view the ages of the persons in descending order, the query is as follows:

mysql> SELECT page FROM person_information -> ORDER BY page DESC;

page
27
23
21

3 rows in set (0.00 sec)

To view the first names of those who are older than 21, the query is as follows:

mysql> SELECT pfirstname FROM person_ information ->WHERE page > 21;

pfirstname
tejaswini
ganesh

2 rows in set (0.00 sec)

To view the first names of those who are older than 21 but sort by ‘pfirstname’, the query is as follows:

```
mysql> SELECT pfirstname FROM person_ information
    ->WHERE page > 21 ORDER BY pfirstname;
```

pfirstname
ganesh
tejaswini

2 rows in set (0.00 sec)

3.19.8 GENERATING SUMMARY

Database systems are not only useful for storing and retrieving data, they can also summarize the data in more concise forms. With the help of summary, one can obtain overall representation of the data. The following are the summary operations in MySQL:

- ✓ By using aggregate functions, one can obtain a summary value rather than getting set of individual values. The aggregate functions include COUNT(), SUM(), AVG(), MIN() and MAX(). Using the GROUP BY clause, is an alternative way which results in an aggregate value for each one by grouping the rows into subsets.
- ✓ To eliminate the duplicate values, use SELECT DISTINCT.
- ✓ To obtain the count of unique or distinct values, use COUNT(DISTINCT).

1. Summarizing with COUNT()

One can use COUNT() function to count the number of times that a particular value occurs, the number of rows in a table or the number of rows that match certain conditions.

Example 9 : employee_workdetails table having the subsequent data:

```
mysql> SELECT * FROM employee_workdetails;
```

empid	empname	worked_date	typed_pages
1	Jack	2012-02-23	150
2	Raj	2012-06-28	240
1	Jack	2012-09-08	270
3	John	2012-12-25	105
4	Jill	2013-01-06	320
5	Zarina	2013-04-05	400
5	Zarina	2013-05-06	270

7 rows in set (0.00 sec)

To count the number of rows in the table, the query is as follows:

```
mysql> SELECT COUNT(*) FROM employee_workdetails;
                7
```

To count the number of records for Jack, the query is as follows:

```
mysql>SELECT COUNT(*) FROM employee_workdetails
    ->WHERE empname=’Jack’;
    2
```

2. Summarizing with MIN() and MAX()

To find a minimum or a maximum value from a set of values, MIN() and MAX() functions are used.

Example 10 : To find the minimum and maximum number of pages typed by employees from employee_workdetails table:

```
mysql> select MAX(typed_pages) FROM employee_workdetails;
          400

mysql> select MIN(typed_pages) FROM employee_workdetails;
          105
```

Example 11:To find the maximum number of pages typed by employees from employee_workdetails table by using the GROUP BY clause:

```
mysql> SELECT empid, empname, MAX(typed_pages)
          -> FROM employee_workdetails GROUP BY empname;
```

empid	empname	typed_pages
1	Jack	270
4	Jill	320
3	John	105
2	Raj	240
5	Zarina	400

Example 12 : To find the minimum number of pages typed by employees from employee_workdetails table by using the GROUP BY clause:

```
mysql> SELECT empid, empname, MIN(typed_pages)
    -> FROM employee_workdetails GROUP BY empname;
```

empid	empname	typed_pages
1	Jack	150
4	Jill	320
3	John	105
2	Raj	240
5	Zarina	270

3. Summarizing with SUM() and AVG()

To find the total and average or mean of a set of values, SUM() and AVG() functions are used.

Example 13: To find the average or mean of the number of pages typed by employees from employee_workdetails table:

```
mysql> select AVG(typed_pages) FROM employee_workdetails;
          250.7143
```

Example 14: To find the average or mean of the number of pages typed by employees from the employee_workdetails table by using the GROUP BY clause:

**mysql> SELECT empname, AVG(typed_pages) -> FROM employee_workdetails
GROUP BY empname;**

empname	**AVG typed_pages**
Jack	210.0000
Jill	320.0000
John	105.0000
Raj	240.0000
Zarina	335.0000

Example 15: To find the sum of number of pages typed by employees from employee_workdetails table:

mysql> SELECT SUM(typed_pages) FROM employee_workdetails;
1755

Example 16:To find the sum of number of pages typed by employees from employee_workdetails table by using the GROUP BY clause:

**mysql> SELECT empname, SUM(typed_pages)
-> FROM employee_workdetails GROUP BY empname;**

empname	**SUM typed_pages**
Jack	420
Jill	320
John	105
Raj	240
Zarina	670

4. **Eliminating the Duplicate Values Rather than using SELECT, use SELECT DISTINCT to eliminate duplicate values.**
By using ORDER BY clause with SELECT DISTINCT view the data in a more meaningful manner.

Example 17: To find the names of employees from employee_workdetails table:

mysql> SELECT DISTINCT empname FROM employee_workdetails ->ORDER BY empname;

empname
Jack
Jill
John
Raj
Zarina

If one view the above query without using distinct, the following result will be obtained:

mysql> SELECT empname FROM employee_workdetails;

empname
Jack
Raj
Jack
John
Jill
Zarina
Zarina

5. Using COUNT(DISTINCT)

By using the above function, one can obtain the unique values from a set of values. DISTINCT gives unlike combinations of column values, whereas COUNT(DISTINCT) gives the count of combinations.

Example 18: Usage of COUNT(DISTINCT) function:

Find the number of employees in the employee_workdetails table:

mysql> SELECT COUNT(DISTINCT empname) FROM employee_workdetails;

```
5
```

3.19.9 WORKING WITH METADATA

Metadata means data about the data or which describes the contents of the database. The metadata contains information about user names, database names, version names and column names. There are three types of metadata:

Information about Query Results: This includes the records which are affected by the UPDATE, SELECT and DELETE statements.

Information about Databases and Tables: This includes the information related to the structure of the database and tables which are useful for applications to catalogue a list of databases hosted on a server or tables in a database. Using this information one can decide whether the databases or tables exist or not.

Information about MySQL Server: This includes the status of the current connection to the server, version number to determine whether it supports a given feature or not, etc.

One can obtain this information at 'mysql' prompt easily by using PHP or PERL API's. The following section shows how to obtain this information by using PHP.

1. Obtaining the Count of Rows Affected by a Query

By invoking mysql_affected_rows() function of PHP, one can find out the number of rows affected by a query.

Example 19: Usage of mysql_affected_rows() function to find out the count of rows affected by a statement:

```
$resultid = mysql_query ($stmt, $conn); # report 0 rows if the query or statement fails
$count = ($resultid ? mysql_affected_rows ($conn) : 0);
print ("Number of affected rows:$count\n");
```

2. **Listing Databases and Tables**

It is easy to list all the databases and tables that exists within a database server. If one do not have enough privileges the result is null.

Example 20: List of database names:

```
<?php $conn = mysql_connect("localhost", "username", "pwd");
        If(!$conn)
        {
                die( 'connection failed:' .mysql_error());
        }
        $dbname_list=mysql_list_dbs($conn);
        while ($dbname = mysql_fetch_object($dbname_list))
        {
                echo $dbname->Database . "<br/>";
        }
        mysql_close($conn);
?>
```

3. Getting Metadata of Server

By executing the following commands in Table 6, either at 'mysql' prompt or in a PHP script, one can obtain information about the server.

Table 6: Metadata Commands

Command	**Description**
SELECT DATABASE( )	It gives the name of current database.
SELECT VERSION( )	It gives the server version as a string.
SELECT CURRENT_USER( )	To check the privileges of the client.
SELECT USER( )	It gives the name of the current user.
SHOW STATUS	It gives the status of server indicators.
SHOW VARIABLES	It gives variables of server configuration.
SHOW GLOBAL STATUS	It gives the global status of server indicators.

Example 21: Metadata of the server:

```
DatabaseMetaData dmd = conn.getMetaData ();
System.out.println ("version of the product: " + dmd.getDatabaseProductVersion());
System.out.println ("Username: " + dmd.getUserName());
```

3.19.10 USING SEQUENCES

1. Creation of Sequence in MySQL

A list of numbers generated in ascending order is known as a sequence . Sequences are used for identifying unique numbers. For example, 'Roll no.', of a student in the school, employee number in HR department, etc. To implement sequences in MySQL, set AUTO_INCREMENT attribute to the column. Then, the column typically becomes a primary key. To use AUTO_INCREMENT attribute, one must satisfy the following rules:

- ✓ There must be only one AUTO_INCREMENT column in each table. The data type of the column must be either a float which is the rare case or an integer.
- ✓ The column with the AUTO_INCREMENT attribute must be either a primary key or a unique key.
- ✓ The column with the AUTO_INCREMENT attribute must not have a NOT NULL constraint. If you do not specify it explicitly, MySQL will make the AUTO_INCREMENT column has NOT NULL.

Example 22 : Example that shows how to create 'student_info' table whose 'std_no' column is AUTO_INCREMENT:

```
my sql>CREATE TABLE student_info( ->std_no INT(5)
        AUTO_INCREMENT PRIMARY KEY, ->firstname VARCHAR(60),
        - >lastname VARCHAR(55) ->);
Query O K, 0 rows affected (0.00 sec)
```

2. **Working with MySQL Sequence**

The column with AUTO_INCREMENT has the following attributes:

- ✓ The value of AUTO_INCREMENT column starts with value 1 and gets increased by 1 when a NULL value is inserted or the value of the column in the INSERT command is omitted.
- ✓ By using LAST_INSERT_ID() function one can obtain the last generated sequence number.
- ✓ If a value for the sequence column while inserting a new record into the table is specified, MySQL inserts that value only if the value does not exist in that column or displays an error if the value already exists.
- ✓ If the inserted new value is larger than the next sequence number, MySQL uses the new value as the starting value of the sequence . For the next use, it generates a distinctive sequence number greater than the recent one. Because of this, gaps are created in the sequence.
- ✓ To update the value of an AUTO_INCREMENT column which is unique indexed with the value that already exists, MySQL displays a duplicate key error. If the updating value is greater than the values in the column, it will use the next number of the last inserted sequence for the next row. For example, if the last inserted sequence number is 4 and is updated to 20, the sequence number will be 5 for the new row.
- ✓ To delete the last insert row use the DELETE command. The sequence numbers may or may not be used depending on the storage engine. For example, the last insert row is 20 and in case it is deleted, MySQL use 21 for the next row.

Example 23. Sequences:

First, insert two new students into the student_info table.

```
mysql> INSERT INTO student_info -> (firstname, lastname)
        -> VALUES ('john', 'erics');
         Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO student_info -> (firstname, lastname)
        -> VALUES ('bhanu', 'kaliketi');
        Query OK, 1 row affected (0.00 sec)
```

Second, select the data

```
mysql> SELECT * FROM student_info;
```

std_no	firstname	lastname
1	john	erics
2	bhanu	kaliketi

2 rows in set (0.00 sec)

Third, delete a row whose std_no is 2.

```
mysql> DELETE FROM student_info
       ->WHERE std_no=2;
       Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM student_info;
```

std_no	firstname	lastname
1	john	erics

```
       1 row in set (0.00 sec)
```

Fourth, insert another new student.

```
mysql> INSERT INTO student_info
       -> (firstname, lastname)
       -> VALUES ('wall', 'linus');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM student_info;
```

std_no	firstname	lastname
1	john	Erics
3	wally	linus

```
       2 rows in set (0.00 sec)
```

The storage engine does not re-use the deleted sequence number, so the next row std_no is 3. Fifth, update the existing std_no 3 to 1.

```
mysql> UPDATE student_info
       ->SET firstname= 'wally',
        ->std_no=1
       -> WHERE std_no=3;
```

Here MYSQL displays an error 'duplicate entry for primary key'. Now let us fix it.

```
mysql> UPDATE student_info ->SET firstname= 'wally',
       ->std_no=11 -> WHERE std_no=3;
Query OK, 1 row affected (0.00 sec) Rows matched: 1 Changed: 1 Warnings: 0

mysql> SELECT * FROM student_info;
```

std_no	firstname	lastname
1	john	Erics
11	wally	linus

```
 2 rows in set (0.00 sec)
```

Sixth, insert another new student.

```
mysql> INSERT INTO student_info -> (firstname, lastname)
        -> VALUES ('raymond', 'larry');
       Query OK, 1 row affected (0.00 sec)
mysql> SELECT * FROM student_info;
```

std_no	firstname	lastname
1	john	Erics
11	wally	Linus
4	raymond	larry

3 rows in set (0.00 sec)

The last insert sequence is 3 so its next sequence number is 4. Therefore, MySQL uses 4 for the next row instead of 12.

3.20. PHP and MySQL DATABASE

Here we will learn how to display the information stored in a database to a web page. To perform this, we have two powerful tools: PHP , a scripting language for server side programming and MySQL , a relational database which is an open source free software. By combining these tools together we can create a database driven website. The main objective of database driven website is to maintain the content of the website in a database. Therefore, we can create web pages for viewing in the browser by obtaining the content from the database dynamically. At one end, we will have the visitor requesting a web page of our website from a browser and the browser in return expects an HTML page as a response to the visitor. At the other end, the content of the website is stored as tables in MySQL database. The MySQL database understands the way of responding to SQL commands.

The following steps describe how the client interacts with MySQL through PHP:

Step 1: The visitor requests the web page of your web server from his web browser.

Step 2: If the requested file is a PHP script, the server invokes the PHP interpreter to run the code enclosed in the file.

Step 3: It connects to the MySQL database by using certain PHP commands and then requests the content.

Step 4: As a response, MySQL database sends required information to the PHP script.

Step 5: The PHP script stores the received content into one or more variables of PHP and output the content as a web page by using 'echo' statement.

Step 6: The PHP interpreter handovers an HTML copy to the web server.

Step 7: Finally, the web server sends the HTML page to the web browser.

3.20.1 Creating a MySQL User Account

Now, we are going to create a separate username and password to connect PHP to MySQL database server by following the given steps:

Step 1: To launch phpMyAdmin into the browser, open XAMPP control panel and click the Admin button which is next to MySQL.

Step 2: Select 'i j db' database from the list.

Step 3: Click on 'privileges' tab.

Step 4: Click on 'Add New User' button which is at the bottom of the list and fill the following details:

User name	- (Use text field)
user's name Host	- (Local) localhost
Password	- (Use text field) your password

Step 5: Grant all the privileges to the database, under 'Database for user'.

Step 6: Leave Global privileges as unchecked.

Step 7: Click Go, which is at the bottom of the form.
Step 8: Finally the phpMyAdmin confirms the new user added by viewing a message.

3.20.2 Connecting to MySQL with PHP

Before inclusion of content of MYSQL database into a web page, establish connection to MySQL inside a PHP script. We can write our own PHP scripts. For establishing connection, use phpMyadmin, which is a PHP web application. The PHP Data Object (PDO) provides the support for connecting to a database. The statement below shows the usage of PDO for establishing connection to the MySQL server:

new PDO('mysql:host=hostname;dbname=database', 'username', 'password')

Consider, 'new PDO' as a built-in function which takes three arguments:

1. The first argument specifies the type of the database (i.e., mysql) which includes the server's host name and database name.
2. The second argument is the MySQL username (for utilisation of PHP).
3. The third argument is the password for that user.

The 'new PDO' function returns a 'PDO' object which helps to identify the establishment of a connection. Make use of the connection to store that value in a variable $pdo.

```
$pdo = new PDO('mysql:host=hostname;dbname=database', 'username', 'password')
```

Example 24 : MySQL with PHP:

```
<?php
        try
        {
                $pdo = new PDO('mysql:host=localhost;dbname=studentdb', 'johnuser',
                'abcd1234');
        } catch (PDOException $e)
        {
                $output= 'Unable to connect to the server.';
                 exit();
        }
?>
```

The alternative way to connect PHP and MySQL with following function:

mysqli_connect(host,username,password,dbname);

In the same way, to disconnect the following function is used:

mysqli_close();

3.20.3 Lightweight Directory Access Protocol

(LDAP) Directory services offer an efficient, secure and consistent means for managing and viewing resources, such as applications, files and printers to the end users, developers and system administrators. The structure of these data repository models the society colleges, departments or physical corporate structures. The extensively used directory service products depends on an open specification known as the Lightweight Directory Access Protocol (LDAP). One can learn LDAP via PHP's LDAP extension. It offers ease of use and flexibility to the developers while developing or creating complex LDAP applications.

Setting LDAP for PHP

To access LDAP server, configure LDAP support to the PHP which is not enabled by default. To install LDAP client, recompile PHP using with-ldap flag. On windows, within php.ini enable php_ldap.dll and make confident that the ssleay32.dll and libeay32.dll files exist in your system path.

Establishing Connection to an LDAP Server

For establishing connection to an LDAP server, use the function ldap_connect() which establishes the connection by identifying a specific host and port number which is optional. The standard port number for LDAP is 389.

Syntax

resource ldap_connect([string hostname [, int port]])

If the connection established successfully, the above command returns a link identifier. Otherwise, it returns an error ‘FALSE’.

Example 25: LDAP server:

```
<?php
        $hostname = “ldap.scetrnd.in”;
        $portno = 389;
        $connection = ldap_connect($hostname, $portno) or die(“Cannot establish
        connection ”);
?>
```

Binding to the LDAP Server

Before retrieving or manipulating the data, pass a set of credentials. For this, use ldap_bind() function. This function requires link_idn which is returned by ldap_connect(), username and password. The general form of ldap_bind() function:

boolean ldap_bind(resource link_idn[, string userrdn [, string pwd]])

Example 26: Binding to the LDAP server:

```
 <?php
        $hostname = “ldap.scetrnd.in”;
        $portno= 389;
        $conn = ldap_connect($hostname, $portno) or die(“Cannot establish connection”);
         ldap_set_option($conn, LDAP_OPT_PROTOCOL_VERSION, 3);
        ldap_bind($conn, $username, $pswd) or die(“Cannot bind to the LDAP server.”);
?>
```

Disconnecting LDAP Server

For closing the connection use ldap_unbind() function which terminates the connection related with link_id. The general form of ldap_unbind() function:

Boolean ldap_unbind(resource link_idn)

Example 27: Disconnect LDAP server:

```
<?php // Connecting to the server
        $conn = ldap_connect(“ldap.scetrnd.in”) or
```

```
                    die("Cannot establish connection"); // Bind to the server
                    ldap_bind($conn) or die("Cannot bind to LDAP server.");
                    // Executing various LDAP commands.
                    // disconnect the connection
                    ldap_unbind($conn)
                    or die("Could not disconnect from LDAP server.");
          ?>
```

Retrieval of Data from LDAP Searching for records:

The ldap_search() function helps in searching a directory based on a particular filter. The general form of ldap_search() function:

resource ldap_search(resource link_idn, string base_dn, string filter
[, array attributes [, int attributesonly [, int size_limit
[, int time_limit [int deref]]]]])

After successful execution of the above command it returns a result set. Otherwise, it returns an error.

Example 28 : To retrieve the name of all the user's first name, starting with letter 'G':

```
$resultset = ldap_search($conn, "dc=scetrnd, dc=in", "givenFirstName=G*");
```

Function to sort records: To sort the result set, use ldap_sort() function. The general form of ldap_sort() function:

boolean ldap_sort(resource link_idn, resource result, string sort_filter)

Example 29 : How to sort the records:

```
<?php // connect and bind to the server
          $resultset = ldap_search($conn,$dirname, "me=K*", array("givenFirstName",
          "me")); // Sort the records with the user's first name
          ldap_sort($conn, $resultset, "givenFirstName");
          $entriesset = ldap_get_entries($conn,$resultset);
          $total = $entriesset["total"]; //loop for sorting
          for($c=0;$c<$total;$c++)
          {
                    printf("%s %s <br />", $entriesset[$c] ["givenFirstName"][0],
                              $entriesset[$c]["me"][0]);
          }
          ldap_unbind($conn);
?>
```

The output generated by this program is:

```
Kamesh
Kumar
```

Adding new entries: To add new entries in the LDAP directory, use ldap_add () function. The general form of ldap_add() function:

boolean ldap_add(resource link_idn, string dn, array entry)

Deleting LDAP data: There are two functions to delete data from the directory. They are ldap_delete() and ldap_mod_del().

Deleting entries: By using ldap_delete() function the entire entry is deleted from the LDAP directory. The general form of ldap_delete() function:

boolean ldap_delete(resource link_idn, string dn)

Example 30 : To delete an entry:

```
$dirname = “cn=srinivas kumar,ou=staff,dc=scetrnd,dc=in”;
 ldap_delete($conn, $dirname) or die(“The entry could not be deleted!”);
```

Deleting entry attributes:

By using ldap_mod_del() the value of the entity is deleted, but not entire entity. The general form of ldap_mod_del() function:

boolean ldap_mod_del(resource link_idn, string dn, array entry)

Example 31: To delete user John Wasil departments attribute:

```
$dirname = “cn=samson, ou=staff,dc=scetrnd,dc=in”;
ldap_mod_delete($conn, $dirname, array(“department”));
```

Error handling:

To handle the errors occurred due to incorrect actions of the users, use the following function. The below function displays the error “time limit exceeded”.

string ldap_err2str(int errno)

Example 32: To handle an error:

```
echo ldap_err2str (3);
```

The output generated by this example is shown here:

```
Time Limit Exceeded
```

Retrieving the most recent error number:

When interacting with a server directory, LDAP offers a list of error codes. To retrieve these codes use ldap_errno() function. The general form of ldap_errno() function:

int ldap_errno(resource link_idn)

Retrieving the most recent error message:

To retrieve the most recent error message generated during connection, use ldap_error() function. The general form of ldap_error() function:

string ldap_error(resource link_idn)

List of LDAP Error Codes

A list of few LDAP error codes is listed below:

- ✓ LDAP_TIMELIMIT_EXCEEDED: Indicates that the execution time limit specified by the client or the server was exceeded.
- ✓ LDAP_INVALID_CREDENTIALS: Indicates that the credentials supplied to the binding function were invalid.
- ✓ LDAP_INSUFFICIENT_ACCESS: Indicates that the user has no access to perform the operations.
- ✓ LDAP_BUSY: Indicates that, at this moment the LDAP server is busy to process the client request.
- ✓ LDAP_SUCCESS: Indicates that the client request completed or processed successfully.

- ✓ LDAPSIZELIMIT_EXCEEDED: Indicates that the size limit specified in search operation was exceeded.
- ✓ LDAP_COMPARE_TRUE: Indicates that the result of comparison operation is true
- ✓ LDAP_COMPARE_FALSE: Indicates that the result of comparison operation is false.
- ✓ LDAP_ADMINLIMIT_EXCEEDED: Indicates that the limit set to the LDAP server by the administrator was exceeded.

REVIEW QUESTIONS

1 List out the advantages of MySQL.

2 Write a query to add 5 days to the today's date.

3 Write a query to display the version of your MySQL. Create a table called book details with columns book_id and book_title. Write a query to display book_titles starting from 20th book_id to 50th book_id.

4 Give a brief explanation about SELECT command.

5 Create the following table and write the following queries.

Empid	Empname	Salary	Bonus	EmailID	Deptno
100	Rajesh	10,000	2000	Rajesh123@gmail.com	10
101	Ramesh	20,000	2000	rameshaddala@yahoo.com	10
102	Srinivas	19,500	1000	srinivasreddy@rediffmail.com	20
103	Geetha	8,000	3000	geethaanjali@yahoo.com	40
104	Satya	9,500	−	satya12ab@gmail.com	30
105	Srilekha	21,000	−	lekhadevi1234@gmail.com	40

(a) Write a query to display total salary paying by the company to their employees.
(b) Write a query to display the empname, empid who is getting maximum salary.
(c) Write a query to display the employee details who are getting a second maximum salary.
(d) Write a query to display the employee record sorted by their names.
(e) Write a query to concatenate the strings 'new' and 'Delhi'.
(f) Explain the differences between TEXT and BLOB?

6. Write a command to display a unique value from the set of duplicate values in a table.

7. Create a table called course_details with columns course_id (primary key), course_name, course_teachername, course_studentname, course_fee, course_duration in MySQL database. Write a PHP script to list the course name, the teacher name who is dealing that course and the student names who are studying that course by connecting to the database through a PHP script.

8 Write a command to describe the structure of the table in MySQL.

9 Write a query to display the length of the string 'I am reading Open Free Software Book'.

10 Write a query to display the difference between two given dates of a month.

11 How many values do the SET() function stores in MySQL?

12 Write a PHP Script to do the following task: A PHP form containing username and password fields upon submission, it must check the MySQL database whether the given username and password exists or not. If exists, display a welcome page. If not, display 'INVALID USERNAME/PASSWORD' message and redirect to the login form.